ACCIONES DE HARDENING PARA MEJORAR LA SEGURIDAD DE LA INFORMACIÓN CUANDO SE USAN LOS SERVICIOS DE HTTP, LDAP, SSH Y SMTP

JOSÉ DAVID LEÓN RODRÍGUEZ

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D.C.
2018

ACCIONES DE HARDENING PARA MEJORAR LA SEGURIDAD DE LA INFORMACIÓN CUANDO SE USAN LOS SERVICIOS DE HTTP, LDAP, SSH Y SMTP

JOSÉ DAVID LEÓN RODRÍGUEZ

Monografía para optar el título de
Especialista en Seguridad Informática

Esp. Ing. Freddy Enrique Acosta
Director del Proyecto

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ D.C.
2018

Nota de aceptación:

_____
_____
_____
_____
_____
_____

Firma del presidente del jurado

Firma del jurado

Firma del jurado

Bogotá D.C (30, 5, 2018)

A mis padres y hermano que siempre han sido una luz en la oscuridad, acompañándome y creyendo en mí en todo momento.

José David León Rodríguez

AGRADECIMIENTOS

El Autor, *José David León Rodríguez* expresa sus agradecimientos a:

Ing. Diego Jurado Pallarés. Por su excelente apoyo en cuanto a la aclaración de conceptos sobre HoneyPots y HoneyNets, su colaboración en la búsqueda de fuentes bibliográficas y en general por su apoyo en la precisión de los objetivos del proyecto y su viabilidad.

Esp. Ing. Freddy Enrique Acosta. Por su dedicada colaboración y apoyo como tutor, director y guía durante el desarrollo del presente trabajo. Por sus enseñanzas sobre Seguridad Informática y de la Información impartidas en diferentes cursos de la Especialización. Gracias a su excelente trabajo como profesor, ha sido posible la definición del presente proyecto corrigiendo temas de semántica, sintaxis, estructura y marco teórico en general.

Alexis Damian Gonzáles López. Por su empeño dedicado al conocimiento de la Seguridad Informática en general, su acompañamiento durante este proyecto ha sido imprescindible permitiéndome aterrizar muchos de los conceptos en seguridad y redes que no he entendido en primer vistazo. Su disposición en cuanto al tema de investigación y conocimiento de *HoneyPots,* un mundo desconocido tanto para él como para mí durante el inicio de este proyecto.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

GLOSARIO

ACTIVO: componente de un Sistema de Información o infraestructura organizacional que puede tener un valor específico para el desarrollo de los procesos de negocio dentro de un ambiente organizacional[1].

AMENAZA: se considera amenaza cualquier tipo de acción o procedimiento que pueda causar un daño inminente a un Sistema de Información en específico. Las amenazas afectan los sistemas, por lo tanto, pueden perjudicar los activos o procesos de una organización[1].

CANAL O CHANNEL: se refiere a cualquier tipo de medio físico o lógico en el cual se pueda transferir datos para un protocolo específico. Este concepto es ampliamente usado en los protocolos que pueden encapsular múltiples especificaciones en la Capa de Transporte[2].

CLIENTE: computadora dispuesta para el consumo de los Servicios informáticos prestados por el Servidor. En un esquema de Cliente/Servidor realiza las peticiones al Servidor, para que éste luego de haber procesado una respuesta, retorne un resultado de valor que el Cliente pueda aprovechar[2].

DECEPTION O ENGAÑO: en el campo de las HoneyPots se trata de tácticas especializadas que buscan simular el comportamiento real de un software, programa informático o en la mayoría de los casos un Servicio; otorgando vulnerabilidades intencionalmente configuradas[3]. El atacante, confiando que puede aprovecharse de esta brecha de seguridad; cae en dicha simulación bajo un engaño donde al realizar sus actividades maliciosas, en realidad está siendo analizado. Las

---

[1] ESPAÑA, GOBIERNO DE LA REPÚBLICA. MAGERIT – Versión 3.0 Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información, Libro 1 – Método [online]. 1 ed. España. Octubre, 2012. [citado 13 mar., 2018]. Método de Análisis de Riesgos, Conceptos paso a paso. Disponible en Internet: < https://www.ccn-cert.cni.es/documentos-publicos/1789-magerit-libro-i-metodo/file.html >

[2] International Organitation for Standardization ISO. ISO/IEC 7498-1:1994 Information Technology – Open System Interconnection – Basic Reference Model: The Basic Model [online]. Switzerland. Junio, 1996. [citado 13 mar., 2018]. Introduction to Open Systems Interconnection (OSI), Definitions. Disponible en Internet: < https://www.iso.org/standard/20269.html >

[3] LAKHANI, Amit. Deception Techniques Using HoneyPots. Londres: University of London. 2007 p 46-49.

tácticas de Deception, llamadas Engaño en castellano; son la clave para el funcionamiento de las HoneyPots.

DISPOSITIVOS CASEROS: componentes computacionales que puedan ser de utilidad para el desarrollo del proyecto ya que pueden ser fácilmente adquiridos y no representan un costo significativo. Estos componentes pueden ser: memorias volátiles, memorias usb, discos duros, procesadores de bajo costo, interfaces de red, entre otros.

EVIDENCIA: cualquier tipo de registro, documento o archivo digital que sea recogido por los Sistemas de Monitorización y puedan ser útiles para su posterior Análisis.

EXPLOIT: puede referirse a la palabra como verbo o sustantivo. Si se refiere a la palabra como verbo, explotar, trata sobre aprovecharse de un hueco de seguridad de un Sistema de Información para ejecutar una característica o comportamiento no esperado. Si se refiere al sustantivo, exploit, hace referencia al código o programa que se utiliza como Prueba de Concepto para demostrar la existencia de un hueco de seguridad y aprovecharla[4].

GATEWAY O PUERTA DE ENLACE: dispositivo, el cual generalmente responde al comportamiento de un router, por el cual se puede tener acceso a Internet.

HONEYNET O RED DE SISTEMAS DE SEÑUELO: composición de distintas HoneyPots en una misma red. Este concepto puede otorgar mayor cobertura al estudio de amenazas para un entorno empresarial, aunque su aplicación puede ser bastante engorrosa dependiendo del tipo de despliegue para cada sistema de señuelo, su número y relación con la red original de la organización[5]. Este concepto suele ser utilizado exclusivamente por expertos en Seguridad Informática.

HONEYPOT O SISTEMAS DE SEÑUELO: programa informático o componente de software vulnerable por defecto, el cual tiene como objetivo capturar acciones maliciosas imitando el comportamiento de su homónimo original[5]. Esto quiere decir que básicamente se trata de una solución, en lo general de tipo software que dispone de una baja seguridad imitando el comportamiento de otro tipo de software

[4] ANLEY, Chris; HEASMAN, John; RICHARTE, Gerardo et al. The Shellcoder's Handbook, Discovering and Exploiting Security Holes. En: Chapter 1 Before you Begin. 2 ed. Estados Unidos: Wiley Publishing Inc, 2007. p 3-10. ISBN 978-0-470-08023-8.

[5] JURADO PALLARÉS, Diego. Op. cit., 20 p.

haciendo creer a una amenaza o atacante que se tratarse de una brecha de seguridad o vulnerabilidad expuesta, de esta manera se atraen este tipo de acciones maliciosas para recolectarlas, registrarlas y/o analizarlas. En español son conocidas como sistemas de señuelo.

HOST: computadora o dispositivo que se encuentra en la Capa de Aplicación ya sea del Modelo TCP/IP o Modelo OSI. Un host puede corresponder a un dispositivo virtualizado o físico; dependiendo de esta categoría se puede hablar de un guest host (Host Invitado) o simplemente host (Anfitrión)[6].

IMPACTO: nivel de daño causado por la manifestación de una amenaza sobre una vulnerabilidad[7]. Puede ser medido por diferentes factores, teniendo en cuenta la dimensión de la amenaza, por ejemplo: Cantidad de Archivos Borrados, Cantidad de Máquinas Dañadas, Cantidad de Cuentas de Usuario Robadas, entre otros.

INTRUSO O ATACANTE: sujeto que, implementando técnicas manuales o automáticas, haciendo uso de malware o sin él; accede de manera ilegal a un Sistema de Información para extraer, corromper o tergiversar información deliberadamente[7].

MALWARE: programa informático con fines maliciosos. Está destinado a la destrucción, manipulación, filtración, espionaje y/o secuestro de la información de la víctima. Según su actividad específica puede ser clasificado en diferentes categorías[4].

MODEM: dispositivo que tiene la capacidad de convertir una señal en otra. Dichas señales pueden ser transformadas de análogas a digitales, o viceversa, dependiendo de la tecnología implementada[6].

PAYLOAD: en redes, hace referencia a la porción de en un paquete de red que contiene información relevante y dinámica donde puede incluirse datos del usuario específico[4]. En malware suele referirse a la sección de código específica donde se aplica la carga de código malicioso, destinado a explotar una vulnerabilidad o hueco de seguridad.

[6] International Organitation for Standardization ISO. Op. cit., p. 19.

[7] ESPAÑA, GOBIERNO DE LA REPÚBLICA. Op. cit., p. 19.

PROTOCOLO DE COMUNICACIÓN DE RED: reglas y estándar para el intercambio de mensajes entre una máquina o dispositivo conocido, en la mayoría de los casos como Cliente, y una máquina Servidor. Se puede pensar que un Protocolo de Comunicación de Red es una abstracción sobre un tipo de lenguaje especial entre dos hosts[6].

PRUEBA DE CONCEPTO POC: código destinado a demostrar un test de seguridad específico, el cual no tiene que ser necesariamente ofensivo. Las pruebas de concepto pueden ser usadas también para demostrar la existencia de una defensa. Normalmente, son asociadas con Exploits[8].

PUERTO: interfaz de tipo lógico con la cual un programa o proceso específico puede comunicarse a través de la red. Normalmente, quien está encargado de la asignación de puertos es el Sistema Operativo, aunque en ciertos casos específicos también puede ser el Firmware[6].

RIESGO INFORMÁTICO: probabilidad de ocurrencia de una amenaza sobre una vulnerabilidad. Puede ser medido de manera cualitativa o cuantitativa[7].

ROUTER: dispositivo de Red destinado a la conexión del tráfico entre dos redes. En ambientes domésticos también se le conoce como modem, ya que usualmente transforma las señales radiales en pulsos eléctricos que pueden ser transferidos mediante medios cableados[6].

SERVICIO: recurso específico sobre el cual se atiente una o varias solicitudes del cliente relacionadas con un programa informático en específico. Usualmente los Servicios no cuentan con interfaz gráfica ya que sólo atienden solicitudes realizadas por medio de protocolos de comunicación entre procesos o computadoras. Los Servicios son mantenidos por la computadora conocida como Servidor[6].

SERVICIOS EN LA NUBE: la computación en la nube es básicamente una arquitectura de computadores distribuidos donde los servicios de red suelen estar en potestad de ciertas compañías proveedoras a internet. Los servicios de red ofrecidos suelen ser aquellos comunes en ambientes organizacionales como: mail, web, dns, directorio activo, vpn, entre otros. Se diferencia de los VPS ya que no es estrictamente necesario el acceso con privilegios de administrador a la máquina que presta los servicios de red, de hecho, es probable que el proveedor ofrezca un panel de control bastante limitado donde se puedan configurar los servicios contratados.

[8] ANLEY, Chris; HEASMAN, John; RICHARTE, Gerardo et al. Op. cit., p. 20.

Los proveedores de Servicios en la Nube normalmente determinan algunas limitantes sobre acuerdos de servicio y políticas de uso[9].

SERVIDOR: computadora dispuesta para la prestación de Servicios informáticos; estos normalmente son accedidos mediante Internet siguiendo con el esquema de Cliente/Servidor. El Servidor orquesta los Servicios solicitados por el cliente, registrando los accesos, errores y otorgando contingencia a posibles eventos inesperados[10].

SISTEMAS DE DETECCIÓN Y PREVENCIÓN DE INTRUSOS: los Sistemas de Detección y Prevención de Intrusos también llamados por sus siglas IDS e IPS respectivamente; son soluciones informáticas que permiten monitorear las acciones llevadas en un Sistema Operativo o la red en general para identificar acciones maliciosas o ejecución de ciertos tipos de malware en específico[11]. Dependiendo de su alcance puede ser orientado a host (HIDS-HIPS) u orientado a red (NIDS-NIPS). Básicamente la diferencia entre un Sistema de Detección de Intrusos (IDS) y un Sistema de Prevención de Intrusos (IPS) radica en el manejo de respuesta a eventos; el primero únicamente alerta, el segundo puede tomar acciones preventivas. En la actualidad, estos dos conceptos se encuentran bastante relacionados, y en ocasiones es un tanto complicado diferenciarlos.

SISTEMAS DE MONITORIZACIÓN: elementos físicos (hardware) y/o lógicos (software) que permiten vigilar el entorno de ejecución de un sistema en específico. Para el desarrollo del proyecto se entiende por Sistemas de Monitorización, todo tipo de programa informático que ayude a verificar el estado del Sistema Operativo, sus directorios, accesos, entre otros. Esto quiere decir, que inicialmente se puede inferir el uso de dos Sistemas de Monitorización, los sniffers y los Sistemas de Detección y Prevención de Intrusos (IDS-IPS)[11].

SNIFFER: elemento usado para la recolección de datos entre dos puntos de comunicación. Puede ser físico (hardware) o lógico (software), su objetivo es

---

[9] Microsoft Azure. What is cloud computing? A beginner's guide [online]. Marzo, 2018. [citado 13 mar., 2018]. Uses of cloud computing. Disponible en Internet: < https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/ >

[10] International Organitation for Standardization ISO. Op. cit., 19 p.

[11] CHAKRABORTY Nilotpal. Intrusion Detection System and Intrusion Prevention System: A Comparative Study [online]. India. Mayo, 2013. [citado 13 mar., 2018]. Capítulo 5: Intrusion Detection System y Capítulo 6: Intrusion Prevention System. Disponible en Internet: < https://pdfs.semanticscholar.org/c91d/d6f155cdbd0c72017a8413c74f1953b517c3.pdf >

capturar los paquetes intercambiados entre un punto A y un punto B para observar la manera en que se pueden intercambiar información[11].

TÚNEL DE RED O TUNNELING: técnica aplicada en redes para encapsular un protocolo mediante el uso de otro tipo de protocolo. Es bastante utilizado en el concepto de VPNs y en algunas ocasiones para crear canales de comunicación secretos que puedan evadir el uso de Firewalls[12]. Algunos de los más conocidos, involucran al protocolo SSH para proteger la información de los datos intercambiados y evadir las posibles detecciones de sistemas IDS/IPS.

VIRTUAL PRIVATE SERVER VPS: se trata de un servicio disponible en internet donde se dispone de un único Servidor Físico dividido en varios servidores virtuales, de esta manera se puede distribuir de forma óptima los recursos del servidor físico. Este servicio se distribuye a través de internet con diferentes propósitos, normalmente los usuarios o clientes tienen privilegios de administrador de su VPS lo cual lo convierte en servicio altamente configurable y personalizable. Los proveedores de VPS normalmente determinan algunas limitantes sobre acuerdos de servicio y políticas de uso[13].

VULNERABILIDAD: característica inherente de un activo o componente de Sistema de Información, la cual puede encontrarse expuesta a una amenaza en específico, esta amenaza puede materializarse sobre la vulnerabilidad en cualquier momento; es por esta razón que las vulnerabilidades pueden ser peligrosas[14].

[12] KARPESKY. What is a Tunneling Protocol? [online]. Marzo, 2018. [citado 13 mar., 2018]. Disponible en Internet: < https://www.kaspersky.com/resource-center/definitions/tunneling-protocol >

[13] MAURER, Ashley. Virtual Private Servers – An Introduction (And 3 Reasons You Might Need One). Abril, 2017. [citado 13 mar., 2018]. Disponible en Internet: < https://www.a2hosting.com/blog/virtual-private-servers/ >

[14] ESPAÑA, GOBIERNO DE LA REPÚBLICA. Op. cit., p. 19.

RESUMEN

El presente proyecto tiene como objetivo demostrar la importancia de los *HoneyPots* o sistemas de señuelo para los ambientes organizacionales. Para ello se toman cuatro de los servicios más comunes en toda organización, los cuales son: HTTP, LDAP, SSH y SMTP. Se observa, además el comportamiento de cada uno de estos protocolos, ya que es necesario para entender cómo deberían comportarse las peticiones y respuestas realizadas a los servicios. Así mismo, se listan las amenazas más conocidas para cada uno de ellos, lo cual permitirá sistematizarlas y discriminarlas.

Por cada servicio se configuran cuatro *HoneyPots* respectivamente en un servidor virtual que pueda ser expuesto para recolectar la información. Para apoyar la recolección de los datos se instalan y configurar tres herramientas de monitorización. Estas herramientas corresponden a un Sistema de Detección de Intrusos IDS, un Antivirus y un Sniffer; siendo su labor capturar información que los *HoneyPots* puedan dejar escapar.

Una vez se recolectan los datos se procede a practicar un análisis de la información para así discriminar cuáles son los ataques identificados. Luego de ello se concluye una serie de acciones de defensa que se pueden adoptar en cuanto a la red, configuración e instalación de los servicios para así mejorar la Seguridad de la Información. De esta manera se demuestra cómo influye el uso de los *HoneyPots* en la definición del modelo de seguridad en las redes organizacionales.

ABSTRACT

Project has as objetive demostrate importance of Honeypots or deception techniques to organizational environments. To make it I have taken four of more common services in a Network Company, which are: HTTP, LDAP, SSH and SMTP. During development Project I studied each protocol behavior because is neccesary understand how should response different services to a common request. Futher Project recolect most known threats, which allow to discriminate and systematize collected HoneyPots and other data.

I configured four HoneyPots to each service in a Virtual Server that can expose to recolect data. To support recolection process I installed and configured three network monitor tools. This tools was a Intrution Detection System IDS, a Antivirus and a sniffer; whose work was capture data that could be invisible to HoneyPots.

Once datas is collected a data analysis was performed to discriminate which are identified threats. Then some actions to protect services are listed which include hardening on network, settings and instalation of each services to improve Information Security. In this way Project demostrate how influ

INTRODUCCIÓN

Alrededor de los años 1989 un astrónomo reconocido llamado Cliff Stoll habría sido trasladado desde su habitual observatorio hasta las instalaciones de la Universidad de Berkeley, uno de los grandes focos de la informática para el siglo anterior. En su nuevo trabajo se le encargó, en especial, el uso de un Sistema Operativo Unix el cual alojaba un importante sistema estadístico de la universidad. Cliff se habría percatado de un inusual comportamiento sobre algunos datos en las finanzas del sistema. Por alguna extraña razón siempre se extraviaban unos 75 centavos cada semana[15]. Para investigar a fondo el error, el astrónomo con poca experiencia en informática se ingenió la forma de monitorear las actividades de los diferentes usuarios. Finalmente, la recolección de estos datos arrojó resultados, demostrando que existía un hueco de seguridad en el complejo sistema que permitía a cualquiera modificar los datos relacionados a la contabilidad financiera. Un año después, en 1990 Bill Cheswick ansioso investigador, deseaba resolver un enigma relacionado a una serie de eventos anormales que acontecían sobre un dispositivo "altamente seguro" que se encontraba en una red específica. El *hardware* permitía el acceso mediante los servicios FTP, SMTP y Telnet básicamente. Mediante algunos trucos y señuelos, el investigador determinaría pronto que un usuario de la red estaba enviando paquetes maliciosos hacia los diferentes servicios[16]. La práctica de Bill identificaría huecos de seguridad en el dispositivo estudiado y retornaría la conclusión que muchos de estos instrumentos ofrecen servicios y protocolos que son por defecto inseguros[17].

Con estos dos importantes trabajos nace el concepto de *HoneyPot.* Un *HoneyPot* puede ser cualquier elemento que simule ser real mintiendo al usuario acerca de sus vulnerabilidades para atraer usos no autorizados o ilícitos. Es un mecanismo que imita el comportamiento de un sistema de información pero que en el fondo está dispuesto como una carnada para los usuarios malintencionados[18]. Permiten a su propietario adoptar medidas de defensa o realizar estudios sobre los vectores de ataque recogidos. Esto ofrece algunos beneficios notorios como disponer de una fuente de primera mano y además confiable sobre acciones perjudiciales o revelar un nuevo estudio sobre tendencias desconocidas de *malware* y explotación de vulnerabilidades.

---

[15] STOLL, Cliff. The Cuckoo's Egg. Cambridge: Universidad de Harvard, 1998. 254 p.

[16] CHESWICK Bill. An Evening with Berfer In Which a Cracker is Lured, Endured and Studied. Nueva Jersey: AT&T Bell Laboratories, 1991. 11 p.

[17] JURADO PALLARÉS, Diego. Op. cit., 20 p.

[18] PETER, Eric y SCHILLER Tood. A Practical Guide to HoneyPots [online]. Washington D.C. Mayo, 2009. [citado 13 mar., 2018]. Disponible en internet: < https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/index.html >

En una red organizacional los *HoneyPots* pueden ayudar a sus administradores a identificar cuáles políticas se deben incluir en los diferentes sistemas de seguridad para proteger la información. En el presente trabajo se pretende demostrar cómo los *HoneyPots* pueden ayudar a verificar amenazas sobre los servicios de los protocolos HTTP, LDAP, SSH y SMTP para así, por medio de un análisis de los datos, se determine acciones correctivas que permitan blindar cada uno de estos servicios. Una labor similar a la realizada por Bill Cheswick y Cliff Stoll. Para ello se consultó herramientas especializadas en *HoneyPots* que se concentran en cada uno de estos servicios, posteriormente se investigó sobre otros elementos de monitorización como Sistemas de Detección y Prevención de Intrusos IDPS, Antivirus, Sniffers y otros que ayudaron a la labor de los Sistemas de Señuelo. Finalmente, se tomó los datos de cada uno de los sistemas observando cuáles fueron los vectores de ataque registrados y qué tipo de protecciones o defensa se pueden adoptar en un escenario real para así prevenirlos. Este trabajo destaca la importancia del uso de las *HoneyPots* en los ambientes organizacionales, como elemento clave para el estudio de los intrusos.

1 DEFINICIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Las redes en las organizaciones se encuentran en un constante estado de amenaza debido a la incidencia y ocurrencia de conocidos o incluso nuevos vectores de ataque. Durante el año 2017 se identificaron diferentes amenazas cuyo objetivo era el sector empresarial. Espionaje, secuestro de información, infección por medio de corrupciones de memoria, movimientos a través de redes internas, puertas traseras y demás; fueron algunas de las amenazas más frecuentes[19]. Se predice que en el año 2018 la incidencia de amenazas será más diversa y aún peor, los vectores de ataque cambiarán haciéndose más sofisticados y se espera que evolucionen algunas amenazas para adquirir capacidades infecciosas o virales[19].

En algunas empresas, en especial aquellas clasificadas como *PYMES,* la seguridad suele ser subestimada, infravalorada o en ciertos casos desconocida completamente para el modelo de negocio. Esto convierte a las compañías en un blanco predilecto para el cibercrimen[20]. En muchos casos las compañías únicamente se concentran en ofrecer medidas de defensa frente a amenazas que probablemente nunca se han manifestado en sus instalaciones o sobre las cuales no tienen suficiente conocimiento.

De forma irónica, los mecanismos de defensa seleccionados podrían no ser adecuados; puesto que se estarían definiendo reglas de seguridad difíciles de administrar, obviando algunas que en realidad son indispensables o determinando normas que esencialmente son innecesarias[21]. Esto nos lleva a una conclusión, el diseño de seguridad de la red está basado en conjeturas.

Las consecuencias de un diseño de seguridad de red basado en conjeturas pueden ser desastrosas ya que se podrían dejar brechas las cuales pueden ser completamente desconocidos para los Administradores de Red. En cualquier

---

[19] RAIU, Costin; GUERRERO S, Juan A y BAUMGARTNER, Kurt. Kaspersky Security Bulletin: Threat Predictions for 2018 [online]. Secure List nov, 2017. [citado 21 mar., 2018]. Disponible en Internet: < https://securelist.com/ksb-threat-predictions-for-2018/83169/ >

[20]BRANDON, John. Why your Business might be a Perfect target for Hackers [online]. Inc.com nov, 2017. [citado 21 mar., 2018]. Disponible en Internet: < https://www.inc.com/magazine/201312/john-brandon/hackers-target-small-business.html >

[21] POSEY, Brein. 10 common Network Security design flaws [online]. Tech Republic oct, 2009. [citado 21 mar., 2018]. Disponible en Internet: < https://www.techrepublic.com/blog/10-things/10-common-network-security-design-flaws/ >

momento este tipo de vulnerabilidades puede ser fácilmente aprovechados comprometiendo a la red en su totalidad y la infraestructura de la organización.

Estas consecuencias son provocadas simplemente por el desconocimiento, el cual a su vez es causado por ignorar los pasos precisos con los cuales una amenaza es desarrollada. Los mecanismos de defensa en redes suelen no contar con las características suficientes para recoger esta información. Es necesario entonces, determinar otro tipo de soluciones que permitan recolectar datos sobre la manifestación de las amenazas para así determinar un verdadero diseño de seguridad de red que se justifique mediante un análisis sólido. Para recolectar este tipo de información es necesario simular sistemas reales de producción, lo cual abre una nueva problemática que es buscar soluciones que se ajusten a las necesidades de la organización sin exponerla a algún tipo de vulnerabilidades. Estos mecanismos de señuelo o carnada deben configurarse para ser un beneficio hacia la totalidad de la red y no significar un riesgo para la misma.

1.2 FORMULACIÓN DEL PROBLEMA

Para estudiar el comportamiento de las amenazas de los diferentes servicios de las redes organizacionales se propone el uso de los *HoneyPots.* Estos son elementos especializados en la monitorización, recolección de información y simulación de sistemas de producción.

Teniendo en cuenta la gran diversidad de componentes que pueden jugar un rol en una red organziacional, se hace necesario tomar una muestra de algunos dispositivos, protocolos o servicios. Sobre esta muestra se podría establecer la configuración de los *HoneyPots* y definir su propósito.

En primer lugar, se evalua la posibilidad de crear sistemas de señuelo que simulen completamente el comportamiento de un *host* dentro de una red, aunque esto podría arrojar datos imprecisos. También se puede recolectar información por medio de sistemas de monitorización como lo son los Sistemas de Detección y Prevención de Intrusos IDPS, *sniffers,* Antivirus y demás. Esta segunda opción tiene la ventaja de poder capturar amenazas en ambientes productivos sin necesidad de configurar un sistema de señuelo, aunque nuevamente como la opción anterior podría arrojar datos variados que difícilmente pueden ser sistematizados.

Con la finalidad de dar una facilidad sobre la sistematización de los datos se puede partir de un conjunto limitado de servicios, aquellos que correspondan a algunos de los más usados en las redes organizacionales. De esta manera se estudian componentes de red importantes en un ambiente organizacional y se limita el análisis de las amenazas específicamente dirigidas a estos servicios.

Así pues, se seleccionan los protocolos HTTP, LDAP, SSH y SMTP los cuales se pueden analizar mediante una correcta configuración de los *HoneyPots*. Si a esta idea se le adjunta la selección de herramientas de monitorización, se puede realizar un análisis tanto de los datos de los *HoneyPots* como los demás mecanismos de seguridad.

Finalmente, tomando como insumo los datos recolectados se plantea el siguiente interrogante ¿Qué tipos de acciones de defensa o *hardening* se pueden configurar en los servicios estudiados para protegerlos de las amenazas identificadas? Dicho interrogante incluye otra pregunta importante, la cual es: ¿Cuál es la forma de configurar los *HoneyPots* para recolectar la información y así poder identificar las acciones maliciosas?

2 JUSTIFICACIÓN

Los HoneyPots ofrecen las características de simulación y recolección de datos que pueden ser necesarias al momento de estudiar las amenazas de redes. Su estructura y principios constan en el seguimiento de los pasos ejecutados por un atacante, aparentando ser un servicio o sistema real de producción[22]. Los datos recogidos por las HoneyPots pueden ayudar a la adopción de medidas de seguridad que en realidad sean comprendidas por los Administradores de Red y que, además sean necesarias para la organización. Los Administradores de Red pueden aprender de sus propios servicios con ayuda de estos mecanismos, identificando cuáles son los vectores de ataque que se desarrollan con mayor frecuencia en sus redes organizacionales.

Ahora, en muchas organizaciones los servicios de red más implementados son: Web; el cual corresponde al protocolo HTTP, Directorio Activo; el cual corresponde al protocolo LDAP y otros, Administración Remota por medio de Terminal; el cual puede corresponder al protocolo SSH y Correos Electrónicos; el cual corresponde a los protocolos SMTP, POP3 e IMAP[23]. Estos servicios son clave para el funcionamiento y administración de la información en las organizaciones. Esta misma característica les convierte en objetivos codiciados por los diferentes criminales que quieren irrumpir en la infraestructura de una organización. Es natural encontrar algunos mecanismos que intentan defender estos servicios, sin embargo, muchos de ellos son definidos en base a conjeturas como se ha descrito en la sección del Planteamiento de Problema.

La configuración de HoneyPots para estos servicios puede ayudar a fomentar una base sólida en el diseño de defensa de la red en general, ya que corresponden pequeños eslabones de una cadena la cual es la seguridad de la red en su totalidad. Por esto, la obtención de resultados basados en vectores de ataque destinados a cada uno de ellos de forma específica permite una sistematización las amenazas, las cuales podrían estar interrelacionadas. Para lograr esta sistematización es necesario disponer de distintas HoneyPots ya que no es lo mismo referir un ataque hacia una aplicación web que una difusión de malware por medio de SPAM, por ejemplo.

Los resultados recogidos para los HoneyPots en estos servicios, pueden ser usados para definir las medidas de protección necesarias para blindar a cada uno de ellos. Cambios en la configuración del servicio, definición de nuevas reglas en el Firewall,

---

[22] JURADO PALLARÉS, Diego. Op. cit., 20 p.

[23] RAMDEV. Ten most frequently used Linux networking services, in enterpirse unix networks [online]. Unix admin schoo may, 2015. [citado 21 mar., 2018]. Disponible en Internet: < http://unixadminschool.com/blog/2012/05/ten-most-frequently-used-linux-networing-services-in-enterprise-networks/ >

políticas de Antivirus o Sistemas de Detección y Prevención de Intrusos; son algunas de las medidas que podría este breve análisis. Blindar algunos de los servicios más conocidos de red puede, además, otorgar las pautas para referir una actitud de defensa más especializada en la red en general. En conclusión, analizar las amenazas de los servicios para los protocolos HTTP, LDAP, SSH y SMTP para así mejorar sus capacidades de seguridad puede ser un acercamiento hacia la definición de un modelo de seguridad de red basado en evidencias y datos palpables.

3 ALCANCE Y LIMITACIONES

3.1 ALCANCE

El presente proyecto de grado el cual se encuentra enmarcado dentro de los proyectos de Seguridad de la Información, lo que preteden es verificar ataques a los servicios de los protocolos HTTP, LDAP, SSH y SMTP que permita determinar acciones de *hardening* para mejorar la Seguridad de la Información en las organizaciones que usen este tipo de servicios.

Las acciones de protección pueden abarcar el cambio de la configuración de cada uno de los servicios, definición de normas en Sistemas de Detección y Prevención de Intrusos, Antivirus o Firewall o incluso incluir nuevos elementos de seguridad en redes que no estén contemplados inicialmente en los objetivos.

La recolección de información sobre las amenazas se realizará por medio de *software* de tipo de *HoneyPots* centrada en los servicios de los protocolos HTTP, LDAP, SSH y SMTP. Estas *HoneyPots* estarán acompañadas de al menos 3 tipos de *software* de monitorización, los cuales pueden ser Sistemas de Detección y Prevención de Intrusos IDS e IPS, Antivirus y/o Sniffers. El montaje del laboratorio puede ser en un *Virtual Private Server* o un dispositivo casero como *Raspberrys,* Computadoras u otros; que se encuentren configurados por medio del *ISP* para tener acceso a Internet o la WAN.

3.2 LIMITACIONES

Es conveniente resaltar que el desarrollo del presente proyecto de grado no abarcará aspectos como los que se definen a continuación:

- El tiempo de exposición de las *HoneyPots* únicamente será de tres semanas.
- Únicamente se estudiarán los servicios para los protocolos HTTP, LDAP, SSH y SMTP. No se tendrán en cuenta configuraciones avanzadas de los mismos.
- Los servicios estudiados se configurarán e instalarán en Máquinas Virtuales aprovechando las capacidades de Virtualización, preferiblemente en Sistemas Operativos basados en el Kernel de Unix/Linux. Sin embargo, el

autor puede usar Sistemas Operativos Windows si lo considera necesario. En caso de usar Windows, el licenciamiento será únicamente por período de evaluación.

- El autor puede decidir si reemplaza un método de virtualización por contenedores de procesos, en caso de considerarlo necesario para mejorar el *performance* del laboratorio.

- El *software* usado para la virtualización o contenedores de procesos contará únicamente con licenciamiento libre u OpenSource.

- Debido a la naturaleza de cierto tipo de amenazas, es posible que no se obtengan datos 100% confiables. Algunas amenazas avanzadas suelen confundir las técnicas de señuelos y requieren de tácticas más especializadas como lo son las *HoneyNets*, las cuales no son abarcadas por el proyecto.

- En caso de no identificarse un *software* específico para la configuración de *HoneyPots* de alguno de los protocolos o servicios, el autor puede configurar alguno de los servicios para crear su propia *HoneyPot* de alta interacción imitando un sistema de producción.

- El *software* que apoyará a los diferentes *HoneyPots,* el cual atañe a los Sistemas de Detección y Prevención de Intrusos IDS e IPS, *Sniffers* y/o Antivirus únicamente será de licenciamiento libre u OpenSource.

- La disponibilidad del montaje o laboratorio, debido a limitantes o restricciones en el servicio contratado o usado, podría no ser de 24/7 (24 horas por 7 días a la semana).

- Debido al corto tiempo establecido en el cronograma, las acciones de *hardening* no podrán ser configuradas y mucho menos verificadas por el autor. Quien desee aplicarlas en un ambiente de producción debe cerciorarse de su pertinencia y validar su correcto funcionamiento.

4 OBJETIVOS

4.1 OBJETIVO GENERAL

Verificar ataques a los servicios de los protocolos HTTP, LDAP, SSH y SMTP a través de un HoneyPots, que permita determinar acciones de hardening para mejorar la seguridad de la información en una organización que usen este tipo de servicios.

4.2 OBJETIVOS ESPECÍFICOS

- Configurar las HoneyPots usando técnicas de virtualización para los servicios de los protocolos HTTP, LDAP, SSH y SMTP.

- Verificar mínimo tres herramientas de software libre existentes en el mercado, que permitan realizar la monitorización a través del HoneyPot de ataques a los servicios de los protocolos HTTP, LDAP, SSH y SMTP.

- Registrar los diferentes ataques a los cuales están expuestos los servicios de los protocolos HTTP, LDAP, SSH y SMTP.

- Proponer las acciones de hardening que deben ser aplicadas a los servicios de los protocolos HTTP, LDAP, SSH y SMTP.

5 MARCO REFERENCIAL

5.1 MARCO TEÓRICO

5.1.1 HoneyPots o Sistemas de Señuelo

5.1.1.1 Objetivos de un HoneyPot

Un *sistema de señuelo* puede tener objetivos especiales según sea su construcción, configuración y despliegue. A continuación, se enumeran algunas de las finalidades más conocidas para un *HoneyPot.*

- Detectar al Enemigo: Durante un ataque informático se puede encontrar con cualquier tipo de enemigos, desde programas automatizados, hasta *Scripting Kiddies, Hackers* o *Espías y Agentes Profesionales.* Una de las estrategias más conocidas frente a los ataques recurrentes es colocar una carnada para registrar las actividades de la amenaza[24]. A diferencia de los mecanismos de auditoría que otorgan los Sistemas Operativos y algunos Servicios, disponer de un *HoneyPot* en el momento del atacante ayudará a detectar cuáles son las conexiones iniciales del atacante; con esto se puede indagar sobre su Geolocalización, Dirección IP, método de entrada (Interno o Externo a la Red), tramas de red utilizadas; entre otros.

- Reportes estadísticos y tendencias de ataque: En los últimos años, diversos tipos de *malware* han sido detectados en todo el mundo. Algunos de ellos como la conocida amenaza *Stuxnet*[25]*,* que afectó a las centrales eléctricas de países del oriente medio; tenían un comportamiento específico. Estos comportamientos lograban que el *malware* únicamente se ejecutara en un país concreto, únicamente para las centrales nucleares[24]. En aquel caso de haberse implementado una *HoneyPot* probablemente se habría identificado rápidamente que el objetivo de la amenaza en específico eran precisamente

[24] SOKOL, Pavol; PEKARČK Patrik y BAJTOŠ Tomáš. Data Collection and Data Analysis in HoneyPots and Honeynets. Košice: Facultad de Ciencias de la Computación Universidad de Pavol Jozef Safárik, 2015. 16 p.

[25] PANDA SECURITY. Enciclopedia de Virus Stuxnet. [online]. Junio, 2010. [citado 13 mar., 2018]. Disponible en Internet: < https://www.pandasecurity.com/peru/homeusers/security-info/222123/information/Stuxnet.A >

las centrales nucleares[26]. Obtener los reportes estadísticos ayudan a realizar comparaciones con las tendencias mundiales sobre ciberamenazas; lo cual al final puede ayudar a determinar si en realidad la empresa es blanco de un criminal en específico (lo cual podría inferir que se trata de alguien que conoce la compañía) o si se trata de alguien cuyos objetivos están relacionados hacia el sector de la organización, la política, el país o simplemente lo hace por diversión.

- Detectar nuevas amenazas (Zero-Days): Una de las ventajas más evidentes de los *sistemas de sueño* es la captura de amenazas de tipo *ZeroDays.* En algunas ocasiones, es posible que las *HoneyPots* debido a su grado de exposición, cuenten también con una característica vulnerable desconocida a la fecha; lo cual se hace bastante tentador ante un atacante quien puede probar una explotación de *Día Cero*[24]*.* Aunque no es usual este tipo de escenarios, es posible su ocurrencia siendo más probable que el elemento explotado sobre la red sea la *HoneyPot* que inicialmente fue expuesta con tal fin.

- Detectar muestras de Malware: Este objetivo es bastante similar al anterior. Mediante estos elementos, además de la posibilidad de detectar *Zero Days,* es posible capturar muestras de variaciones de *malware* ya existentes. Este tipo de información suele ser útil al momento de personalizar los mecanismos de seguridad que se tienen dentro de la organización[24].

- Distraer al enemigo: No sólo se pretende crear un escenario donde se pueda estudiar las amenazas actuales. Los *sistemas de sueño,* además de esta característica; pueden distraer y desviar los ataques de un enemigo en específico hacia ambientes controlados[24]. Mientras el enemigo cree que estaría explotando vulnerabilidades reales de la organización, en realidad del otro lado estaría siendo cuidadosamente estudiado bajo un ambiente controlado que proteja a los demás elementos en un entorno organizacional.

- Monitorear todo tipo de acciones: Este objetivo es quizás el más importante de una *HoneyPot.* Se debe tomar nota de todas las acciones realizadas para llevar a cabo la explotación o intento de explotación de un sistema; con esta información se pueden generar los reportes y otorgar datos de valor a los analistas de la *Honey.* Datos como: tramas de paquetes, software utilizado, ataque desarrollado, entre otros; pueden ser útiles al momento de analizar qué tipos de amenaza se han materializado[24].

---

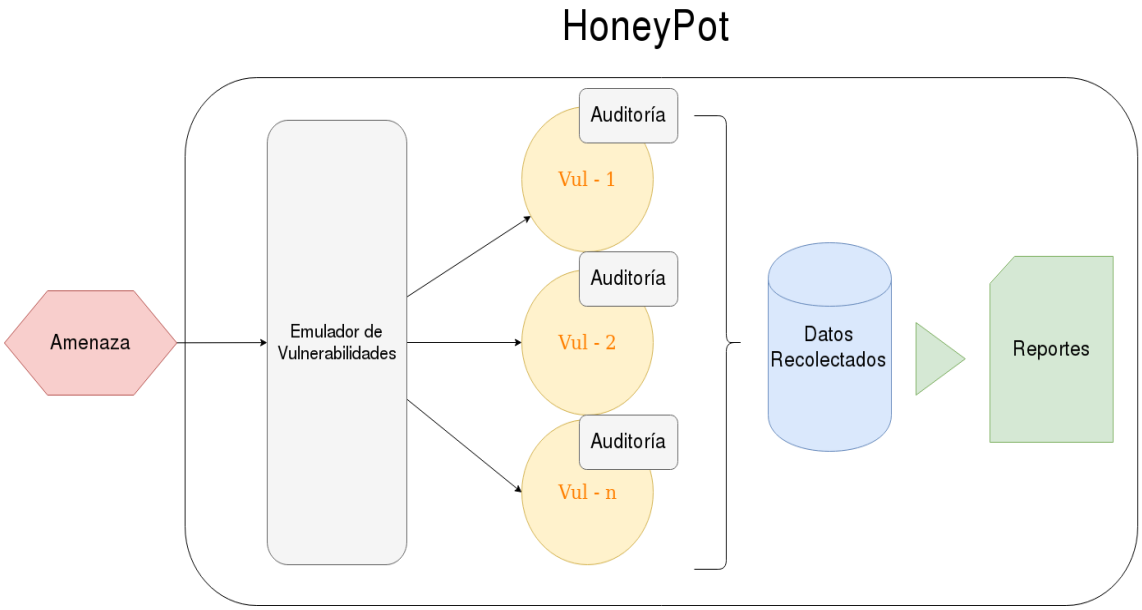
[26] FRUHLINGER Josh. What is Stuxnet, who created it and how does it work [online]. Agosto, 2017. [citado 13 mar., 2018]. Disponible en Internet: < https://www.csoonline.com/article/3218104/malware/what-is-stuxnet-who-created-it-and-how-does-it-work.html >

Estos objetivos en general, ayudarán y apoyarán un proceso de toma de decisiones dentro de la empresa; donde además de determinar qué acciones preventivas se pueden llevar a cabo para el resto de la red, se puede incluso estudiar la forma de contrataque; en caso que el sistema comprometido se trate de una entidad gubernamental o con información bastante sensible.

5.1.1.2 Estructura de un HoneyPot

Cada *HoneyPot* corresponde un conjunto de elementos dedicados a labores y tareas específicas. Cada uno de estos elementos se puede observar en la Figura 1.

Figura 1 Estructura General de una HoneyPot



Fuente: El Autor.

Los elementos expuestos en la Figura 1 son los siguientes.

- Emulador de Vulnerabilidades: Se encarga de simular el comportamiento real de uno o varios Servicios en específico, valiéndose de una configuración por defecto compuesta por un set de vulnerabilidades intencionalmente expuestas ante una amenaza. Es aquí donde se implementan algunas

tácticas de *Deception* o engaño, para intentar atraer la atención de la amenaza[27].

- Vulnerabilidades: Son aquellas características de un Servicio en específico expuestas ante una amenaza. En realidad, las *HoneyPots* no exponen la vulnerabilidad completamente, sino alguna simulación de la misma. Por ejemplo: El Servicio *SSH* se sabe que puede ser configurado de forma que reciba credenciales de usuario, lo cual puede ser víctima de un ataque de fuerza bruta o dirigido por diccionario; sabiendo de antemano esta vulnerabilidad se puede configurar la *HoneyPot* para que haga creer a la amenaza que es un servicio *SSH* cuya configuración acepta el par de credenciales de usuario[27]. Sin embargo, en caso que el atacante proporcione las debidas credenciales de usuario, dependiendo de la implementación de la *HoneyPot* no se daría nunca acceso a la amenaza ya que consiste en un Servicio Falso. Las vulnerabilidades son concentradas y emuladas por el *Emulador de Vulnerabilidades,* ya que pueden existir un conjunto de ellas para un Servicio en Específico.

- Auditorías: La amenaza no surtirá efecto en realidad, aunque si todas las acciones realizadas serán registradas. En el ejemplo anterior, todas las credenciales de usuario probadas por *SSH* serán guardadas[27].

- Datos Recolectados: Las auditorías alimentan una Base de Datos o conjunto de *logs* en general, donde se puede consultar la información relacionada a las actividades maliciosas.

- Reportes: Basados en la información y datos recolectados es posible que la *HoneyPot* tenga la capacidad de generar reportes estadísticos que permitan clasificar cada uno de los tipos de ataque[27]. Esto no es una característica obligatoria en la estructura de un *sistema de señuelo* y puede ser desarrollado por un *software* externo, sin embargo, algunas soluciones en la actualidad como el Sistema Operativo *T-Pot* puede proveer esta característica.

Mientras se esté registrando las actividades maliciosas, el atacante recibirá una respuesta dependiendo del tipo de ataque realizado; de esta manera se completa la simulación del Servicio. Es posible que las respuestas otorgadas por la *HoneyPot* busquen desconcertar al atacante, por ejemplo; en el Servicio *HTTP* es bastante conocido que el método *PUT* permite subir archivos al servidor, lo cual puede ser aprovechado por un atacante que desee colocar un *Web Shell* en el Servidor; una *HoneyPot* de *HTTP* podría exponer este método, al momento que la amenaza lleve

[27] JURADO PALLARÉS, Diego. Op. cit., 20 p.

a cabo la ejecución de este método para el protocolo; podría un sistema de señuelo retornar una respuesta satisfactoria aunque el atacante no podría definir si el Servidor en realidad ha obtenido el archivo para posteriormente guardarlo o simplemente el método en realidad ha fallado. Esta simulación de respuestas es bastante útil, en especial cuando el tipo de amenaza es una *botnet*[28].

5.1.1.3 Ventajas y Desventajas de los HoneyPots

Los *sistemas de señuelo* son otro tipo de *software* con ciertos objetivos mencionados con anterioridad. Aun así, no dejan de ser programas informáticos; por ende, se debe tener presentes cuáles son sus ventajas y desventajas para lograr una correcta aplicación de acuerdo a las necesidades en una red empresarial. La Tabla 1 da un resumen las ventajas y desventajas de un sistema de señuelos.

Tabla 1 Ventajas y Desventajas de los HoneyPots o Sistemas de Señuelo

Ventajas	Desventajas
Baja cantidad de falsos positivos	Pueden ser usadas para atacar otras vulnerabilidades.
Capturar información de valor.	Sólo recogen información sobre su emulador de vulnerabilidades, no sobre la totalidad de la red
No requieren de ataques conocidos para ejecutar sus funciones.	Podrían ser fácilmente dectadas por un atacante o amenaza sofisticada.
Funcionan en entornos cifrados.	Los registros podrían estar contaminados con datos falsos.

Fuente: El Autor.

A continuación, se expone cada uno de los aspectos a mayor profundidad.

[28] ZOU Cliff, CUNNINGHAM Ryan. Honey-Aware Advanced Botnet Construction and Maintenance. Orlando: School of Electrical Engineering and Computer Sciente, University of Central Florida. 2009. 22 p.

- Ventajas

1. *Baja cantidad de falsos positivos.* Si en realidad el sistema ha sido correctamente configurado, es muy probable que todo el tráfico capturado sea legítimo[29]. Para ello se debe procurar que sólo los encargados de seguridad conozcan de la existencia de la *Honey,* y que esta información no sea divulgada.

2. *Capturan información de valor siempre y cuando no se contaminen los logs.* Bajo un entorno ideal, las *HoneyPots* se centran únicamente en el tráfico sobre el cual se identifiquen el desarrollo de amenazas. Esto es una ventaja en contraste a otro tipo de componentes como *sniffers,* donde se registra toda la actividad de la red sin distinguir si existe un ataque llevándose a cabo. Sin embargo, la dualidad de *sniffer* y *HoneyPot* suele ser bastante útil si se quiere estudiar de manera detallada las amenazas recogidas por las vulnerabilidades expuestas[29].

3. *En su mayoría, no requieren ataques conocidos.* A diferencia de otro tipo de elementos de red como los *Sistemas de Detección y Prevención de Intrusos (IDS-IPS),* la captura de una *HoneyPot* no se limita únicamente a las firmas reconocidas y acciones definidas en las reglas de estos programas informáticos. El *sistema de señuelo* puede registrar actividades maliciosas desconocidas, lo cual le coloca un paso más delante de los *IDS-IPS*[29]*.*

4. *Pueden trabajar en entornos cifrados.* La emulación de servicios puede no limitarse únicamente a Servicios cuyos protocolos sean texto plano, se puede agregar soporte para el manejo de cifrado si así es requerido; dando la impresión de un punto de ataque más codiciado por la amenaza[29]. Del mismo modo, todos los resultados pueden ser guardados en discos y archivos cifrados, de manera que en caso de una intrusión completa al sistema sería difícil identificar cuáles son las evidencias tomadas sobre las amenazas. Además, recolectar la información en archivos cifrados permite que únicamente los involucrados en su despliegue y vigilancia puedan disponer de los datos recogidos.

[29] PETER, Eric y SCHILLER, Tood. A practical Guide to Honeypots [online]. Washington D.C. Abril, 2008. [citado 13 mar., 2018]. Apache Foundation. Disponible en Internet: < https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/index.html >

- Desventajas

1. *Puede ser usada por el atacante para atacar otras vulnerabilidades.* Esto sucede si se carece de una emulación adecuada de las vulnerabilidades para la *Honey.* En ocasiones, existen malas prácticas donde se configurar servicios por defecto y se permite el acceso al atacante; sin embargo, al disponerse de esta manera de despliegue la amenaza aún puede sacar provecho del Servicio, precisamente porque se trata de una implementación real que carece de *Deception* o técnicas de engaño[29]. Al tenerse un sistema tan vulnerable y que pueda ser explotable, además de lograr escalar su explotación; se estaría otorgando la posibilidad al criminal de enmascararse bajo la propia *HoneyPot,* es por eso que se recomienda el uso de soluciones especializadas y probadas con anterioridad.

2. *Únicamente la HoneyPot puede detectar los ataques de su emulador de vulnerabilidades, no puede detectar ataques a través de la red*. Esto quiere decir que un sistema de señuelo se tiene dispuesto en un único host, donde se dispone de la emulación de vulnerabilidades en específico. Si el atacante decide, por suspicacia o casualidad, dirigir su amenaza hacia otros elementos de la red; es posible que no se tengan resultados[29]. Para mitigar este riesgo, se puede introducir el concepto de HoneyNet.

3. *Pueden ser detectadas por el atacante.* En un escenario donde se esté enfrentando a un tipo de atacante experimentado, es posible que éste puede identificar que se ha expuesto una vulnerabilidad intencionalmente para ser atacada. Esto depende de la intuición de quien lleva a cabo la amenaza y el nivel de exposición de la *HoneyPot.* Un *sistema de señuelo* que tenga muchos tipos de vulnerabilidades diversos, puede ser bastante evidente para alguien experimentado en Seguridad Informática. El riesgo de detección aumenta cuando el tipo de amenaza es llevado a cabo por un humano y puede ser relativamente bajo cuando se trata de un proceso automatizado como el caso de escaneos de vulnerabilidades, gusanos de internet o *botnets.*

4. *Los registros y datos recolectados por la HoneyPot pueden ser contaminados.* Alguien que tenga conocimiento de la instalación y despliegue de la *Honey,* puede enviar tráfico o realizar acciones legítimas sobre su emulador de vulnerabilidades. Irónicamente, esto es una situación no deseada; ya que pueden alterar los datos de la *HoneyPot* inundándola de tráfico basura.

Estas ventajas y desventajas pueden variar con base al ambiente de configuración del *HoneyPot.*

5.1.1.4 Tipos de HoneyPots

Una *HoneyPot* puede ser clasificada en dos dimensiones básicamente, *Según su Propósito* y *Según su Interacción.*

a. *Según su Propósito*

Básicamente depende del uso y finalidad que se quiera dar al *sistema de señuelo.* Existen dos tipos en esta dimensión los cuales son: *Propósito Investigativo* y *Propósito Productivo.*

- Propósito Investigativo

Destinada para el estudio de amenazas actuales, recolección e identificación de tendencias. Este tipo de *Honeys* pueden otorgar a las organizaciones el valor de la retroalimentación, permitiendo identificar cuáles son las vulnerabilidades más explotadas dentro de su entorno empresarial. Pueden suponer una gran desventaja, ya que en ocasiones pueden requerir un sistema completamente vulnerable dentro de la red empresarial; por lo cual podría ser utilizada para atacar otros elementos de la misma red. Muchas de las organizaciones no implementan este tipo de mecanismos por falta de conocimiento, temor o porque simplemente consideran que es un gasto innecesario[30].

Realmente una *HoneyPot de Propósito Investigativo,* puede ser una inversión para las organizaciones; si se sabe implementar de forma que las desventajas sean reducidas al mínimo. Algunas de las empresas y firmas más reconocidas de Seguridad Informática utilizan este tipo de mecanismos como sondas de recolección a nivel internacional, encontrando de esta manera cualquier tipo de amenaza.

- Propósito Productivo

Dirigidas a la distracción del atacante y mitigación de las amenazas. Dentro de sus fines se pueden encontrar algunas similitudes al tipo anterior, aunque no entran a

[30] VUSAL Aliyev. Using honeypots to study skill level of attackers base don the exploited vulnerabilities in the network. Göteborg: Department of Computer Science and Engineering Division of Computer Security, Chalmers University of Technology. 2010. 69 p.

mucho detalle sobre la identificación de tendencias; únicamente interesa generar una configuración espejo a algunos equipos de la organización, tomando como base la estructura de la red empresarial y los Servicios ofrecidos[30]. De esta manera se pueden obtener las amenazas en específico, que puedan ser destinadas a un tipo de despliegue como el otorgado en los ambientes productivos de la red. Con los resultados arrojados de este tipo de *Honey* se pretenden realizar procesos de *hardening* detallados y específicos, con el objetivo de prevenir a los elementos de la red de algún tipo de amenaza reciente. De igual manera, puede distraer al atacante siendo punto de foco y concentración de los atacantes.

b. *Según su Interacción*

Depende de la forma en que el atacante interactúa con el emulador de vulnerabilidades de la *HoneyPots,* depende del grado de exposición e interacción de las vulnerabilidades con la amenaza. Existen tres tipos básicamente: *Bajo, Medio* y *Alto.* Según sea el grado de interacción mayor podrían ser los datos recolectados por la *Honey,* es decir; si el nivel es *Bajo* posiblemente se capture una densidad de datos menor a si es de tipo *Medio* o *Alto.*

- Baja Interacción

Utilizado para la identificación de nuevas amenazas ya que carece prácticamente de cualquier tipo de riesgo, ya que en realidad el emulador de vulnerabilidades no da bastantes opciones al atacante. Por lo general se trata de Máquinas Virtuales donde se desarrollan emulaciones de amenazas, es decir, se dispone de un conjunto de Sistemas Operativos en los cuales se ejecuta intencionalmente la amenaza; bajo el entorno controlado se estudia qué tipo de afectaciones puede ocurrir con el *malware* que se encuentra analizado[30]. Aunque permite algunos topos de acciones, podría no proporcionar datos muy específicos a la amenaza y al atacante por lo cual es posible que algunas de las características estudiadas se encuentren bastante limitadas. Aunque sus resultados no sean confiables en un cien por ciento, ya que puede limitar a la amenaza capturada; es fácil de mantener.

- Media Interacción

Recogen más información de las amenazas en contraste a las *Honeys* de *Baja Interacción,* sin embargo, no es suficientemente detallada como para considerarse

de *Alta Interacción.* En lo general se encuentran ejecutadas bajo entornos reales, es decir no se dispone de *software* de virtualización; lo cual puede acarrear un nivel de riesgo más elevado. Su funcionamiento se basa en la emulación del comportamiento real de los Servicios estudiados, por lo tanto, quien desee implementarla debe contar con conocimientos básicos sobre la gestión de redes; lo cual es una de sus mayores desventajas[30]. Su instalación y despliegue requiere tiempo de configuración y la presentación de un fallo o explotación de una vulnerabilidad real podría colocar en riesgo el resto de los elementos en red.

- Alta Interacción

Básicamente se trata de una Máquina Real, en un entorno de red real con los Sistemas Operativos, *software* y Servicios comunes que se disponen para un usuario real. Este tipo de interacciones supone un alto riesgo para la *HoneyPot* y la red donde se encuentra. En teoría no debe usar emulación de Servicios o virtualización, es decir que se dispone de un sistema que en realidad ha sido mal configurado; por lo tanto, el componente de emulador de vulnerabilidades es prácticamente inexistente para este tipo[30]. Bajo una implementación y despliegue adecuados, debería utilizar *software* de terceros o mecanismos de auditoría del Sistema Operativo para obtener la mayor cantidad posible de información. En este tipo de *HoneyPots* se hacen necesarias las aplicaciones de elementos de seguridad de red como *Firewalls, Sistemas de Detección y Prevención de Intrusos (IDS-IPS), Auditorías, Zonas Desmilitarizadas DMZ* y/o *Seguridad de la Información y Administración de Eventos (SIEM).*

5.1.1.5 Ubicación en Red de una HoneyPot

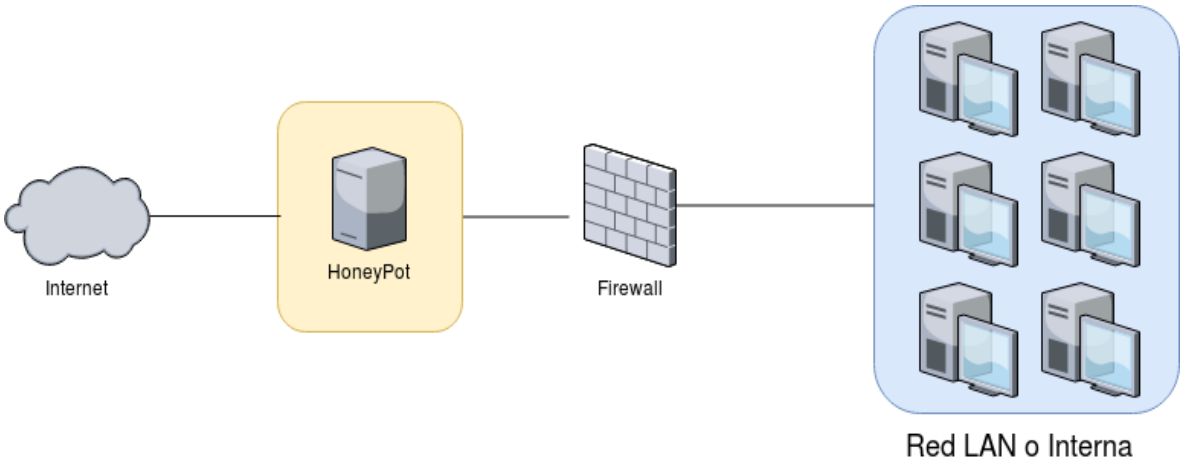
Tener en cuenta la ubicación de una *HoneyPot* según los requerimientos de la investigación es una parte fundamental para su implementación. Colocar un sistema de señuelo en un punto de red poco concurrido, podrá eliminar sus capacidades de atracción para las diferentes amenazas en la red, por otra parte; si se coloca en un punto demasiado obvio es posible que las amenazas puedan identificarla fácilmente y la eviten deliberadamente. Básicamente existen tres puntos donde se recomienda ubicar una *HoneyPot.*

a. *Delante del Firewall*

Los sistemas de señuelos ubicados delante del *Firewall* ofrecen la capacidad de recoger información sobre las amenazas sin exponer a las redes internas. El

Firewall haría frente a cualquier tipo de amenaza que intentase redirigirse hacia las redes internas[31]. Además de esto, evita las configuraciones adicionales sobre otros mecanismos de seguridad en las redes internas; teniendo en cuenta que después del *Firewall* no existiría ningún tipo de tráfico que cualquier otro sistema de seguridad como *IPS* o *IDS* pudiese considerar como malicioso. En pocas palabras, la *HoneyPot* sería el único elemento de red que estaría completamente expuesto. Esta manera de ubicar la *HoneyPot* la puede hacer muy evidente, y podría no detectar amenazas que ya se encuentran dentro de la red interna, igualmente es posible que se presente un ancho de banda elevado si las amenazas son de tipo *Denegación del Servicio*. La Figura 2 expone a grandes rasgos la topología en red en donde se identifica la ubicación de la *HoneyPot.*

Figura 2 Ubicación de HoneyPot Delante del Firewall



Fuente: El Autor.

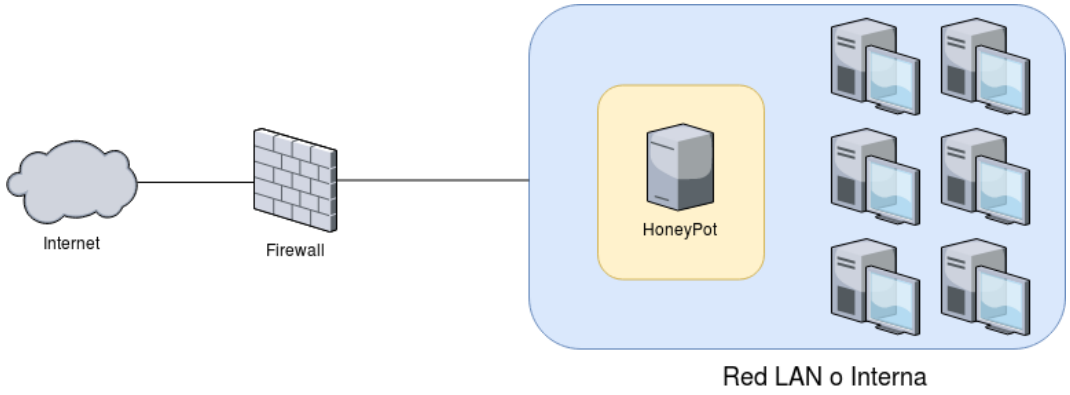
b. *Detrás del Firewall*

Ubicarla detrás del *Firewall* supone que la *HoneyPot* estará ligada a cualquier tipo de restricción impuesta por este sistema de seguridad, esto mejora un poco el enfoque en cuanto a manejo de ancho de banda y no la hace tan vulnerable frente a amenazas de tipo *Denegación del Servicio*[31]*.* También otorga la ventaja de proporcionar resultados sobre amenazas internas, que se encuentran más allá de las restricciones del *Firewall.* Pese a estas ventajas, es poco recomendado ubicar un sistema de señuelo de esta manera; ya que requiere configuraciones adicionales para los demás sistemas o mecanismos de seguridad de la red interna, haciendo

[31] CYBSEC. Seguridad Informática HoneyPots [online]. CYBSEC jul, 2008. [citado 13 mar., 2018]. 3. Ubicación. Disponible en Internet: < http://www.cybsec.com/upload/ESPE_Honeypots.pdf >

más engorrosa la mantenibilidad y protección de la red interna. Es necesario tener presente que, al configurar los mecanismos de seguridad de la red interna, si se agregan normas muy generales a listas blancas para evitar que el tráfico de la *HoneyPot* lance falsos positivos; se puede estar dejando un hueco de seguridad. La Figura 3 expone a grandes rasgos la topología en red en donde se identifica la ubicación de la *HoneyPot.*

Figura 3 Ubicación de HoneyPot Detrás del Firewall



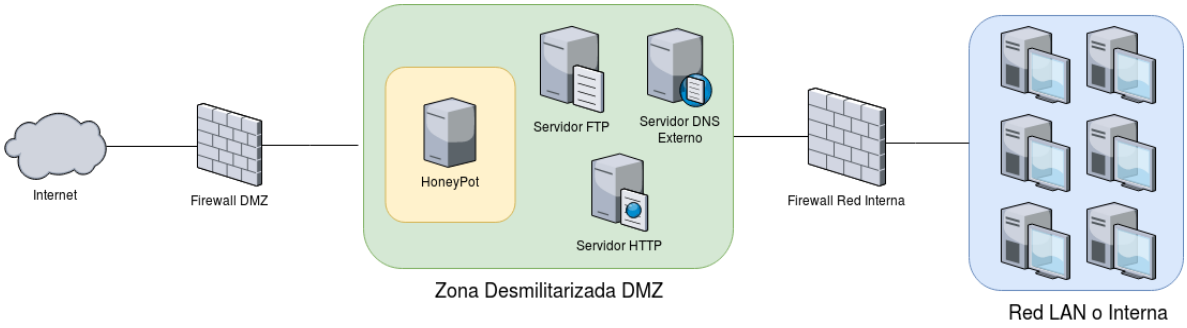
Fuente: El Autor.

c. *En la Zona Desmilitarizada DMZ*

Quizás sea el tipo de ubicación más recomendado y seguro, al menos para el desarrollo del proyecto, ya que implica una separación de la red interna, así como de las amenazas que se están más allá de la primera barrera de protección *Firewall*[32]*.* Esta configuración implica que existe un primer *Firewall* encargado de filtrar cierto tipo de tráfico específico a la red, luego de este *Firewall* vienen los elementos de la *DMZ* entre ellos incluidos la *HoneyPot,* posteriormente existe un segundo *Firewall* que se dirige hacia la red interna. Esta configuración impide que ataques de tipo *Denegación de Servicio* dirigidos hacia la *HoneyPot* puedan consumir un ancho de banda considerable, sin embargo, las amenazas internas de la red podrían pasar desapercibidas. La Figura 4 expone a grandes rasgos la topología en red en donde se identifica la ubicación de la *HoneyPot.*

[32] CYBSEC. Seguridad Informática HoneyPots. Op. cit., p. 26.

Figura 4 Ubicación de HoneyPot en la Zona Desmilitarizada



Fuente: El Autor.

5.1.1.6 Algunas soluciones conocidas de HoneyPots

Aplicar un *HoneyPot* puede ser algo complejo, en especial si se quiere reducir el riesgo de afectación de la red o las máquinas que se dispondrán para esta tarea. Como se ha observado antes para lograr este tipo de implementación se debería optar por *HoneyPots de Interacción Baja* o *Media.* Esto supone un conocimiento elevado sobre los protocolos de red y sus respectivos servicios. Invertir en estos conocimientos puede requerir un esfuerzo muy alto para una organización que al final simplemente busca la forma de implementar el concepto. Afortunadamente en la actualidad, con la popularización de las herramientas de *pentesting,* se han popularizado algunas herramientas especializadas que permiten emular el comportamiento de algunos de los servicios para los protocolos más populares de una red. Algunas de ellas son.

a. *T-Pot O.S – HoneyPot Multiplataforma*

T-Pot es un Sistema Operativo basado *Ubuntu* que básicamente permite la implementación de varias *HoneyPots* multiplataforma o Servicios múltiples. Es bastante sencillo de implementar y ofrece una interfaz gráfica bastante amigable donde presenta una recolección de amenazas identificadas; ofrece algunas de las mejores tecnologías en el campo de *HoneyPot* siendo fácil de instalar y desplegar. Básicamente es la solución para las empresas que desean implementar este concepto de forma experimental y rápida[33].

La recolección de datos se hace mediante *Sistemas de Detección de Intrusos basados en Hosts (HIDS)* y *Sistemas de Detección de Intrusos basados en Red*

[33] DEUTSCHE TELEKOM AG. T-Pot: A Multi-HoneyPot Platform [online]. Github mar, 2015. [citado 20 mar., 2018]. Disponible en Internet: < http://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html >

(NIDS), en combinación con algunos servicios *Honey* y herramientas para la captura de ataques. Todos sus servicios se encuentran basados en *Docker,* lo cual ofrece una capa adicional de seguridad al estar desplegados bajo un contenedor. Aun así, la dependencia total a este tipo de soluciones *dockerizadas* le da una fuerte desventaja, si *Docker* quiebra todos los servicios lo harán con él, por lo tanto, se puede perder información. En su documentación oficial se refieren a que el concepto de rapidez no tiene en cuenta los mecanismos para guardado de los datos de manera confiable[33]. Este riesgo puede ser mitigado configurando de forma personalizada cada una de las imágenes de *Docker,* y los servicios desplegados con ellas.

Puede ser adquirida de manera gratuita en el siguiente enlace:

https://github.com/dtag-dev-sec/tpotce [33]

b. Glastopf – HoneyPot para Aplicación Web

Aplicación vulnerable creada en *Python* donde se exponen algunas vulnerabilidades comunes como *Remote File Inclusion, Local File Inclusion,* escape de *sandboxing.* Provee además simulación para captura de inyecciones en peticiones *POST* del protocolo *HTTP.* También puede otorgar una emulación de *SQL Injection* mediante la exposición de formularios. Esta herramienta fue fundada por *Lukas Rist* y es bastante conocida para el testeo de vulnerabilidades que afectan al protocolo *HTTP* y algunas implementaciones o tecnologías de las aplicaciones web. Comprende una gran cantidad de vulnerabilidades y se encuentra abalada por el *HoneyNet Project,* lo cual le otorga un título de alta confiablidad[34]. Es un tipo de *HoneyPot* de baja interacción, esto significa que su implementación no supone mucho riesgo para una red organizacional.

Puede ser adquirida de manera gratuita en el siguiente enlace.

https://github.com/mushorg/glastopf [34]

c. Kippo – SSH HoneyPot

HoneyPot de nivel de *Interacción Media* diseñada para capturar los ataques de fuerza bruta y algunas de las más importantes vulnerabilidades del protocolo *SSH.* Esta herramienta provee algunas características como la falsificación de Sistemas de archivos para hacer creer al atacante que está eliminando o accediendo a

[34] GLASLOS. Glastopf Web Application HoneyPot [online]. Github feb, 2014. Github. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/glastopf >

información real del sistema, requiere la utilización de *Python 2, PyCrypto, Zope Interface* y *Twisted >=8 y <=15* [35]*.* Tiene algunos mecanismos de auditoría o *logs* compatible con *software* externo. Fue fundado por *Michel Oosterhof's.*

Puede ser adquirida de manera gratuita en el siguiente enlace.

https://github.com/desaster/kippo [35]

d. Maloney – SMTP HoneyPot

Maloney es una simple implementación para una *HoneyPot* en el Servicio de Correo Electrónico *Simple Mail Transfer Protocol (SMTP).* Simplemente captura el tráfico generado en el Servicio, guardando todos los correos electrónicos que se quieran enviar a través de este Servicio o que puedan ser enviados a éste. Dispone de tres tipos de ejecución: *open_relay* que únicamente guarda todos los correos entrantes y salientes del Servicio, *postfix_creds* que guarda todo tipo de credenciales con los cuales intentaron ingresar al sistema y *schizo_open_relay* que consiste en la activación de ambos[36].

Puede ser adquirida de forma gratuita en el siguiente enlace.

https://github.com/awhitehatter/mailoney [36]

e. SpamHAT – HoneyPot para captura de SPAM

Creado por *Miguel Raúl Bautista Soria* es una herramienta que permite el registro de todo tipo de *SPAM* por lo tanto está intrínsecamente relacionada al protocolo *SMTP*[37]*.* En conjunto con la herramienta anterior pueden otorgar un concepto de captura de amenazas más general sobre el protocolo de Correos Electrónicos. Requiere el uso de *Perl 5, CPAN Modules* y *MySQL.*

Puede ser adquirida de forma gratuita en el siguiente enlace.

https://github.com/miguelraulb/spamhat [37]

---

[35] DESASTER. Kippo SSH HoneyPot [online]. Github mar, 2010. Github. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/desaster/kippo >

[36] AWHITEHATTER. Mailoney SMTP HoneyPot [online]. Github jul, 2016. [citado 20 mar.,2018]. Disponible en Internet: < https://github.com/awhitehatter/mailoney >

[37] MKRUL. Spam HoneyPot Tool [online]. Github jun, 2015. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/miguelraulb/spamhat >

f. HoneyPotBuster

Creada por *Eyal Neemany* se trata de un *script* escrito en el lenguaje *Powershell* el cual puede simular algunas características importantes para un Servicio de Directorio Activo, características que pueden incluir algunas peticiones y respuestas sobre el protocolo *LDAP.* Permite la simulación de las características: Servicio de *kerberoasting* de cuentas de usuario, Falsificación de credenciales en memoria, Falsificación de cuentas de usuario para una computadora específica, Detección de manipulación de las entradas *DNS,* Falsificación de Cuentas de Administrador, entre otras características[38].

Su objetivo principal es identificar técnicas de evasión que normalmente se usan en un escenario de ataque, cuando el objetivo se trata de un Servicio de Directorio Activo[38]. El hecho de estar escrito en *powershell* lo hace compatible con la mayoría de *hosts* con Sistema Operativo Windows, además se puede usar e instalar fácilmente.

El *script* puede ser adquirido de manera gratuita en el siguiente enlace.

https://github.com/JavelinNetworks/HoneypotBuster [38]

g. Repositorios de HoneyPots Adicionales

Otro tipo de *HoneyPots* se pueden encontrar en los repositorios de *awesome honeypots.* Estos dos repositorios albergan referencias hacia diferentes soluciones que abarcan desde otros protocolos de red, servicios e incluso *software* y aplicaciones nativas que están explícitamente dedicados a la recolección de información, simulación de vulnerabilidades y otras características intrínsecas al concepto de *HoneyPot [39] [40].* Es posible que durante el desarrollo del proyecto se seleccione algunos de los *HoneyPots* que se exponen en estos repositorios.

---

[38] NEEMANY, Eyal. Honeypot Buster: A Unique Red-Team Tool [online]. Javeli Networks jul, 2017. [citado 20 mar., 2018]. Disponible en Internet: < https://jblog.javelin-networks.com/blog/the-honeypot-buster/ >

[39] PARALAX. Awesome HoneyPots [online]. Github mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/sindresorhus/awesome >

[40] JAYANDCATCHFIRE y SINDRESOURHUS. Awesome list [online]. Github mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/sindresorhus/awesome >

La dirección de los dos repositorios es la siguiente:

https://github.com/paralax/awesome-honeypots[39]

https://github.com/sindresorhus/awesome[40] (Además de contener *HoneyPots* también referencia a otro tipo de herramientas interesantes en la seguridad y el desarrollo de *software*).

5.1.2 Servicios y Protocolos de Red a Evaluar

Conocer el funcionamiento de los servicios para los diferentes protocolos de red a estudiar es clave para dilucidar las configuraciones de los *HoneyPots.* Así mismo, es importante conocer cuál es el flujo básico de una comunicación mediante los protocolos tratados en el proyecto y qué tipo de flujos alternos podrían considerarse como amenazas. De esta manera, al momento de analizar la información recolectada es posible destacar qué tipo de variables influyeron en el despliegue de los *HoneyPots.*

5.1.2.1 Servicio Web (Protocolo: Hyper-Text Transfer Protocol HTTP)

Servicio usado para el acceso a páginas *web,* en su mayoría de aplicaciones *web*, aunque puede ser implementado para la comunicación de datos de aplicaciones de escritorio y móviles mediante el uso de *web services* y *web sockets.* Básicamente atiende peticiones de tipo HTTP o HTTPS, este segundo solamente si se soporta la comunicación por medio de TLS/SSL. Otorga al cliente una respuesta que puede ser interpretada por un *software* que se encuentra completamente desligado al servicio[41]. Ejemplo: un cliente normalmente cuenta con un *software* conocido como navegador *web* o *Web browser,* el cual lanza peticiones hacia un Servidor *Web,* el servidor contesta con una respuesta que puede ser interpretada de manera gráfica por el cliente o simplemente utilizada para complementar algún tipo de información. Tanto los conceptos como *web services* y *web sockets* también funcionan bajo el protocolo HTTP*.* El *software* responsable de la ejecución de Servicio *Web* se conoce como *Servidor Web*[41]*.*

Entre el software más popular para los *Servidores Web* se pueden encontrar el Internet Information Services IIS de Windows, Apache Server de Apache Foundation, JBoos o Wildfly de Red Hat, entre otros[41].

---

[41] ROSENBLUM, Mendel. Web Servers [online]. Stanford mar, 2016. [citado 20 mar., 2018]. Disponible en Internet: < https://web.stanford.edu/class/cs142/cgi-bin/slides/WebServers.pdf >

Las diferentes implementaciones de los Servicios *Web* pueden acarrear el uso de tecnologías, compiladores, lenguajes de programación y *frameworks* distintos; sin embargo, el funcionamiento del Servicio sigue un mismo patrón debido a la definición del protocolo. Según las tecnologías y capacidades del Servidor *Web,* se puede encontrar un conjunto variado de vulnerabilidades y posibles huecos de seguridad.

a. Funcionamiento Básico del Protocolo

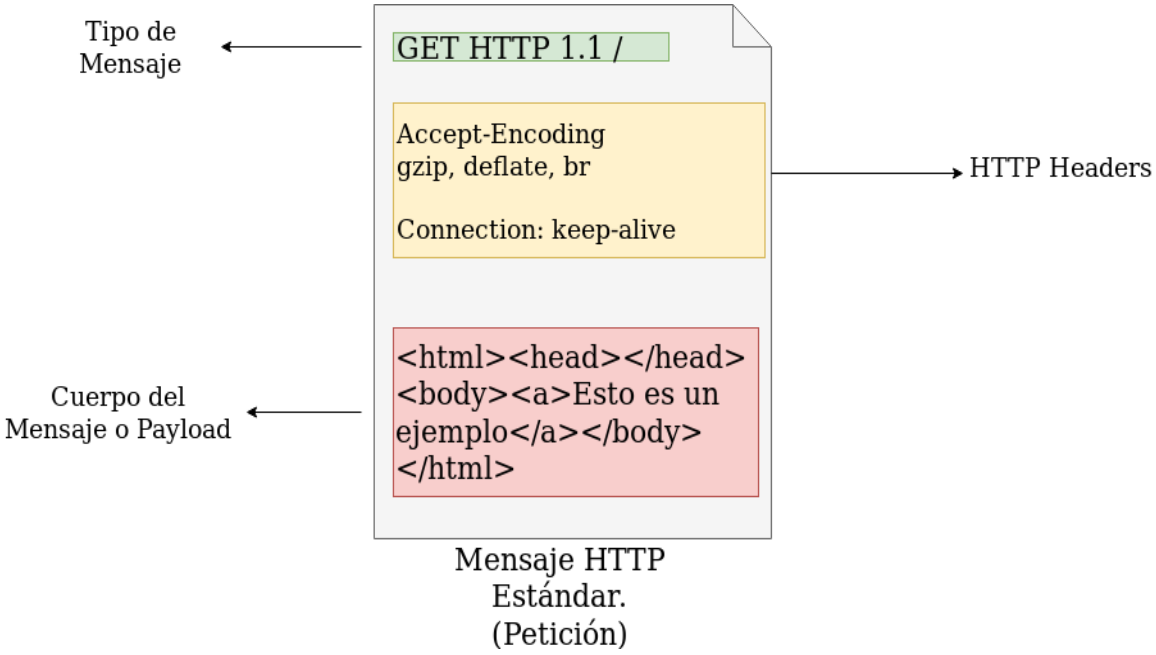
El protocolo *HTTP* se ubica en la capa de *Aplicación* según el *Modelo OSI* y *Modelo TCP/IP,* el cual en su mayoría de implementaciones viaja a través de paquetes *TCP,* por lo cual se estaría hablando de un protocolo de alto nivel que comunica a dos tipos de *software* en el cliente y servidor. Entre sus funcionalidades sus capacidades se puede encontrar la transferencia de diferentes archivos multimedia por medio de una gran variedad de tipos conocidos como *MIME Types.* Este protocolo es genérico y carece de estado, es decir que las diferentes peticiones realizadas al servidor podrían no tener una secuencia fija. Se considera parte del estándar *Wolrd-Wide Web* desde 1990 en su versión *HTTP 1.1.* Actualmente, en ambientes productivos se puede encontrar las versiones del protocolo *HTTP 1.1* y *HTTP 2.0* en su mayoría, siendo el primero la más común hasta el momento por otorgar compatibilidad con la gran mayoría de navegadores y *software* del lado del cliente[42]. La principal diferencia entre *HTTP 1.1* y *HTTP 2.0* es el rendimiento, la versión más reciente está pensada en otorgar mejoras a las conexiones llevadas a cabo por el servidor. Aun así, la sintaxis y semántica de ambos es bastante similar.

El protocolo *HTTP,* en cualquiera de sus versiones, se fomenta a raíz de los mensajes con estructura homónima al mismo. Estos, son una secuencia de octetos los cuales tienen una estructura estándar. Todo mensaje *HTTP* se puede dividir en dos tipos: Petición y Respuesta. Como su nombre lo indica, una *Petición (Request)* es un mensaje dirigido hacia el Servidor; mientras una *Respuesta (Response)* es un mensaje que proviene del Servidor[43]. La estructura general de ambos tipos se puede observar en la Figura 5.

[42] GOOGLE INC, MOZILLA, BITGO et al. HyperText Transfer Protocol Version 2 [online]. Internet Engineering Task Force IETF may, 2015. [citado 20 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc7540 >

[43] XEROS, W3C/MIT, MICROSOFT y otros. HyperText Transfer Protocol HTTP 1.1 [online]. Internet Engineering Task Force IETF jun, 1999. [citado 20 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2616 >

Figura 5 Estructura general de un Mensaje HTTP



Fuente: El Autor.

La Figura 5 toma como ejemplo un mensaje de tipo solicitud o *Request,* aunque la estructura es bastante similar para las respuestas. Básicamente un mensaje se divide en tres partes[43]:

- Tipo de Mensaje: Indica si es una *Petición* o *Respuesta,* además de especificar otras características importantes sobre el mensaje; como lo es la versión, entre otros aspectos que serán explicados posteriormente.

- HTTP Headers: Cabeceras *HTTP* que suelen ser bastante importantes para definir comportamientos tanto del Servicio *Web,* en caso que sea un mensaje de *Petición,* como del navegador *web,* en caso que sea un mensaje de *Respuesta.* También pueden incluir metadatos importantes sobre lo que se encuentra en el Cuerpo del mensaje o *Payload.*

- Cuerpo del Mensaje o Payload: Se trata del contenido del mensaje en sí, los datos que son transferidos por medio del protocolo y que tienen como objeto ser procesados por el Servidor/Cliente según sea su destinatario.

Evidentemente, aunque la estructura general sea igual existen algunas características específicas para las *Peticiones* como *Respuestas.* Para entenderlas a mayor profundidad, a continuación, se exponen las características específicas de cada tipo de mensaje[44].

Mensajes de Petición (Request): Como se mencionaba anteriormente, son aquellos mensajes que serán procesados por el Servicio *Web* para cumplir los requerimientos del cliente. Una solicitud está compuesta por los siguientes elementos.

- *Línea de Solicitud (Request Line):* Comienza otorgando el método de la solicitud y luego específica la versión del protocolo que se usará. Está compuesta por:
    - *Método de Acceso (Method):* Específica el método con el cual será accedido el recurso. Los métodos más conocidos son: *POST,* para enviar datos al servidor, *GET,* para obtener datos del servidor, *PUT* para subir archivos al servidor, *DELETE* para borrar datos del servidor, *TRACE* usado para identificar problemas de red y diagnóstico de errores y *OPTIONS* utilizado para obtener información sobre los métodos soportados por el Servidor *Web.*
    - *URI Solicitada (Request URI):* Se trata de una cadena que cumple el estándar de *Identificador Uniforme de Recursos* o *URI.* Éste identifica cuál es el elemento que se encuentra del lado del servidor que será solicitado. Para esta *URI* es muy importante tener en cuenta el método con el que será accedido.
- *Cabeceras de Solicitud (Request Header Fields):* Permiten al *software* del cliente especificar información y metadatos adicionales para ser enviados al servidor. Por lo general, estos metadatos definen atributos del cliente mismo y ayudan al Servicio *Web* a identificar cuál sería la mejor forma de comunicación entre ambos. Entre los *Headers* más conocidos se encuentran.
    - *Accept-Charset:* Set de caracteres que soporta el cliente.
    - *Upgrade-Insecure-Requests:* Indica al servidor que el *software* del cliente prefiere el uso de canales de comunicación seguros o *HTTPS.*
    - *Accept-Language:* Lenguaje soportado por el *software* cliente, en caso que el Servidor *Web* soporte internacionalización.
    - *User-Agent:* Específica la versión y *software* con la cual se está comunicando el cliente.

[44] XEROS, W3C/MIT, MICROSOFT et al. Op. cit., p. 54.

- *Referer:* Específica si el recurso accedido ha sido en realidad referenciado por algún otro recurso interno o externo al Servidor *Web.*

- *Proxy-Authorization:* Permite al cliente identificar el *software* o al usuario del mismo a un *Proxy* el cual requiere de autenticación.

- *Cookie:* Indica si la petición requiere de alguna *Cookie* en específico.

- *Identificador de Recurso por la Solicitud:* Para generar un identificar único, es necesario usar el *Header* que identifica al *host* en conjunto con la *URI* del recurso solicitado.

- *Parámetros de la Solicitud (Request Parameters):* En ocasiones, es necesario para el Servidor *Web* procesar cierta información relacionada a la lógica de negocio de las aplicaciones que se despliegan para otorgar una debida respuesta. Para ello puede ser necesario el envío de parámetros. Estos parámetros podrían ser enviados según el método, en la definición de la *URI* o formando parte del cuerpo mismo de la solicitud. Por lo general se puede observar la transferencia de parámetros en formularios *web* a través del método *POST,* o mediante el acceso a *URI* que tengan al final la estructura: "*?parametro=value"* a través del método *GET.*

Mensaje de Respuesta (Response): Cuando el servidor recibe una petición, éste la interpreta y procesa los *headers* y *parámetros* necesarios para así construir una respuesta. La estructura de la respueta cuenta con las siguientes características.

- *Línea de Estado (Status Line):* Consiste en la versión del protocolo seguido de un estado que indica si el servidor ha dado una respuesta consistente a la petición. Los códigos de estados son muy variados, pero básicamente se pueden distinguir entre las siguientes categorías:

  - *Estados 1xx:* Respuestas de información meramente.

  - *Estados 2xx:* Respuesta completada, acción recibida y entendida.

  - *Estados 3xx:* Respuestas de redirección, primero se debe llevar a cabo una redirección antes de completar la respuesta.

  - *Estados 4xx:* El cliente ha presentado un error, quizás la respuesta está mal construida.

  - *Estados 5xx:* El Servicio *Web* no puede procesar la respuesta ya que tiene un error de programación o lógica. Esta respuesta en particular es bastante

importante al momento de estudiar los posibles vectores de ataque del servidor.

- *Cabeceras de Respuesta (Response Header Fileds):* Así como las *Cabeceras de Petición,* permiten transferir información adicional al cliente, la cual no puede ser colocada en la *Línea de Estado.* Estas cabeceras especifican metadatos, en su mayoría, sobre cómo debería manejarse el *Payload* o *Cuerpo del Mensaje* el cual retorna el Servidor. Algunos de los más conocidos son:

  - *Location:* En caso de haberse obtenido una respuesta de tipo *3xx* indica la *URL* donde debe ser redireccionado el cliente.

  - *Set-Cookie:* Indica al cliente si debe establecer una *Cookie,* la cual puede tener como objetivo guardar información importante acerca de la comunicación Cliente/Servidor.

  - *Allow:* Indica cuáles son los métodos soportados por el Servicio *Web.*

  - *Content-Type:* Indica cuál es el tipo de contenido que tiene el *Payload* de la respuesta. Su valor coincide con el estándar de *MIME Types* y es clave para definir la interpretación por parte del *software* cliente.

  - *Content-Length:* Determina cuál es el tamaño total del *Cuerpo del Mensaje* o *Payload.*

- *Payload:* Cuerpo del mensaje de la respuesta. Usualmente corresponde al tipo mencionado en el *header, Content-Type.* En la mayoría de los casos se tratará de archivos *CSS, HTML* y *Javascript;* aunque también puede tratarse de *JSON, XML, YAML,* entre otros archivos varios.

Además de los *Mensajes HTTP* existen dos conceptos bastante importantes para este protocolo. Como se mencionó en un principio, *HTTP* no posee las capacidades de mantener un estado, es decir, siempre que se hace una petición el Servidor *Web* no puede identificar que se trata del mismo cliente que hace pocos minutos había realizado una petición. Para solucionar este inconveniente y no comprometer el rendimiento del protocolo, se crean los conceptos de *Sesión y Cookies* [45].

- *Sesión HTTP (HTTP Session):* Cada vez que se genera una nueva conexión al Servidor *Web,* desde las versiones *HTTP 1.1* se ha agregado la capacidad de identificar la sesión por cada conexión. De esta forma, por parte del Servicio *Web* se añade y reserva una porción de memoria dedicada a guardar datos específicos de dicha conexión. La sesión termina cuando el navegador o

[45] XEROS, W3C/MIT, MICROSOFT et al. Op. cit., p. 54.

software del cliente pierda comunicación por un tiempo predefinido con el Servidor. Los datos guardados en *Sesión,* virtualmente no pueden ser accedidos por ningún otro cliente.

- *Cookie:* Además de las sesiones *HTTP* existe otro elemento que puede ayudar a guardar información sin comprometer la memoria del Servidor. A este concepto se conoce como *Cookie* y es ampliamente usado para alivianar un poco la carga del Servidor *Web* en caso de ser accedido por muchos clientes. Las *Cookies* son implementadas en la mayoría de los casos, para guardar datos que el cliente o servidor necesitan consultar de manera frecuente.

b. Hyper-Text Transfer Protocol Secure HTTPS

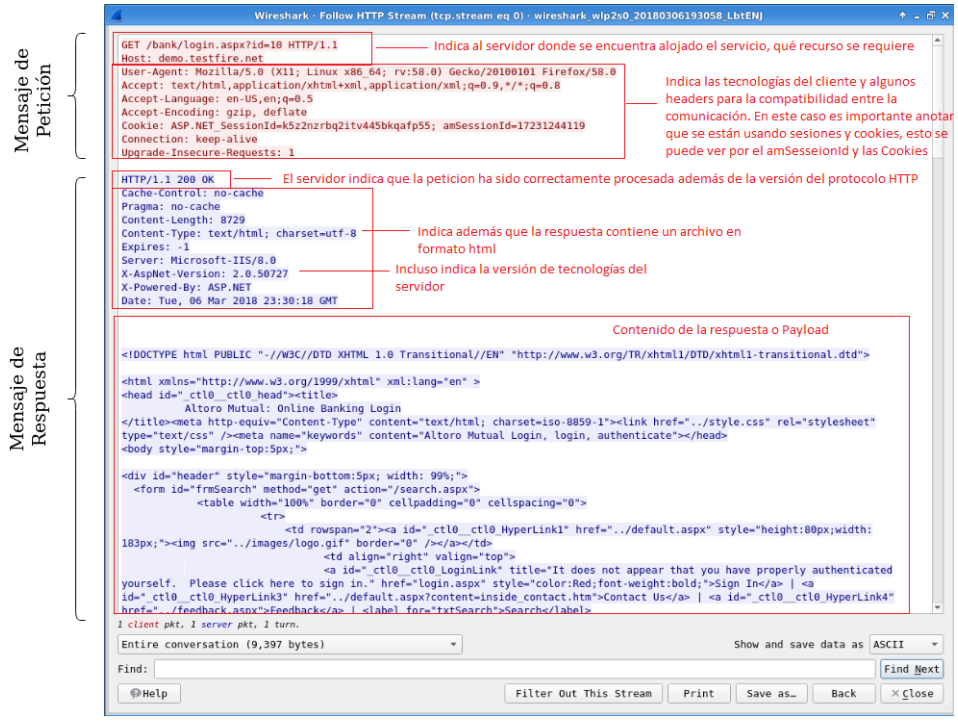
Se trata de la implementación de *HTTP* usando protocolos con soporte criptográfico como lo son *SSL* y/o *TLS.* Es utilizado para mejorar en gran medida la seguridad de *HTTP,* sobretodo en temas de confidencialidad e integridad de los datos. Los mensajes *HTTP* se encapsulan bajo tramas *SSL* y/o *TLS,* quienes se encargan de establecer la comunicación segura entre el cliente y el servidor. La implementación de *HTTPS* puede ser efectiva contra ataques del tipo *Eaveasdropping, Man in the Middle MitM* e incluso algunos vectores de ataque como *Inyección* y *Exposición de Datos Sensibles*[46].

c. Ejemplo de Petición y Respuesta HTTP

Para comprender un poco mejor la teoría expuesta, la Figura 6 muestra un ejemplo simple de una petición realizada por medio del método *GET* hacia un Servicio *Web.* En ella se pueden identificar algunos de los elementos descritos anteriormente.

[46] OWASP OPEN WEB APPLICATION SECURITY PROYECT. OWASP Top 10 [online]. OWASP ene, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf >

Figura 6 Trama de Petición y Respuesta HTTP versión 1.1 capturada con Wireshark



Fuente: El Autor.

d. Amenazas Comunes

El Servicio *Web* es uno de los servicios con mayor cantidad de amenazas, precisamente por ser uno de los más utilizados. Las amenazas a las que podría ser sometido pueden tener como objetivo la corrupción o explotación del mismo *software* que presta el servicio, o de alguna de las aplicaciones desplegadas en los mismos. Afortunadamente gracias a la organización *OWASP* (*Open Web Application Security Project*), desde hace varios años se ha venido priorizando algunas de estas amenazas clasificándolas en un *TOP 10* donde prevalecen las más conocidas. En esta sección, se expondrán algunas de estas amenazas teniendo en cuenta que muchas de ellas se relacionan al uso, configuración, instalación y despliegue de aplicaciones en un Servicio *Web*[46].

- Inyección (Injection o Parameter Tampering)

Algunas de las aplicaciones que son desplegadas bajo el Servicio *Web,* y en ocasiones el *software* que presta el servicio mismo; pueden tener el riesgo de

aceptar parámetros o datos que son modificables por el usuario final, que finalmente pueda ser interpretado como algún tipo de lenguaje de programación o *scripting,* desembocando en una ejecución de código. Debido a este comportamiento, donde se manipulan los parámetros de entrada se le conoce a este vector de ataque como *Inyección.*

Los datos que son manipulados por el usuario final pueden variar según la lógica de la aplicación y el servicio *Web* atacado. Básicamente pueden contener cualquier tipo de código como *SQL, LDAP, XPath, NoSQL,* Comandos del Sistema Operativo, *XML, ORM,* Headers *SMTP,* entre otros.

- Consecuencias

La inyección es una de las amenazas más graves que presenta el servicio y aplicaciones *Web.* Puede significar la pérdida de datos, corrupción, filtración de información, denegación del servicio y en ocasiones la *pérdida total del equipo* donde se encuentra alojado el servicio[47].

- ¿Cuándo se es vulnerable?

En general se es vulnerable cuando los datos que son enviados al servidor *Web* no son validados ni por mecanismos de control como *Web Application Firewalls WAF,* el Servidor *Web* o la misma aplicación que es encargada de procesar los datos. Permitir cualquier tipo de información modificable por el usuario final, sin una debida validación, puede ser bastante riesgo para las aplicaciones *Web* y para la integridad de la información que guarda el *Host.*

- Ejemplos

Un ejemplo bastante común es, en caso de aceptar un parámetro *command* el cual puede ser modificado por el usuario final y dicho parámetro se ejecuta en el Sistema Operativo, situación común en algunos *firmwares* para routers y otros dispositivos *IoT,* al menos se debe validar que exista una lista de comandos aceptados, de lo contrario se estaría dando puerta abierta a cualquier usuario malintencionado a ejecutar código de manera remota en el servidor.

Otro ejemplo puede suceder con las cabeceras *HTTP,* las cuales en ocasiones son procesadas por las aplicaciones o el Servidor *Web.* Supóngase un escenario donde

[47] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 59 p.

se esté usando una *Cookie* para obtener el nombre de usuario de cierto cliente, dicho nombre de usuario es utilizado para hacer alguna consulta a la Base de Datos de tipo: *SELECT * FROM CLIENTES WHERE USERNAME=””’+valor_cookie+”’”* si alguien llegase a enviar una *cookie,* con el valor de una ‘ inmediatamente obtendría un error *500* del servidor, esto porque ha llevado a cabo uno de los ataques más conocidos en aplicaciones *Web:* el *SQL Injection.*

- Autenticación Débil (Broken Authentication)

El Servicio *Web* puede proveer dos tipos de autenticación primordiales: aquellas basadas en el protocolo *HTTP* (Mediante *Headers* y mensajes especiales) y aquellas que se basan en formularios, mayoritariamente usando peticiones con métodos *POST.* El problema que existe con ello es que muchos de los Servicios *Web* contienen aplicaciones que cuentan con contraseñas y usuarios por defecto o que fácilmente pueden adivinarse[47].

Por otra parte, el uso de protocolos de comunicación inseguros y la falta de configuración de la aplicación y el servicio *Web* frente a tiempos de expiración de sesión y *cookies* pueden provocar que cualquier usuario malintencionado esté en condiciones de saltarse el proceso de autenticación.

Hacerse con las claves de usuario puede ser una amenaza para el servicio, puesto que dependiendo del grado de acceso otorgado por la aplicación; se podría estar vulnerando todo el entorno de configuración del mismo.

- Consecuencias

Depende intrínsecamente de la lógica de la aplicación *Web* que despliega el servicio, así como de las funcionalidades que puede ofrecer. Un sistema vulnerado por un acceso no autorizado puede significar fraude, corrupción de datos, filtración de información o falsificación de identidad.

- ¿Cuándo se es vulnerable?

Se puede ser vulnerable cuando se escogen credenciales de usuario que fácilmente pueden ser adivinadas mediante procesos de automatización. Cuando la aplicación es susceptible a ataques de fuerza bruta, cuando se expone información sensible en algunos de los archivos, URI o incluso dentro del contenido de los recursos descargados, cuando se utiliza el protocolo HTTP en vez de HTTPS, entre otros

escenarios, cuando los tiempos de sesión para cada usuario son bastante extensos[48].

- Ejemplos

Un Servicio *Web* en específico cuenta con un tiempo límite de sesión y expiración de *cookies* de más de un día. Suponga que un empleado utiliza este servicio *Web,* dejando su sesión abierta pues le incomoda tener que escribir sus credenciales repetidas veces o simplemente cerrar la sesión de la aplicación *web.* Un tercero que se percate del tiempo de sesión elevado para la aplicación *web,* simplemente puede acceder al sistema mientras el usuario que se encontraba logueado esté descuidado.

Un caso que sucede mucho con los *firmwares* de *IoT* que ofrecen paneles de administración por medio de aplicaciones *Web,* es que publican sus credenciales por defecto en las documentaciones de la aplicación. Muchas veces, cuando existen empresas que instalan dichas aplicaciones, no se percatan de esto y nunca cambian las credenciales por defecto de los usuarios. Esto se traduce a que cualquier sujeto que lea la documentación puede intentar loguearse con las credenciales por defecto, accediendo a la aplicación *web*.

- Exposición de Datos Sensibles (Sensitive Data Exposure)

Un servicio *web,* se traduce finalmente en un mecanismo utilizado para exponer aplicaciones *Web* que pueden ser desarrolladas por cualquiera que tenga conocimientos básicos en su estructura[48]. Este *software* puede ser, además, configurado por alguien que no necesariamente puede ser un experto en el manejo de *HTTP*. La exposición de datos sensibles, consiste en aprovecharse de estos descuidos de los desarrolladores, o en su defecto de quienes han configurado el servicio *web,* para obtener información que pueda usarse en otro tipo de amenazas.

Datos como: llaves privadas que se dejan en comentarios del código *HTML,* credenciales de usuario que son colocados en archivos que pueden ser accedidos por el mismo servicio *Web,* archivos de bitácoras o log's expuestos, datos personales sobre empleados o personas afiliadas a la aplicación *web*, conexiones a Bases de Datos y otros sistemas, son algunos de los casos más conocidos para este tipo de amenaza.

[48] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 59 p.

- Consecuencia

Como su nombre lo indica, su principal consecuencia es la filtración de datos privados, datos que no deberían conocerse por cualquiera sin que se haya llevado a cabo una correcta segregación de permisos y autenticación. Los datos que se pueden filtrar de esta manera pueden ser tan sencillos como el nombre de algún desarrollador que hizo la aplicación *web,* o incluso ir hasta la clave usada por el administrador del sistema para acceder al *host*[48]*.*

- ¿Cuándo se es vulnerable?

Cuando no se utiliza el protocolo *HTTPS* para cifrar las conexiones entre el cliente y el servidor y además no se toman las medidas de control necesarias para que no se realice un *downgrade*. Cuando se publica cualquier archivo del lado del servidor, sin verificar la audiencia que puede accederlo. Cuando se dejan comentarios por los desarrolladores en las aplicaciones *Web* y estos comentarios pueden ser visualizados por el cliente. Cuando se utilizan llaves criptográficas por defecto o son expuestas a los clientes.

- Ejemplos

Un sitio *web* puede ser accedido mediante *HTTPS* y *HTTP,* las cabeceras que han sido configuradas por el servicio *web* no proveen las restricciones necesarias para utilizar canales cifrados. Por esta razón, un atacante que se ha interpuesto en el medio de una comunicación *HTTPS* ha conocido esta característica y ha redirigido el tráfico de la víctima de *HTTPS* a *HTTP,* como el servidor acepta ambas conexiones; el atacante puede robar la *sesión* y *cookie* de la víctima, lo cual puede ser utilizado posteriormente para suplantar su identidad o secuestrar la sesión.

Un administrador de sistemas, que además es un administrador de Bases de Datos, ha decidido realizar un *backup* de los sistemas de gestión para cierta aplicación *web.* Como no desea olvidar que se trata del *backup* de cierta aplicación; ha decidido colocar el archivo resultante del proceso en un directorio *css* de la misma aplicación *web.* Algún usuario por simple azar, o haciendo uso de fuerza bruta de directorios, decide acceder a la *URI* que contiene a *css* descubriendo este archivo y estando en la capacidad de descargarlo.

- Control de Accesos Débil (Broken Access Control)

Muchos de los controles que existen en los servicios *web,* como se vio anteriormente, requieren métodos de autenticación los cuales a su vez están acompañados de procesos de autorización para así determinar si un usuario tiene o no acceso a un recurso. En ocasiones pueden existir recursos importantes que pueden ser accedidos sin necesidad de pasar por estos puntos de control. En pocas palabras, se salta el control de acceso[49].

También es posible que se manifieste esta amenaza cuando el proceso de segregación de permisos es inadecuado o cuando la lógica del servicio *web* o alguna de sus aplicaciones, otorga la capacidad de escalar privilegios adquiriendo mayores capacidades de acceso al sistema víctima.

- Consecuencias

Exposición de datos o información sensible, ya que se podría acceder a zonas que deberían estar restringidas. Ejecución de tareas con altos privilegios o suplantando identidades de otros usuarios[49].

- ¿Cuándo se es vulnerable?

Cuando los filtros que se aplican en la lógica de las aplicaciones *web,* pueden ser saltados fácilmente manipulando la estructura de las peticiones: es decir, cambiando el método, *URI,* parámetros, entre otros. Cuando se puede cambiar algún parámetro de identificador de usuario a una petición que segrega permisos; reemplazando valores en las *cookies* o *sesiones* actuales. Cuando se puede realizar una fuerza bruta para reconocer directorios o *URIs* que no son accedidas por medio de la interfaz gráfica[49].

- Ejemplos

Un atacante puede ejecutar una herramienta que automatice el acceso a directorios del servicio *Web,* identificando algunos recursos donde no estén aplicados el control de acceso.

[49] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 59 p.

Algunos recursos no se les aplica el control de acceso si son solicitados mediante métodos que no corresponden a una lógica de la aplicación *web,* pero que si están habilitados por el servicio *web.* Ejemplo: en vez de realizar una petición *POST* a cierta *URI,* llevar a cabo una petición *DELETE* o *PUT.*

- Configuraciones de Seguridad Débiles (Security Misconfiguration)

Los servicios *web,* como cualquier otro tipo de *software,* requieren de actualizaciones periódicas y revisiones cuidadosas sobre las configuraciones y opciones habilitadas. Cuando se accede a un servicio *web,* en ciertos casos específicos, un atacante puede reconocer la falta de una opción de configuración importante o un hueco de seguridad relacionado a una versión obsoleta del *software* que implementa el servicio[49].

- Consecuencias

Depende del tipo de falla en la configuración que se haya cometido. Una falla relacionada a la actualización puede variar desde una Denegación de Servicio hasta la pérdida total del *Hosts.* Si se trata de alguna configuración relacionada a los *HTTP Headers* del protocolo, puede ser una filtración de información o rompimiento de accesos de control y autenticación[49].

- ¿Cuándo se es vulnerable?

En pocas palabras, cuando no se es consciente de los requerimientos de las aplicaciones *web* y del servicio mismo. Cuando se aplican configuraciones desconociendo su impacto, y no se tienen en cuenta los planes de actualización constante.

- Ejemplos

Uno de los ejemplos más conocidos está relacionado al robo de *cookies,* incluso cuando una conexión es *HTTPS.* Suponga que existe un servidor *web* el cual por falta de configuración no ha colocado a las *cookies* el atributo de *requireSSL.* Esto permite que un atacante que coloque un *proxy* en medio de la comunicación del cliente y el servidor, pueda fácilmente reenviar la petición a un canal no seguro filtrando la información de la *cookie* que no tiene el atributo.

Algunos servidores *web* no cuentan con el concepto de *chroot jail* o *jaula.* Este concepto es aplicado para mitigar los ataques que intenten acceder a archivos que se encuentran fuera del directorio de trabajo del servidor *web,* ya sea por una vulnerabilidad de la aplicación o del mismo servicio. Si un administrador de sistemas no crea por sus propios medios las restricciones para que el usuario quien ejecuta el servicio *web,* sólo pueda trabajar en los directorios específicos; una vulnerabilidad que permita navegar por los diferentes archivos del Sistema Operativo puede convertirse en un verdadero problema.

- Cross-Site Scripting XSS

Uno de los recursos más consultados en las aplicaciones *web* son aquellos relacionados a la ejecución de *Scripts* del lado del cliente. Estos *scripts* son ejecutados mediante los navegadores y ampliamente son reconocidos dos lenguajes predominantes los cuales son: *Javascript* y *TypeScript.* Normalmente las aplicaciones *web* se valen de estos *scripts* para desarrollar lógica que atañe a la presentación de las aplicaciones. Esta amenaza trata del riesgo que puede existir al permitir a los usuarios finales la manipulación de datos modificables que finalmente se traduzcan o puedan traducirse en la ejecución de estos *scripts* del lado del cliente[50].

Cabe denotar que esta es la principal diferencia con la *Inyección,* ya que el objetivo final es el cliente más no el servidor *web.* Sin embargo, se estaría usando el servicio *web* para así poder ejecutar el vector de ataque. Existen tres tipos de *XSS,* aquellos que son reflejados; es decir que se reciben por medio de un parámetro de la solicitud la cual se ve reflejada en la respuesta, aquellos que son persistentes; los cuales se guardan de manera permanente en la página web o recurso atacado y aquellos que se basan en los atributos del árbol *DOM,* quienes se aprovechan de los atributos de los elementos *HTML* para llevar a cabo sus acciones maliciosas. Todos ellos pueden ser igualmente graves dependiendo de su manera de explotación.

- Consecuencias

Robo de *sesiones* o *cookies* si no se tienen aplicadas las configuraciones de seguridad necesarias. Corrupción de información y manipulación de la página *web* a gusto del atacante. Exposición de datos sensibles si se roban contraseñas o se aplican técnicas de seguimiento a las actividades de la víctima. Denegación de Servicio, en ciertos casos atípicos[50].

[50] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 59 p.

- ¿Cuándo se es vulnerable?

Cuando no se aplican los filtros necesarios para distinguir cuáles son los parámetros aceptables por el navegador. En caso de aceptar código, cuando no se aplica el concepto de *sandboxing.* Cuando no se codifican las salidas de las aplicaciones *web* en el lenguaje de formato adecuado.

- Ejemplos

Suponga que existe una página la cual acepta el parámetro *id,* siendo su estructura algo como */pagina?id=numero.* Ahora bien, si este número se ve reflejado en la respuesta tal cual se ha enviado y no requiere de un procesamiento o validaciones estrictas, cualquier usuario podría cambiar dicho parámetro por */pagina?id=<script>alert("XSS Reflejado")</script>.* Si el código se ejecuta, se tiene un *XSS* de tipo reflejado.

Otro caso similar sucede cuando existe información en Bases de Datos, que puede ser modificada por algún medio diferente al del servicio y acceso *web,* pero que finalmente se puede acceder por dicho servicio. Por ejemplo, suponga que un *software* tiene un método de importación de archivos *xml,* el cual es accedido de manera local y los resultados son impresos en la *web* para que sean visualizados por cualquiera. Alguien se percata que mucha de la información que se muestra en el *xml* también se muestra en la aplicación *web,* además que es guardada tal y como está en la Base de Datos que usa la aplicación. Simplemente, se puede cambiar el *xml* para incluir código *Javascript* o *TypeScript,* según sea el caso, para que así sea almacenado en la Base de Datos y se ejecute cuando se renderice la página vulnerable.

- Deserialización Insegura (Insecure Deserialization)

La transferencia de datos, en especial de objetos, se realiza del cliente al servidor y viceversa; por medio de datos serializados. Estos datos pueden ser representados de diferentes maneras como: *JSON, XML, YML, PHP Objects* o simplemente hexadecimales. Debido a la naturaleza de muchos lenguajes de programación y librerías usadas por los servicios *web,* algunos de los procesos de serialización y deserialización involucran la ejecución de métodos *create* y *destroy.* Cuando un objeto tiene definidos estos métodos, se ejecutan eventos específicos al momento de serializar o deserializar la información, esto se traduce en que un atacante puede ejecutar acciones a su conveniencia[51].

[51] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 59 p.

De igual manera se presenta la amenaza cuando la aplicación confía de forma ciega en los datos serializados/deserializados, sin hacer una comprobación de los mismos.

- Consecuencias

En la mayoría de los casos se hace referencia a la Ejecución de Código Remoto, aunque es probable que pueda usarse para Denegación de Servicio. También puede ser usado para evadir controles de acceso y escalar privilegios.

- ¿Cuándo se es vulnerable?

Cuando se implementan librerías vulnerables que llevan a cabo los procesos de serialización y deserialización. De igual manera, cuando no se aplica una correcta validación de los datos que serán serializados o deserializados. También se puede ser vulnerable al aplicar malas prácticas de programación.

- Ejemplos

Una aplicación declara una *cookie* donde se contienen los datos del usuario. Estos datos son: Nombre de Usuario, Token de acceso y roles. Si un atacante logra deserializar la *cookie* y cambia el valor de los roles; volviendo a serializar la *cookie* y enviándola al servidor, es posible que pueda practicar una escalada de privilegios.

Una aplicación de tipo *React* llama las tecnologías de *Spring Boot Micrsoservices.* Los desarrolladores, por falta de buenas prácticas, deciden no asegurar que el código puede ser inmutable y envían un estado serializado por cada solicitud/respuesta para interactuar con el usuario final. Un atacante se percata de los datos, encontrando la notación *R00 Java Object.* Luego procede a usar la herramienta *Java Serial Killer* para obtener la ejecución remota de código en la aplicación servidor[51].

- Ataques de Fuerza Bruta

A diferencia de las amenazas anteriores, esta no se lista en el *OWASP Top 10,* sin embargo; es de tener en cuenta. El protocolo *HTTP* al no manejar estados y no aplicar seguimiento a las solicitudes realizadas, puede ser atacado fácilmente por medio de fuerza bruta. Un ataque de fuerza bruta para un servicio *web* puede tener el objetivo de romper una autenticación, aplicar un descubrimiento de los directorios

de la aplicación *web* o saturar el servicio *web* aprovechándose de la definición de espacios en memoria para *sesiones* y *cookies.* Ahora bien, esta situación se complica teniendo en cuenta que en las versiones *HTTP 1.1* y menores, el protocolo y servicio tienen algunos problemas de rendimiento ya que no se definen múltiples canales de comunicación[51].

- Consecuencias

Se puede descubrir y reconocer la estructura de la aplicación. También se puede romper una autenticación, si se usan credenciales fácilmente predecibles. Finalmente, la mayor consecuencia puede ser un ataque de Denegación de Servicio si no se cuentan con los recursos necesarios.

- ¿Cuándo se es vulnerable?

Cuando el servidor *web* no cuenta con mecanismos de protección como *IDS/IPS, Web Application Firewalls* o configuraciones internas que permitan identificar cuáles han sido las conexiones fraudulentas. Cuando la lógica de la aplicación misma no restringe número de intentos o peticiones máximas. Cuando la estructura de la aplicación no cuenta con un control de acceso fuerte.

- Ejemplo

Dirbuster es una herramienta bastante usada para el reconocimiento de directorios ocultos por medio de ataques de fuerza bruta dirigidos por diccionario. Permite identificar cuáles son las *URI* que han otorgado respuestas con códigos interesantes, códigos como: *200, 302* y *500.*

Una aplicación *web* tiene un formulario login, el cual puede ser accedido un sinfín de veces sin establecer valores de *captcha,* ya que el desarrollador no le interesa cuántos intentos fallidos de autenticación puedan existir. Un atacante puede dirigir una amenaza de tipo fuerza bruta hacia el formulario de login, generando peticiones hasta finalmente dar con las credenciales de usuario necesarias para ingresar al sistema.

5.1.2.2 Servicio de Directorios y LDAP

Para comprender el funcionamiento del *Servicio de Directorios* es necesario hacer referencia al estándar *X.500.* Dicho estándar define la manera en que diferentes

componentes pueden cooperar para administración la información de objetos específicos como: organizaciones, usuarios, personas, entre otras. En la mayoría de los casos el estándar define una estructura de clave/valor de una Base de Datos o un conjunto de *Objetos* y *Atributos.* Este concepto es comúnmente llamado como *Directorio*. Algunos de los elementos clave de un *Servicio de Directorios* son[52].

e. Agente de Servicio de Directorio (*Directory Service Agent DUA*): Es el nombre que recibe el cliente en una comunicación con un directorio de servicios. Entiéndase por cliente, cualquier tipo de *software* que se esté usando para intentar acceder a la información del directorio.

f. Agente del Sistema de Directorio (*Directory System Agent DSA*): Se trata del nombre que se otorga al servidor, quien es el propietario de la información del directorio en general.

g. Árbol de Información de Directorio (*Directory Information Tree DIT*): Es la estructura la cual representa la Base de Datos de los *Servicios de Directorios.* Tiene una estructura arbórea y jerárquica, siempre partiendo de un elemento raíz.

h. Protocolo de Acceso al Directorio (*Directory Access Protocol DAP*): Normas de comunicación establecidas entre un *DUA* y *DSA* para permitir al usuario final acceder a los datos que se encuentran almacenados en el *DIT.*

En la mayoría de las implementaciones, la información debe ser codificada bajo el estándar *ASN.1 (Abstract Syntax Notation)* usando el formato *BER (Basic Encoding Rules)*. Ya que se trata de un sistema de almacenamiento, naturalmente la información que contiene el *DIT* puede ser consultada, añadida, borrada o modificada según los requerimientos del usuario. La estructura de un directorio de servicio podría ser modificada según la implementación. Los *Servicios de Directorios* usan un patrón de diseño *PKI (Public Key Infraestructure)* el cual se encuentra definido en el estándar *X.509*[53]*.* Otras especificaciones importantes pueden encontrarse en las siguientes especificaciones[52]:

[52] TIVOLI SOFTWARE. Understanding LDAP Design and Implementation. IBM: United States. 2004. 774 p.

[53] ORACLE. X.500 Standard [online]. Oracle mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://docs.oracle.com/javase/jndi/tutorial/ldap/models/x500.html >

- *X.501:* Define modelos y conceptos.
- *X.509*: Define los métodos de autenticación.
- *X.511:* Definición abstracta del concepto de servicio.
- *X.518:* Procedimientos y operaciones distribuidas, en caso de contar con múltiples servidores.
- *X.519:* Especificaciones de los protocolos relacionados a *X.500.*
- *X.520:* Selección de tipos de atributos y definición de los datos.
- *X.521:* Selección de Clases de Objetos y definición de clases.
- *X.525*: Replicación y comunicación con múltiples servidores

En otras palabras, el *Servicio de Directorios* está compuesto por un gran número de componentes que interactúan entre sí para proveer el acceso a cierto tipo de información, por esto el servicio es quizás uno de los más robustos que existe en el mundo de las redes. Debido a su nivel de abstracción elevado, su implementación puede variar desde el manejo de dominios, usuarios hasta información de países, idiomas, certificados, entre otros[53].

Como se ha mencionado anteriormente, uno de los elementos clave para el manejo del *Servicio de Directorios,* es la manera en que es accedido por medio del protocolo de red, es decir la comunicación *DAP*. El presente trabajo se centra en el estudio de amenazas dirigidas al este componente del *Servicio,* teniendo en cuenta la robustez del mismo y que un estudio detallado podría requerir todo un proyecto de investigación aparte. Para ser más precisos el proyecto se centra en una de las implementaciones más usadas de *DAP,* conocida como *Lightweight Directory Access Protocol* o *LDAP[54].* Existe una contraparte de *LDAP* el cual es conocido como *Heavyweight Directory Access Portocol HDAP,* pero no es tan usado por varios problemas de rendimiento, escalabilidad, costos y en especial porque carece de compatibilidad con las capas del *Modelo OSI* y *TCP/IP.* Por esta razón en el presente proyecto se hace referencia al *Servicio LDAP* en vez de *Servicios de Directorios* ya que busca precisar el campo de estudio abarcado por el proyecto.

Cabe denotar que *LDAP* en algunos Sistemas Operativos como aquellos basados en *Unix,* cuenta con un servicio específico. Aunque represente un *software* aislado, debe tomarse siempre como parte del concepto de *Servicios de Directorios.* Por otra parte, quien lleva en el mercado el liderazgo sobre el uso de *Servicios de*

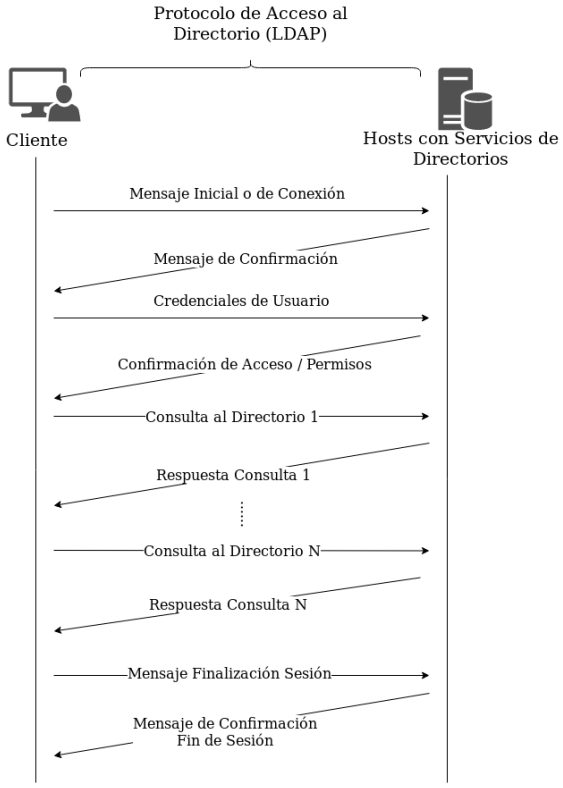
[54] TIVOLI SOFTWARE. Op. cit., p. 71.

Directorios, se trata de *Windows* con su implementación *Directorio Activo de Microsoft (Microsoft Active Directory),* dicha implementación también cuenta con soporte para *LDAP*[54]*.*

a. Funcionamiento Básico del Protocolo

El protocolo *LDAP* se ubica en la capa de aplicación según el *Modelo OSI* y *Modelo TCP/IP,* en su mayoría de implementaciones basa su funcionamiento sobre el protocolo de transporte *TCP,* aunque también puede ser soportado bajo *UDP. LDAP* cuenta con un manejo de sesiones propio por lo cual se sabe que tiene capacidades de manejo de estados e incluye de forma implícita los procesos de autenticación. La Figura 7 da un ejemplo de alto nivel de abstracción entre una comunicación del cliente *DUA* y el servidor *DSA* usando *LDAP.* Una de las ventajas de este protocolo es que puede usar de manera integrada la adición de *TLS/SSL* sin requerir alguna extensión como lo hace *HTTP.*

Figura 7 Flujo Básico de Comunicación por medio de LDAP entre el Cliente DUA y el Servidor DSA



Fuente: El Autor.

Como se puede observar en la Figura 7, básicamente se están ejecutando los siguientes pasos cada vez que un usuario o cliente desea consultar información del *Servicio de Directorios* del servidor[54].

1. El mensaje establece una conexión con el *DSA,* este concepto es conocido como *binding* y se realiza usando el *hostname* y la dirección *IP* del cliente al igual que el puerto *TCP.* En otras palabras, una conexión *TCP/IP* simple. El servidor contesta con un mensaje de confirmación de conexión, el cual podría indicar el *banner* o versión del protocolo de comunicación o bien del *software* que presta el servicio.

2. El cliente puede proveer unas credenciales de usuario necesarias para iniciar la sesión de comunicación. En ocasiones los servidores podrían estar configurados para aceptar accesos anónimos con lo cual el *DUA* no necesita agregar ningún tipo de credenciales. El servidor contesta con una serie de permisos e información que necesita el *DUA* para continuar la comunicación.

3. El cliente efectúa operaciones sobre los datos del directorio, por medio de consultas *LDAP* que están dirigidas básicamente al *DIT.* El servidor o *DSA,* está encargado de procesar cada una de estas consultas retornando la información o mensajes de confirmación según haya sido la consulta enviada. Dicho proceso se puede repetir hasta que el cliente pierda la comunicación o decida por propia voluntad cerrar su sesión.

4. Finalmente, el cliente o *DUA,* decide cerrar su sesión con el servidor o *DSA.* A este proceso se le conoce como *unbinding.*

Es necesario recordar que *LDAP* es una implementación de *DAP,* lo cual significa que se usan muchos de los conceptos del estándar *X.500.* Como nota especial, actualmente se usa la versión 3 del protocolo *LDAP* en la mayoría de las instalaciones[55].

Dentro de los *RFC* más importantes para comprender este protocolo se encuentran:

- *RFC 2251 Lightweight Directory Access Protocol v3.* Describe el punto de partida y estructura general del protocolo[56].

[55] TIVOLI SOFTWARE. Op. cit., p. 71.

[56] CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3) [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2251 >

- *RFC 2252 Lightweight Directory Access Protocol v3. Attribute Syntax and Definitions.* Describe la manera en que las cadenas de consulta son intercambiadas, cómo se definen los diferentes tipos de datos de *LDAP* en octetos de *bytes*[57].

- *RFC 2253 Lightweight Directory Access Protocol v3 UTF-8 String Representation of Distinguished Names* Describe la importancia de los *DNs* y cuál debe ser su codificación. La codificación puede diferir del estándar *X.500*[58].

- *RFC 2254 The String Representation of LDAP Search Filters.* Criterios de búsqueda para filtrar la información[59].

- *RFC 2255 The LDAP URL Format.* URL's usadas para llevar a cabo búsquedas en el servidor[60].

- *RFC 2256 A Summary of the X.500(96) User Schema for use with LDAPv3.* Diferentes tipos y clases de objetos que todo servicio que implemente *LDAP* debería reconocer[61].

---

[57] CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2252 >

[58] CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2253 >

[59] NETSCAPE COMMUNICATIONS CORP. The String Representation of LDAP Search Filters [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2254 >

[60] NETSCAPE COMMUNICATIONS CORP. The LDAP URL Format [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2255 >

[61] CRITICAL ANGLE INC. A Summary of the X.500(96) User Schema for use with LDAPv3 [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2256 >

b. Métodos de Autenticación

Debido a la información sensible que maneja el protocolo *LDAP,* es necesario definir unos métodos de autenticación que permitan proveer los mecanismos necesarios para asegurar el acceso legítimo al *DTI,* los métodos de autenticación se encuentran listados bajo el *RFC 2829 Authentication Methods for LDAP* [62], algunos de los más importantes son:

i. *Autenticación Anónima.* Como muchos otros servicios de red, el acceso al directorio por medio de *LDAP* puede darse por medio de la autenticación anónima. Esto quiere decir, el usuario no define sus credenciales de usuario por lo cual la petición inicial *binding* podría no completarse de manera exitosa. Para identificar un tipo de autenticación anónima debe enviarse una cabecera con un octeto de *bytes* específicos. Este tipo de autenticación puede ser peligrosa si no se lleva una correcta segregación de los permisos de usuario. Aunque exista una autenticación anónima, *LDAP* soporta la comunicación *TLS* para proteger los datos que viajan a través del canal de comunicación.

j. *Autenticación basada en Contraseña.* Cualquier implementación de *LDAP* debe utilizar soportar el método de autenticación por contraseña usando los mecanismos *DIGEST-MD5 SASL.* El cliente y el servidor deben ponerse de acuerdo en los mecanismos que usarán para llevar este tipo de autenticación durante el envío del paquete *binding,* este caso en particular sucede cuando se habilita un nuevo componente de autenticación llamado *DSE.* Otro posible sucede cuando *LDAP* basa su autenticación en un registro del mismo directorio que se desea acceder por medio del atributo *userPassword,* en este caso el cliente se puede autenticar compartiendo su contraseña al servidor por medio de una conexión *TLS.*

k. *Autenticación basada en Certificados.* El protocolo *LDAP* o cualquier implementación del mismo debe soportar el manejo de certificados para la autenticación por medio de llaves públicas/privadas. El manejo de certificados puede ser usado en conjunto con *TLS* para mejorar la seguridad del protocolo, además puede acoplarse al uso de *Public Key Infraestructure PKI.*

l. *Otros métodos de Autenticación.* Algunos métodos adicionales son soportados como el uso de *CRAM-MD5* en vez de *DIGEST-MD5,* el manejo

[62] SUN MICROSYSTEMS INC, EDB MAXWARE, OBLIX INC y Otros. Authentication Methods for LDAP [online]. Internet Engineering Task Force IETF may, 2000. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2829 >

de *Kerberos versión 4* y *GSSAPI.* Estos no se explican durante el proyecto ya que requieren de un conocimiento más avanzado en cuanto al despliegue de *Servicios de Directorio.*

c. Estructura Básica de Almacenamiento en Directorios

Para comprender el comportamiento de las peticiones *LDAP* es necesario conocer la estructura de almacenamiento de los directorios. El estándar *X.500* establece una jerarquía bastante simple de modo arbóreo basado en elementos conocidos como *registros.* Cada registro tiene un nombre que lo distingue y es único, dicho nombre se conoce como *Distinguished Name (DN).* El *DN* consiste una secuencia de diferentes partes que son llamadas *Relative Distinguished Names (RDNs).* Los *RDNs* son el átomo que compone a la estructura *DIT.* Para comprender mejor la definición de *DN* y *RDN* se puede pensar en un Sistema de Archivos de cualquier Sistema Operativo donde los *DN* hacen referencia a la dirección absoluta mientras que un *RDN* puede ser el nombre de la carpeta padre o nombre de los archivos[63]. Luego se encuentra el concepto de registro, cada registro contiene uno o varios atributos que lo componen; cada atributo contiene un valor[63].

Un ejemplo que puede coincidir con la estructura de un directorio, son precisamente los directorios telefónicos. Cuando se ingresa a un directorio telefónico, en primer lugar se tiene al nodo base que es el mismo directorio, luego se pueden observar algunos nodos que se desprenden del mismo las cuales pueden ser categorías; por ejemplo, teléfonos de abogados, restaurantes, oficinas, etc; luego cada una de estas categorías también puede estar subdividida por direcciones o zonas, después se puede encontrar un registro de una persona u empresa el cual se podría considerar como el registro, ahora cada registro puede contener información sobre diferentes teléfonos y direcciones; es decir, finalmente estos teléfonos y direcciones vendrían siendo los atributos del registro.

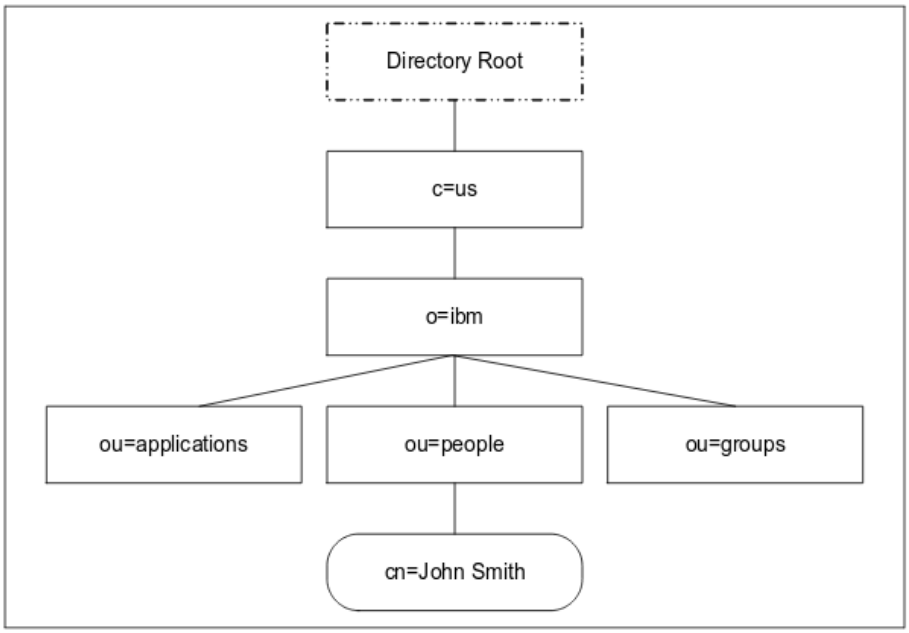
Finalmente, los registros pueden seguir un *template* específico. Dicho *template* o plantilla se llama *Clase de Objeto* y puede ser útil para representación de los metadatos de diferentes registros.

- El Modelo de Nombres

El modelo de nombres define la estructura general del *DIT* y cómo están organizados los diferentes registros que corresponden a *RDNs* o *DNs.* Para comprender rápidamente su funcionamiento se propone la siguiente estructura *DIT,* definida en la Figura 8*.*

[63] TIVOLI SOFTWARE. Op. cit., p. 71.

Figura 8 Estructura DIT Ejemplo para Modelo de Nombres LDAP.



Fuente: Tivoli Software [63].

En este caso el *DN* por ejemplo de *cn=John Smith* sería *cn=John Smith, ou=people, o=ibm, c=us.* Este *DN* estaría compuesto por los *RDNs cn, ou, o* y *c* correspondientes. Como se puede observar un *DN* básicamente es una cadena de *RDNs* separadas por coma. El *RDN* es usado para representar un nivel de jerarquía alto del elemento consultado. En el Modelo de Nombres de *LDAP* existen dos maneras de representar los *DN.*

a. *Forma de Cadena:* La forma de cadena es bastante similar a la del ejemplo anterior. Otro ejemplo podría ser, si se desea consultar el nodo *ou=groups,* se podría acceder al *DN ou=groups, o=ibm, c=us.*

b. *Forma de URL:* Se representan de una manera similar a la anterior, únicamente que hacen referencia al *host* y *puerto* del servicio *LDAP,* además de estar precedidas por el prefijo *ldap.* Un ejemplo de la forma anterior sería consultar por medio de *ldap://host:puerto/ou=groups,o=ibm,c=us.* Al igual que en *HTTP* en este caso los atributos pueden ser accedidos por medio del carácter "*?",* por ejemplo, *ldap://host:puerto/ou=groups?Nombre* aquí se estaría buscando al registro *c=us* y accediendo a su atributo nombre.

Las iniciales de los *RDN* pueden hacer parte de la especificación de *X.500* algunas de las más importantes son:

- *Common Name CN.*
- *Organizational Unit OU.*
- *Domain Component DC.*
- *Organizational O.*
- *Country C.*

Estas abreviaciones son únicamente para diferenciar jerarquías en un directorio, aunque no son estrictamente obligatorias en su implementación; muchos de los servicios o *software* que implementan *Servicios de Directorios,* las usan como estándar.

d. Consultas LDAP

En el punto anterior se observa la forma que son nombrados los registros que pertenecen a un directorio que soporta *LDAP.* Aunque esta pueda ser una manera de accederlos, podría ser complicado si no se conoce exactamente cuál es el registro que se desea buscar. Para ello existen las consultas *LDAP.* Estas consultas cumplen con un tipo de sintaxis similar al *SQL* y son usadas para buscar en su mayoría, cualquier tipo de registro. Para edición, eliminación y agregación se suele usar el *Modelo de Nombres* ya que es más específico.

En las consultas *LDAP* se pueden encontrar operadores tanto lógicos como relacionales. Algunos de ellos son:

- *Mayor (>), Menor (<), Mayor Igual (>=), Menor Igual (<=), Igual (=):* Como su nombre lo indica permiten comparar dos valores o atributos.
- *Operador Y (&), Operador O (|):* Permite relacionar varias premisas lógicas para crear consultas complejas.

- *Operador wildcard (*):* Indica que cierto atributo o registro puede tener cualquier valor.

- *Operador de Negación (!):* Niega una premisa lógica.

Algunos ejemplos del uso de búsqueda por medio de *LDAP* se listan a continuación.

(!Nombre=Jose) # Lista todos los registros cuyo atributo Nombre no sea Jose

(teléfono=*) # Lista todos los registros cuyo teléfono puede ser cualquier cosa

(&(Nombre=Jose)(teléfono=321)) #Lista todos los registros cuyo nombre es José y teléfono es 321

(&(objectClass=user)(|(Nombre=Jose)(Nombre=Juan))) # Lista los registros cuya clase de objeto sea usuario y los nombres sean Jose o Juan.

- Herramienta LDAP Search

En los sistemas operativos basados en *Unix* se puede encontrar una herramienta bastante útil para practicar la sintaxis de *LDAP* y la búsqueda por medio de *DNs, RDNs* y consultas. Dicha herramienta se llama *LDAPSearch* y puede ser ejecutada desde la consola de comandos. Existen algunos servidores públicos únicamente con permisos de lectura para practicar este tipo de sintaxis, a continuación, se disponen de los datos de un servidor público que puede ayudar a comprender el funcionamiento de este servicio[64].

Nombre del Host: ldap.forumsys.com

Base DN (Recomendado): “uid=tesla,dc=example,dc=com”

Estos son algunos de los ejemplos que han sido ejecutados por medio de la herramienta.

[64] YUNUS, Mammon. Online LDAP Test Server [online]. Forum System The Leader in API Security Management feb, 2014. [citado 22 mar., 2018]. Disponible en Internet: < https://www.forumsys.com/tutorials/integration-how-to/ldap/online-ldap-test-server/ >

ldapsearch –h ldap.forumsys.com –s base * + # Indica que se consultarán todos los DN Base

ldapsearch –H ldap://ldap.forumsys.com –x –s base –b “” # Misma consulta anterior con uso de forma por URL

ldapsearch –W –h ldap.forumsys.com –D “uid=tesla,dc=example,dc=com” “telephoneNumber=*” # Busca en el DN uid=tesla,dc=example,dc=com alguna entrada que tenga un atributo telephoneNumber con cualquier valor

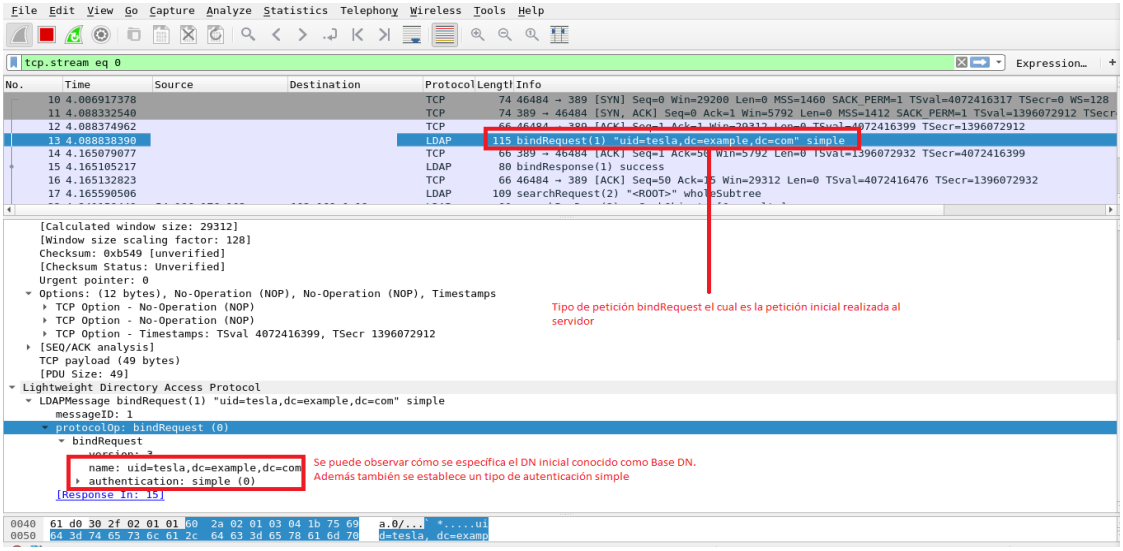
ldapsearch –W –h ldap.forumsys.com –D “dc=example” –s one # Obtiene todos los objetos en un nivel de jerarquía de grado 1, mayor al dc=example

e. Ejemplo de Comunicación LDAP

A continuación, se muestran algunas imágenes las cuales se expone una porción de trama capturada con la herramienta *wireshark.* En ellas se pueden identificar algunos de los componentes anteriormente explicados relacionados a las peticiones *LDAP.*

La Figura 9 expone la comunicación con el servidor *LDAP* llevada a cabo cuando se inicia una sesión. En ella se puede observar el tipo de autenticación y la petición inicial, también llamada petición *binding* con la cual se establece la comunicación con el *DSA.*

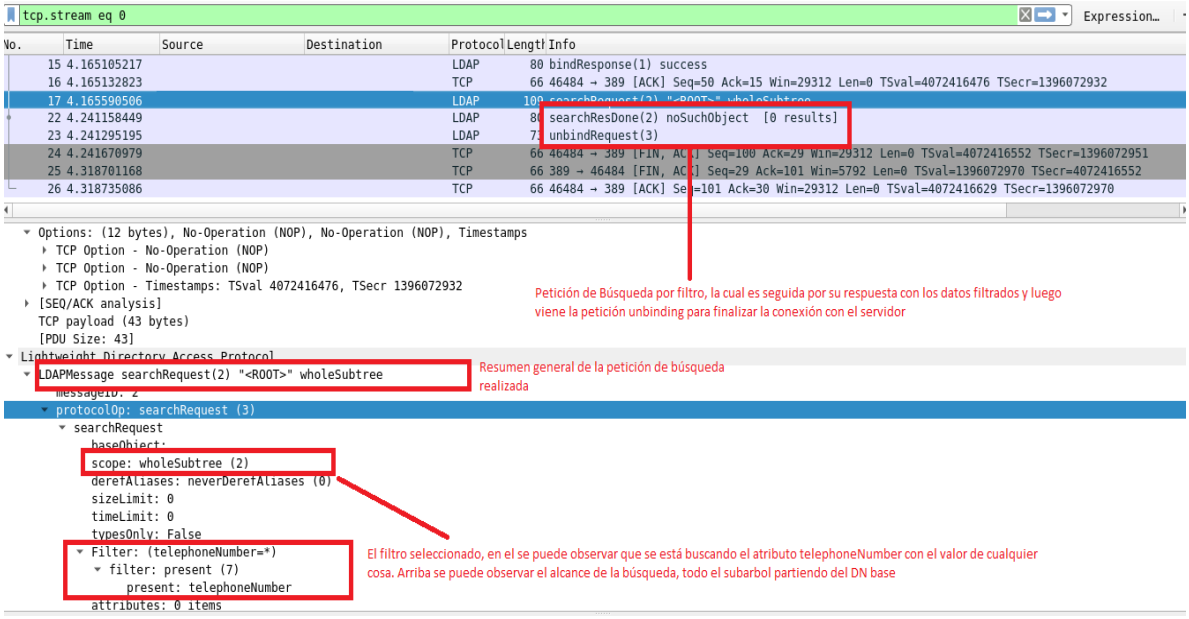
Figura 9 Ejemplo de Trama LDAP – Comunicación Inicial con el DSA



Fuente: El Autor.

Por otro lado; la Figura 10, muestra un mensaje intercambiado posteriormente donde se puede observar la realización de una búsqueda utilizando la sintaxis de las consultas. En ella se intenta buscar cualquier registro, tomando como base el *DN* anterior, donde exista un atributo *telephoneNumber* el cual contenga cualquier valor. También se puede observar que la comunicación es finalizada con la petición *unbinding.*

Figura 10 Ejemplo de Trama LDAP – Búsqueda por medio de Filtros



Fuente: El Autor.

f. Amenazas Comunes

Las amenazas dirigidas hacia los servicios que proveen comunicación bajo el protocolo *LDAP* son bastante reducidas teniendo en cuenta que su propósito se encuentra más precisado que otros servicios como *Web* bajo el protocolo *HTTP.* En su mayoría las amenazas se centran en el proceso de autenticación ya que es primordial para el acceso al directorio y a la información protegida por *LDAP.* Algunas de estas amenazas son.

- Modificación de Operaciones de Obtención de Datos

Un atacante que pueda interceptar los paquetes intercambiados entre el *DUA* y *DSA,* podría cambiar ciertos octetos de *bytes* para así modificar las consultas del usuario del servicio produciendo así que el servidor arroje resultados fraudulentos u obteniendo información confidencial que guarda el directorio[65].

- Consecuencias

Filtrado de información confidencial y exposición de datos sensibles, tanto de las empresas como los empleados o usuarios que usan el *Servicio de Directorios.*

- ¿Cuándo se es Vulnerable?

Cuando la configuración de la implementación *LDAP* no provee los estándares de comunicación bajo *TLS/SSL,* o en su defecto no han sido debidamente configurados bajo el manejo de llaves criptográficas fuertes y usando *PKI.*

- Ejemplo

Durante la exposición del funcionamiento del protocolo se pudo observar dos paquetes de *LDAP* que básicamente fueron recogidos por medio de la herramienta *wireshark.* El poderlos observar sin ningún problema, significa que la comunicación se estaba realizando sin las medidas criptográficas pertinentes. En la consulta *telephoneNumber=** cualquier atacante o individuo en medio de la comunicación podría modificar el paquete a *creditCard=*,* obteniendo registros con información bancaria de usuarios del directorio.

- Monitoreo de Métodos de Acceso Débiles

Si el servicio no se encuentra debidamente configurado un atacante puede aplicar un *downgrading* a los métodos de autenticación o simplemente atacar aquellos que presenten mecanismos de seguridad débiles. Esto puede ser mucho más sencillo si la comunicación que se realiza entre el cliente y el servidor carece de la

[65] TIVOLI SOFTWARE. Op. cit., p. 71.

configuración *TLS/SSL* recomendada para proveer mecanismos de manejo de integridad y confidencialidad[65].

- Consecuencias

Exposición de datos sensibles, filtración de información confidencial, modificación de la configuración del servidor *LDAP,* corrupción del *DTI* o directorio e incluso se podría perder acceso y control sobre el servicio dependiendo de los permisos del usuario vulnerado.

- ¿Cuándo se es Vulnerable?

Cuando el servicio que presta la comunicación *LDAP* no se configura de manera adecuada para establecer un máximo número de intentos de sesión fallidos. Cuando se establecen muchos mecanismos de autenticación que no se usan, por ejemplo, establecer una autenticación basada en certificados al mismo tiempo que se tiene un acceso anónimo ambos con los mismos permisos. Cuando no se establece una correcta segregación de permisos y accesos. Cuando no se configura el servicio para soportar comunicaciones *SSL/LDAP.*

- Ejemplos

Suponga que un *Servicio de Directorios* que es accedido por medio de *LDAP* soporta la autenticación basada en certificados, pero dichos certificados son autofirmados y no emitidos por una *CA.* Al no aplicar una correcta administración de los certificados, la llave privada de uno de ellos puede ser robada fácilmente por uno de los empleados de la compañía sin que el Administrador del Sistema se percate de ello. Luego con la llave privada de dicho certificado el atacante puede, en teoría, generar otros certificados públicos válidos para la autenticación.

Un *Servicio de Directorios* que puede ser accedido por medio de *LDAP* tiene configurada la autenticación *simple* la cual se basa en contraseñas. Así mismo también provee la autenticación anónima, la cual tiene permisos que no han sido distribuidos correctamente. El usuario anónimo podría modificar y consultar el atributo *userPassword* que se encuentra en el mismo directorio.

- Modificación de Datos no Autorizada

La amenaza anterior permite el acceso no autorizado al *Servicio de Directorios,* lo cual podría dejar a disposición del atacante la modificación y corrupción de la información del directorio[66].

- Consecuencias

Corrupción de información y datos confidenciales, daños severos de integridad de los datos del directorio y su estructura, posible Denegación de Servicio si se modifica la configuración de acceso. Si existe un cambio de configuración, se podría perder el control del servicio *LDAP*.

- ¿Cuándo se es Vulnerable?

Cuando no existe una correcta segregación de permisos en el directorio. De igual manera cuando los mecanismos de autenticación son débiles.

- Ejemplos

Un atacante se ha hecho con el acceso anónimo a un *Servicio de Directorios* por medio de *LDAP.* El atacante identifica la configuración de acceso de dicho servicio y procede a la modificación del mismo para que sólo él tenga acceso.

Un *Servicio de Directorios* guarda la información relacionada al número máximo de intentos fallidos de cierto Sistema de Información. Un atacante quien se ha hecho con el acceso a dicho servicio procede actualizar este valor a 1, muchos de los usuarios tienden a equivocarse al menos una vez al ingresar su contraseña; por lo cual sus cuentas podrían quedar inactivas.

[66] TIVOLI SOFTWARE. Op. cit., p 71.

- Denegación del Servicio por Uso Excesivo de los Recursos

Un atacante puede comprometer un *Servicio de Directorios* que use *LDAP* utilizando el típico ataque de Denegación de Servicio *TCP Flood* donde se envían muchas peticiones de tipo *TCP SYN.* Esto puede saturar la memoria del servicio evitando que ejecute y procese las solicitudes legítimas.

Del mismo modo, si el servicio *LDAP* usa una variante que soporte *UDP;* como lo es, por ejemplo, *CLDAP.* Los atacantes pueden llevar a cabo un ataque de Denegación de Servicio amplificado y reflejado, haciendo uso de *IP Spoofing* para enviar paquetes *UDP* hacia el *DSA* el cual respondería a la *IP* víctima.

Otro posible ataque puede llevarse a cabo por medio de las consultas y peticiones de alta complejidad, algo que requiere el consumo de muchos recursos por parte del *DSA*[66]*.*

- Consecuencias

Interrupción del servicio de directorio para los Sistemas de Información que se encuentran suscritos, algo que puede interrumpir otros servicios y actividades en un entorno empresarial. Posible culpabilidad en la partición de Denegaciones de Servicio Distribuidas DDoS al poder reflejar paquetes y además amplificarlos.

- ¿Cuándo se es Vulnerable?

Cuando los métodos de autenticación son débiles y permiten los accesos no autorizados. En ocasiones, cuando no se utilizan otros medios de control como *IDS/IPS* para monitorear el tráfico entrante y saliente de una red o simplemente para vigilar los paquetes de *LDAP.* Como se ha mencionado anteriormente, cuando se usan implementaciones que están basadas sobre *UDP* el cual no aplica validaciones ni sumas de comprobación.

- Ejemplos

Un atacante desarrolla una herramienta automatizada que simplemente envía paquetes de tipo *TCP SYN,* la herramienta apunta hacia un puerto donde se encuentra desplegado un *Servicio de Directorios* por medio de *LDAP.* Se envía un

sinfín de tramas *TCP SYN,* si el servidor tiene configurados algunos métodos de autenticación susceptibles; es probable que se esté saturando el acceso a *LDAP* impidiendo que cualquier otro usuario pueda autenticarse.

Un usuario malintencionado identifica que la trama de paquetes intercambiada con un *Servicio de Directorios* usa una implementación de *LDAP* bajo *UDP.* Dicho atacante, si tiene acceso al directorio, podría especificar un puerto y dirección *IP* falsas enviando varios paquetes de tipo *UDP* solicitando grandes cantidades de información al *DSA* los cuales responderán a la víctima causándole una Denegación de Servicio. Este ataque, además dejaría registrada la *IP* del servidor donde se encuentra el *DSA,* lo cual lo hace partícipe de una Denegación de Servicio ajena al conocimiento de sus administradores.

- Spoofing del Directorio

Básicamente, hacer creer a la víctima que se está accediendo a un directorio o *Servicios de Directorio* legítimos cuando en realidad se estaría estableciendo una conexión con un servicio que recoge información sobre las consultas realizadas y credenciales de acceso[67].

- Consecuencias

Exposición de datos sensibles al compartir credenciales de usuario y consultas que puedan revelar la estructura del *DTI* original, filtración de datos confidenciales, entre otros.

- ¿Cuándo se es Vulnerable?

Cuando no se tiene la manera de verificar la identidad del *DSA,* en pocas palabras, cuando no existe una implementación de los protocolos *TLS/SSL* por medio de certificados emitidos por Entidades Certificadoras oficiales. Aunque los certificados autofirmados puedan parecer una solución a primera mano, cabe recordar que no están corroborados lo cual los hace fácilmente falsificables.

[67] TIVOLI SOFTWARE. Op. cit., p 71.

- Ejemplo

En el ejemplo expuesto sobre las tramas de comunicación capturadas, un atacante que hubiese observado la red de manera pasiva podría haber notado el dominio al cual se establecía la conexión. Por medio de *DNS Spoofin* podría cambiar la dirección *IP* del dominio solicitado y usando otro tipo de directorio, con una estructura de *DTI* completamente diferente; podría capturar la información de las credenciales de usuario, además de capturar todas las consultas que pudieran ser realizadas al mismo. El cliente o usuario final, al no poder comprobar la identidad del servicio sería fácilmente una víctima de esta amenaza. Este escenario podría ocurrir incluso cuando se utilizan certificados autofirmados.

5.1.2.3 Servicio de Secure Shell (SSH) o Terminal Remota

Como su nombre lo indica, es un servicio que provee el acceso al intérprete de comandos de manera segura. Su objetivo es asegurar la confidencialidad y la integridad de los paquetes entre un cliente y servidor, incluso cuando se están comunicando sobre redes inseguras. Este servicio nace para mejorar la administración remota de los equipos, ya que otros servicios que usaban protocolos como *Telnet,* no aseguraban la conexión. Sus capacidades criptográficas proveen un cifrado fuerte, métodos de autenticación criptográficos y protección elevada contra la integridad. Cabe aclarar que, aunque el servicio inicialmente provee el manejo y administración de la terminal de forma remota, las implementaciones del protocolo pueden ir mucho más allá siendo implementado en ocasiones para asegurar las comunicaciones de otros servicios y encapsulando otro tipo de protocolos[68].

Este servicio es soportado de manera nativa para Sistemas Operativos basados en *Unix,* como lo son *Mac* o *Linux.* De igual manera otros sistemas como *BSD* lo soportan. En *Windows* puede requerir el uso de *software* como *Putty,* aunque es probable que en futuras versiones también se encuentre soportado de forma nativa.

[68] SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Transport layer Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4253 >

a. Funcionamiento Básico del Protocolo

El protocolo *Secure Shell* se ubica en la Capa de Transporte cuando es usado para establecer una comunicación segura, aunque si se habla de su capacidad de administración remota puede ser ubicado en la Capa de Aplicación, esto según el *Modelo TCP/IP* y *Modelo OSI.* Requiere de una comunicación *TCP/IP* para funcionar, en la mayoría de los casos. El protocolo ha sido diseñado para ser simple y flexible, a pesar de tener implementaciones criptográficas que podrían considerarse complejas. Todos los parámetros de este protocolo son negociables entre el cliente y servidor, por ejemplo: el método de intercambio, los algoritmos de clave pública, los algoritmos de cifrado simétrico, métodos de autenticación, algoritmos de *hash* y algoritmos de mensajes de autenticación *MAC*[68]. La versión actual del protocolo es la 2.0, aunque es común encontrar algunas implementaciones de versiones 1.x en *firmwares* de dispositivos obsoletos. Cabe destacar que este protocolo se basa en la transferencia de estructuras serializadas mediante *bytes,* por lo cual en sus diferentes *RFC* se pueden encontrar alusiones al lenguaje de programación C/C++[68]. Algunos de los *RFC* más importantes relacionados al protocolo son:

- *RFC 4253: The Secure Shell (SSH) Transport Layer Protocolo:* Habla sobre el funcionamiento del protocolo a nivel de la capa de transporte[68].

- *RFC 4254: The Secure Shell (SSH) Connection Protocol:* Expone el funcionamiento de las *pseudo-terminales,* Administración remota por medio de comandos, canales de comunicación, entre otros[69].

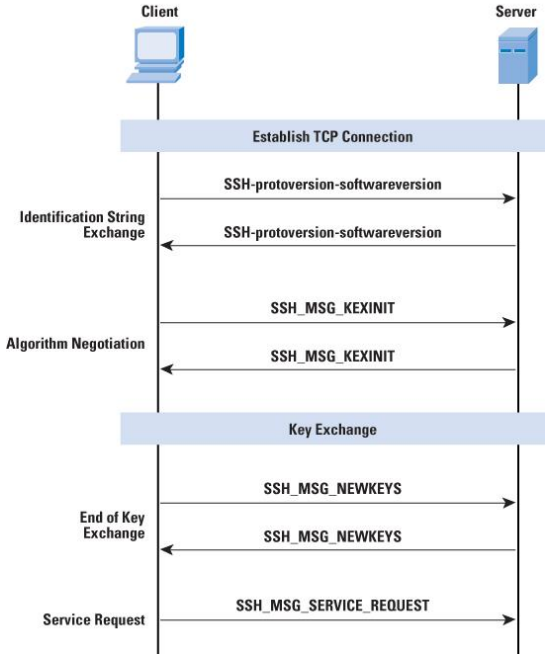
- *RFC 4252: The Secure Shell (SSH) Authentication Protocol:* Define el estándar de intercambio de información para métodos de autenticación y autorización[70].

La Figura 11 expone un ejemplo de la comunicación entre un Cliente y Servidor, por medio del protocolo *SSH.*

[69] SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Connection Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4254 >

[70] SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Authentication Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4252 >

Figura 11 Flujo de Comunicación Protocolo SSH.



Fuente: Cisco Systems[71]

Como se puede observar, existen diferentes estructuras para el intercambio de información. Cada una de estas estructuras establece un *frame* con cierta cantidad de octetos para sus diferentes cabeceras y *payloads.* Ahora, la comunicación requiere establecer algunas normas a nivel de la Capa de Transporte para así proceder al intercambio de información en la Capa de Aplicación, ambas capas según el *Modelo TCP/IP* o *Modelo OSI,* por esta razón en el diagrama se pueden encontrar algunas divisiones en Fases, las cuales se explican a continuación.

- SSH en la Capa de Transporte

En la capa de transporte *SSH* juega un rol fundamental para el aseguramiento de la información y la verificación de la integridad. Para lograrlo, debe proveer la

[71] STALLINGS, William. Protocol Basics: Secure Shell Protocol [online]. The Internet Protocol Journal Cisco Systems dic, 2009. [citado 21 mar., 2018]. Vol. 12, no. 4. Disponible en Internet: < https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html >

compatibilidad necesaria entre el cliente y servidor por lo cual, en cualquier conexión SSH se realizan los siguientes pasos básicos[71].

a. *Intercambio de Versiones del Protocolo:* Tanto cliente como servidor deben compartir una cadena conocida como, *cadena de identificación.* Esta cadena debe contener la versión del protocolo y la versión del *software* que se está usando. Esta fase también se conoce como *Identification Fase Exchange.*

b. *Intercambio de Claves:* Luego de compartirse las versiones del protocolo y *software* se procede a realizar el intercambio de claves, también llamado *KEX (Key Exchange).* En este punto, el cliente y servidor deben listar todos los algoritmos soportados; cada uno de ellos debe tener configurado algún algoritmo preferido o predeterminado. Si su contraparte soporta dicho algoritmo, la comunicación se debe realizar por medio del mismo. De lo contrario, si esto no sucede, la comunicación podría terminarse. En la Figura 11 esta etapa se puede observar en *Algorithm Negotiation* y *Key Exchange.*

   A continuación, se listan algunos de los atributos más relevantes determinados en la fase del intercambio de claves.

   - *Cookie:* Se trata de un valor aleatorio generado por quien envía la solicitud, s intención es hacer virtualmente imposible determinar a qué llave pertenece una sesión.

   - *Algoritmos de Llave (kex_algorithms):* La lista de algoritmos soportados tanto por el cliente como por el servidor. Como se ha mencionado anteriormente, se debe declarar un algoritmo predefinido el cual es el primero que se encuentra en la lista. Si el algoritmo seleccionado requiere el uso de otro algoritmo de cifrado, el servidor y el cliente deben tener soporte para dicho algoritmo de cifrado.

   - *Algoritmos de Llave del Servidor (server_host_key_algorithms):* Algoritmos soportados por el Servidor para el uso de llaves públicas/privadas. Un mismo servidor probablemente pueda soportar el uso de diferentes llaves, las cuales tienen diferentes algoritmos específicos; por esta razón este campo es requerido. El cliente debe aceptar al menos uno de los algoritmos listado por el servidor.

   - *Algoritmos de Cifrado (encryption_algorithms):* Lista de los algoritmos de cifrado de tipo simétrico que son soportados, deben ir ordenados según su preferencia. El algoritmo seleccionado debe ser el primero que se encuentra en la lista soportada por el cliente que también sea soportado por el servidor.

- *Algoritmos de Mensaje de Autenticación MAC (mac_algorithms):* Lista que contiene los algoritmos de cifrados específicos para Mensajes de Autenticación, igualmente ordenados por preferencia. Es seleccionado el primero que se encuentre en la lista otorgada por el cliente, que además se encuentre en el servidor.

- *Algoritmos de Compresión (compression_algorithms):* El protocolo *SSH* acepta la comprensión de paquetes para garantizar el rendimiento de la transferencia de datos sobretodo en redes con bajas tazas de transferencia. Se debe establecer una lista de algoritmos de compresión de paquetes de red que sean soportados tanto por el cliente como por el servidor, ordenados por preferencia; es seleccionado aquel primero soportado por el cliente que también es soportado por el servidor.

- *Lenguaje (languages):* En ocasiones es posible que el protocolo pueda tener soporte de internacionalización, si esta lista es proveída se aplica la misma lógica anterior para seleccionar los lenguajes usados. En lo general, este atributo no es configurado.

- *Primer Paquete de Intercambio Siguiente (first_kex_packet_follows):* Indica si el procesamiento de las llaves compartidas y secretas se ha dado correctamente por ambas partes. Estas llaves serán producto de este paso. Si alguna de las partes ha fallado en su cálculo, se debe enviar este mensaje notificando a su contraparte que ha ocurrido algún error y debe esperar un mensaje especial para continuar con el proceso.

- *Mensaje de Inicio (SSH_MSG_KEXINIT):* Este mensaje indica a ambas partes que se ha acordado un algoritmo de intercambio de llaves y es necesario colocarse en ejecución.

Lo anterior se traduce en paquetes que tienen la siguiente estructura.

```
byte         SSH_MSG_KEXINIT
byte[16]     cookie (random bytes)
name-list    kex_algorithms
name-list    server_host_key_algorithms
name-list    encryption_algorithms_client_to_server
name-list    encryption_algorithms_server_to_client
name-list    mac_algorithms_client_to_server
name-list    mac_algorithms_server_to_client
name-list    compression_algorithms_client_to_server
name-list    compression_algorithms_server_to_client
name-list    languages_client_to_server
name-list    languages_server_to_client
```

```
boolean      first_kex_packet_follows
uint32       0 (reserved for future extension)
```

El proceso anterior deja como resultado un *secreto compartido* llamado *K* y un *hash* de intercambio conocido como *H.* La autenticación y cifrado de los datos serán resultado de la aplicación de algunas fórmulas matemáticas sobre estos dos aspectos. En especial el *hash H* será usado los métodos de autenticación.

c. *Nuevo intento de intercambio de Llaves:* Este proceso sucede cuando por algún motivo, la negociación del intercambio de llaves ha fallado. Básicamente se vuelve a repetir el paso anterior, lo único que cambia es el valor de la *Cookie,* la cual debe ser única para asegurar la confidencialidad del protocolo.

d. *Finalización del Intercambio de Llaves:* Se envían mensajes de confirmación sobre el intercambio de llaves; de esta manera se puede comprobar que la negociación inicial fue correcta.

e. *Solicitud de Servicio:* Finalmente, cuando se establece la comunicación el cliente puede llevar a cabo una solicitud del servicio donde determinar cuál de los servicios disponibles por el Servidor puede usar. Algunos de los más conocidos como *ssh-userauth* y *ssh-connection,* los cuales hacen alusión al protocolo de Aplicación de SSH. Esta fase se ve reflejada en el diagrama como *Service Request.*

Es necesario anotar que durante el proceso de negociación se pueden llevar a cabo el procesamiento de diferentes algoritmos criptográficos, tanto de tipo simétrico como asimétricos. Por ejemplo, para obtener el *secreto compartido* se aplica un algoritmo *Deffie-Hellman,* estos procesos no se explican ya que se consideran fuera del alcance al objeto del proyecto.

- SSH en la Capa de Aplicación

Una vez se ha desarrollado el intercambio y negociación de los algoritmos, así como de la llave pública, se procede a llevar a cabo otro tipo de protocolo; en el caso de usarse el servicio de *Secure Shell* específicamente. El flujo básico consiste en:

a. *Llevar a cabo la comunicación de Autenticación del Usuario:* El servidor comienza a dirigir el proceso de autenticación, en caso de estar configurado. En primer lugar, se listan los métodos disponibles para la autenticación soportados por el Servidor. El cliente se encuentra en libertad de seleccionar cualquiera de los métodos listados. Cuando se selecciona uno de los métodos, se debe

compartir las credenciales de usuario teniendo en cuenta una estructura específica para la petición y respuesta[72].

b. *Intercambio de Información y Datos:* Según la configuración del Servicio, se procederá a intercambiar las peticiones de comandos y respuestas. El canal seguro ya ha sido establecido por lo cual no debería existir un riesgo de infracción a la confidencialidad y/o integridad de los datos.

b. Métodos de Autenticación

Como muchos otros protocolos y servicios de Red, *SSH* ofrece una variedad de métodos de autenticación. Las configuraciones más conocidas se ven a continuación.

- *Llave Pública:* Método de autenticación que siempre se encuentra requerido por defecto y es recomendado, de hecho, por el *RFC 4252* [72]. Este método se vale de las capacidades del cifrado asimétrico. Se define una llave privada la cual es usada para generar una llave pública firmada con dicha llave privada. La autenticación se basa en la validez de esta firma. La llave pública debe compartirse entre el cliente y servidor antes de iniciar la comunicación *SSH.*

- *Basado en Contraseña:* Utiliza el método clásico de nombre de usuario y contraseña. Es quizás el método de autenticación más débil que posee el servicio, ya que usualmente se asocia a los usuarios del Sistema Operativo.

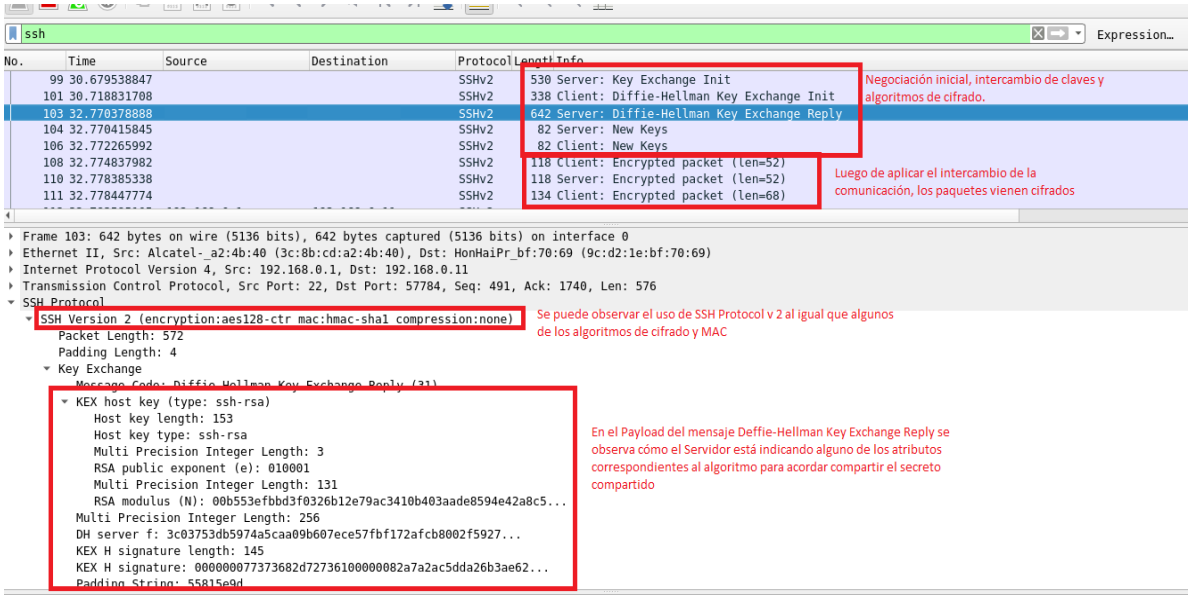
- *Autenticación basada en el Host:* Aplica un filtro de autenticación teniendo en cuenta algunos atributos del *host* del cliente. Este método de autenticación es opcional, aunque suele ser bastante conveniente. Utiliza una directiva de coincidencia para determinar si el nombre del *host* o su dirección *IP* se encuentra en un rango aceptado por el servicio.

[72] SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. Op. cit., 90 p.

c. Ejemplo de Comunicación SSH

La Figura 12 muestra cómo se lleva a cabo una comunicación inicial por medio de *SSH.*

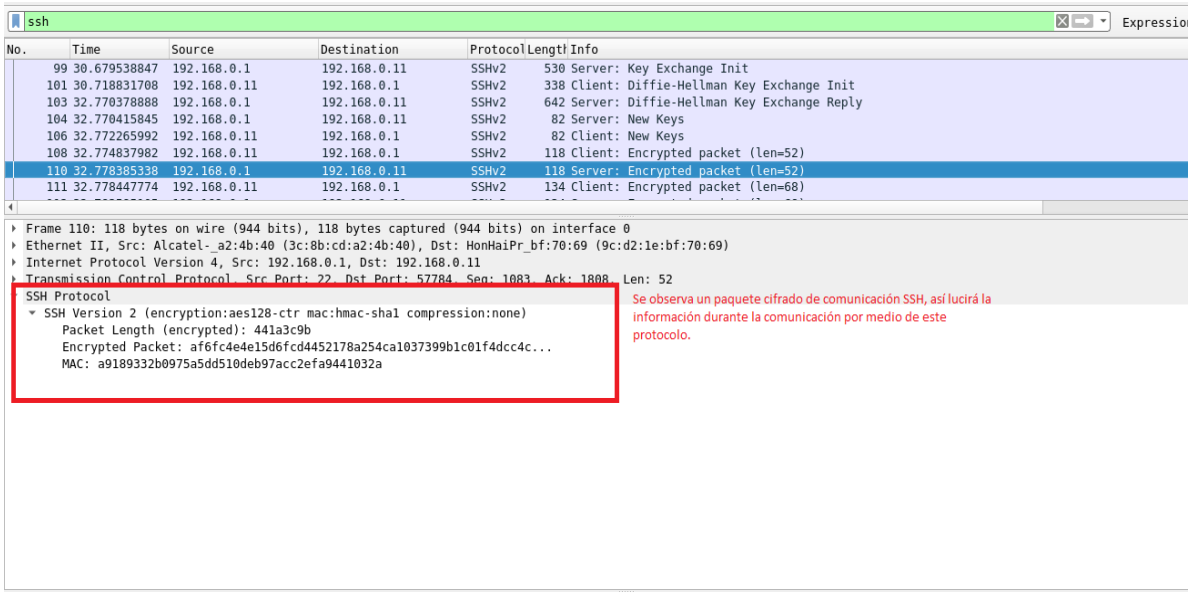
Figura 12 Ejemplo de Comunicación SSH en Wireshark – Negociación de Llaves y Algoritmos de Cifrado



Fuente: El Autor.

En primer lugar, se puede observar cómo ocurre el intercambio de las versiones de protocolos, luego se da lugar a el intercambio de llaves el cual incluye el intercambio del *secreto compartido* por medio del algoritmo *Deffie-Hellman.* Posteriormente; cuando se han ingresado las credenciales de usuario se puede observar el uso de una comunicación cifrada, la cual guarda la confidencialidad e integridad de los datos, esto se puede observar en la Figura 13.

Figura 13 Ejemplo de Comunicación SSH en Wireshark – Paquete Cifrado con SSH



Fuente: El Autor.

d. Amenazas Comunes

Aunque es considerado uno de los protocolos más seguros en redes, junto con sus implementaciones en servicios debido al alto uso de criptografía en la comunicación; existen algunas amenazas que no pueden ser prevenidas por su implementación. Estas son:

- *Password Cracking* y Ataques de Fuerza Bruta

No importa cuántos mecanismos de seguridad puedan ofrecerse a nivel de la red, las contraseñas son débiles o fuertes dependiendo de quien las haya implementado. El método de autenticación basado en contraseñas que ofrece *SSH* tiende a ser uno de los más vulnerados debido al descuido de los usuarios finales [73].

[73] BARRET, Daniel y SILVERMAN Richard. Threats that SSH Doesn't Prevent [online]. SSH: The Secure Shell The Definitive Guide. O'Reilly ene, 2001. [citado 21 mar., 2018]. Disponible en Internet: < https://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch03_11.htm >

- Consecuencias

Control total del Servidor donde se tiene configurado el Servicio *SSH,* corrupción de información, filtración de información confidencial y exposición de datos sensibles.

- ¿Cuándo se es Vulnerable?

Al confiar ciegamente en los métodos de autenticación basados en contraseñas. En la actualidad estos métodos deben ser abolidos completamente, teniendo en cuenta el avance indiscriminado de la tecnología; donde los ataques por fuerza bruta pueden ser ejecutados con mayor potencia. Para prevenir este tipo de ataques se debe considerar implementar una autenticación por *hosts* o por medio de *llave pública.*

- Ejemplos

Muchos de los *ISP* de la actualidad debido a la gran cantidad de *routers* y otros dispositivos *IoT* que deben administrar de forma remota, deciden implementar servicios como *SSH* para facilitar su trabajo. Para hacer aún más fácil las cosas deciden colocar una contraseña y usuario por defecto al servicio *SSH,* o en su defecto no cambian aquel que viene preconfigurado por fabricante. Cualquier usuario puede averiguar por medio de una consulta en Internet el uso de esta contraseña teniendo la capacidad de acceder al *router* de manera remota. Todo por falta de configuración.

Algunos proveedores de *SSH* tienen un problema en los tiempos de respuesta que permiten analizar la probabilidad de coincidencia de una contraseña. Esto quiere decir, que por ejemplo un usuario malintencionado analiza los tiempos de respuesta del conjunto de credenciales *x* e *y* en un servicio *SSH,* cambiando constantemente ambos valores y analizando que los tiempos de respuesta sean menores; puede darse una idea de los usuarios que están configurados en el Servidor. Posteriormente se puede inducir un ataque de fuerza bruta para *crackear* la contraseña.

- Ataques de Denegación de Servicio IP y TCP

SSH normalmente se ejecuta bajo los protocolos *TCP/IP* siguiendo la arquitectura ampliamente conocida en Internet. El problema de estos protocolos es que no ofrecen suficientes capacidades de tolerancia a fallos. Como consecuencia de ello un atacante puede llevar a cabo los siguientes vectores de amenaza [74]:

SYN Flood. Inundar el Servidor con este tipo de mensajes puede comprometer completamente los recursos del Servicio, si esto sucede *SSH* dejaría de contestar apropiadamente denegando las solicitudes legítimas.

TCP RST, bogus ICMP. Existen mensajes de tipo *ICMP* y *TCP* que pueden inhabilitar la conexión entre el Cliente y Servidor. Algunos mensajes de *ICMP* podrían determinar que el puerto solicitado por el usuario no puede ser alcanzado y de igual manera el mensaje *RST* de *TCP* puede reiniciar una conexión entre el Cliente y Servidor en cualquier instante de la comunicación.

TCP desynchronization and Hijacking. Si un atacante tiene conocimientos específicos sobre el protocolo *TCP* puede cambiar la sincronización y flujo normal de los paquetes, inyectando octetos de *bytes* lo cual se traduce en inyectar tráfico durante una conexión. Se considera Denegación de Servicio ya que *SSH* termina inmediatamente cualquier conexión que no cumple con los requisitos de seguridad.

- Consecuencias

Denegación del Servicio del servicio *SSH.* Si el *host* víctima únicamente puede ser accedido por este servicio; es posible que se pierda el control sobre el Servidor.

- ¿Cuándo se es Vulnerable?

Cuando no existen mecanismos de control como *Sistemas de Detección y Prevención de Intrusos IDS/IPS.* Cuando no se aplican técnicas como *Port Knocking* y se tienen configuraciones débiles en el *Firewall* del Servidor donde se encuentra el *SSH.*

[74] BARRET, Daniel y SILVERMAN Richard. Op. cit., p. 97.

- Ejemplos

Un atacante ha llevado a cabo con éxito el vector de amenaza *MitM,* con lo cual se ha establecido en medio de la comunicación entre una víctima y el Servidor *SSH.* Si dicho atacante comienza a editar los octetos de *bytes* de la comunicación, el servicio *SSH* terminará la sesión con el cliente legítimo. Si este proceso se automatiza y el *MitM* no es detectado, la víctima no podría acceder al *SSH.*

Un atacante puede automatizar la respuesta de paquetes *ICMP* que indiquen que cierto *hosts* o *puerto* no puede ser alcanzado por determinada dirección *IP.* Esto causaría que la víctima no pueda autenticarse o hacer peticiones al servicio *SSH.*

- Canales en Cubierto

La gran ventaja de *SSH* puede convertirse en su talón de Aquiles si es usado con fines maliciosos. *SSH* provee grandes capacidades de seguridad en la comunicación, lo cual lo convierte ideal para el filtrado de datos confidenciales. Un atacante que ya se encuentre en posesión del servidor, puede usar canales por medio de *SSH* para cubrir la información filtrada desde el Servidor[74].

- Consecuencias

Filtración de información confidencial y exposición de datos sensibles.

- ¿Cuándo se es vulnerable?

Cuando no se han aplicado mecanismos de monitorización a nivel local como *Sistemas de Detección y Prevención de Intrusos IDS/IPS.* De igual manera, cuando el equipo donde se encuentra instalado *SSH* ya ha sido comprometido por otro medio. Cuando no se aplica una correcta revisión de los archivos *log's* para *SSH.*

- Ejemplo

Un atacante ha podido ingresar a un Servidor Web que además contiene un Servicio de *SSH.* Esto lo ha logrado fácilmente por medio de la ejecución de código remota, es decir, una *Web Shell.* Como es probable que dicha *Web Shell* pueda ser identificada por los antivirus del *host;* el atacante decide modificar algunos de los archivos de configuración del Servicio *SSH* para permitir la conexión a su equipo. Así mismo se hace con la contraseña de uno de los usuarios del Sistema Operativo. A partir de ahora, el atacante puede gozar de una comunicación cifrada sin riesgo a ser detectado *IDS/IPS* a nivel de la red.

5.1.2.4 Servicio de Simple Mail Transfer Protocol SMTP o Transferencia de Correo Electrónico.

El correo electrónico es uno de los medios de comunicación, quizás, que ha presentado mayor acogida desde la aparición de Internet y la masificación de las telecomunicaciones. Éste, permite a las personas llevar interacciones de manera asíncrona con sus parientes y allegados que se encuentran distantes. Desde sus inicios, el servicio de correo electrónico ha sido comúnmente usados en las organizaciones, redes privadas y públicas para conectar a cierto grupo de personas. Este servicio de correos electrónicos, en su estructura incluye el funcionamiento de otro tipo de servicios que permiten el acceso a buzón, envío y transferencia de los correos, evaluación del contenido de los correos, adición de firmas digitales, entre otros[75].

El Servicio *SMTP* es uno de los componentes clave para prestar el Servicio de Correos Electrónicos, se encarga básicamente de transferir los correos independientemente de su red de origen y su red destino. Básicamente, hace posible que un mensaje pueda ser enviado desde cualquier sitio o lugar hacia otro. Su responsabilidad es únicamente la transferencia de los correos, a diferencia de otro tipo de servicios y protocolos como lo son *POP3, IMAP* y *SMAP [76];* quienes por su parte otorgan el acceso a los buzones de correo.

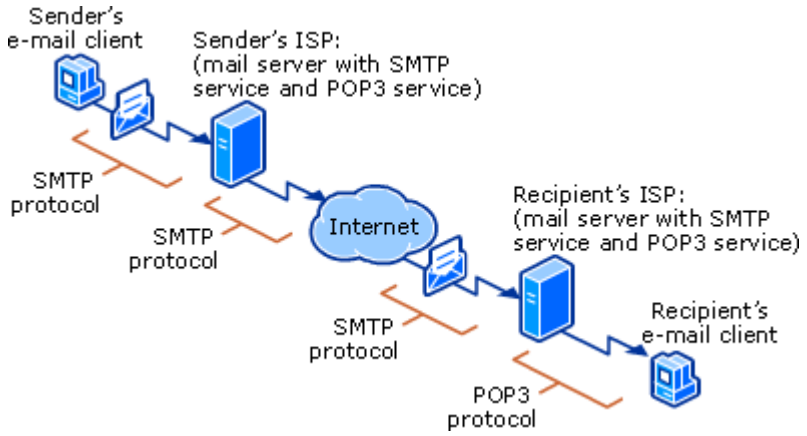
Se puede entender el rol del servicio dentro de la organización por medio de la Figura 14.

---

[75] KLENSIN. Simple Mail Transfer Protocol [online]. Internet Engineering Task Force IETF oct, 2008. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc5321 >

[76] SITEGROUND. Email Protocols – POP3, SMTP and IMAP Tutorial [online]. SiteGround mar, 2018. [citado 21 mar., 2018]. Disponible en Internet: < https://www.siteground.com/tutorials/email/protocols-pop3-smtp-imap/ >

Figura 14 Ubicación del Servicio y Protocolo SMTP dentro del Servicio de Correo Electrónico.



Fuente: Microsoft[77].

En otras palabras, se presenta un esquema completo del Servicio de Correo Electrónico cuando al menos existe un protocolo para el acceso a buzón de correo y otro que se encarga de su transferencia y envío. En la Figura 14, el encargado de acceso a buzón es *POP3* y quien lleva la transferencia de correo es *SMTP.*

La ilustración anterior expone además el uso de diferentes Servidores quienes se encargan de hacer el *forward* de los correos electrónicos hasta llegar a algún servidor que conozca la ubicación de su destinatario. Los elementos que participan en una comunicación *SMTP* son los siguientes[77].

- *User Agent:* Programa o aplicación usada por el cliente para acceder al correo electrónico. Normalmente se trata de una aplicación que se tiene instalada en el *host* tanto del emisor como receptor. Puede contener diferentes funcionalidades como el envío correos, archivos, consulta de buzón de correo entre otros. No se debe confundir con el acceso por medio de navegadores, pues éstos son un tipo de cliente para acceder al correo por medio de *HTTP.* También son conocidos como *Mail Agents MA* o *Mail User Agents MUA.* Algunos ejemplos en la actualidad son: *Mozilla Thunderbird, Outlook, MailBird,* entre otros.

- *Mail Transfer Agent:* Servidor encargado de la transferencia del correo electrónico mediante el uso de *SMTP.* Su tarea trata de recuperar el archivo recibido de un *User Agent* y ubicar a su Servidor destinatario para así realizar

[77] MICROSOFT. How POP3 Service Works [online]. Microsoft mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc737236(v=ws.10) >

la correcta transferencia. Algunos ejemplos, en cuanto a *software* se refiere, pueden ser: *Sendmail, Qmail, postfix, Exim, Mdaemon, Microsoft Exchange Servergotmail,* entre otros.

- *Mail Delivery Agent:* Servidor encargado de recibir el correo electrónico y guardarlo para que éste pueda ser accedido desde el *User Agent* del destinatario.

- *Mail eXchanger:* Recurso *DNS* que específica cómo se debe enrutar un correo electrónico a través de Internet. Apuntan a servidores de correo electrónico los cuales verifican cuál de ellos puede contener la dirección de correo especificada en un correo electrónico.

También es importante conocer las secciones que componen un mensaje de correo electrónico, teniendo en cuenta que más adelante muchos de estos campos podrán verse reflejados en el desarrollo de la comunicación del protocolo. Las siguientes, son las características más importantes de un mensaje de correo electrónico [78].

- *Destinatario (To):* Elemento básico de un correo electrónico. Indica hacia quien va dirigido el correo. Debe ser una o varias direcciones de correo sintácticamente válidas, es decir acompañada de *@dominio.com.*

- *Asunto (Subject):* Encabezado del correo electrónico, el cual es utilizado para describir la idea y asunto principal del envío del correo.

- *CC, CCO* y *BCC:* Correos electrónicos a quienes enviar una copia del mensaje. En caso de ser públicos se indica que es *CC;* esto quiere decir que el destinatario puede observar claramente las direcciones de correo a quienes se ha enviado el mismo. Cuando se trata de *CCO* o *BCC,* significa que será enviado el mensaje con *copia oculta;* es decir, el o los destinatarios no podrán ver estas direcciones. Deben ser una o varias direcciones de correo sintácticamente válidas, es decir, acompañadas de *@dominio.com.*

- *Cuerpo (Body):* La esencia y elemento principal del correo electrónico. Enmarca el mensaje como tal y lo que desea comunicar.

- *Archivos Adjuntos (Attachments):* Contenido adicional que puede contener el correo electrónico. Corresponde a un archivo que pueda ser catalogado dentro de los *MIME Types,* especificación que establece los diferentes tipos de archivo de multimedia que pueden ser transferidos en Internet.

[78] QUALCOMM INCORPORATED. Internet Message Format [online]. Internet Engineering Task Force IETF oct, 2008. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc5322 >

- *Firma (Signature):* Firma que se coloca de forma personalizada como pie de página para el cuerpo del mensaje del correo electrónico. No se debe confundir con la firma digital, ya que ésta no garantiza ningún tipo de autenticación legítima. Las firmas podrían encasillarse dentro del cuerpo del mensaje.

- *Firma Electrónica (Digital Signature):* Firma especializada en autenticar al emisor de un correo electrónico. No es estrictamente necesaria, pero en ocasiones es utilizada para impedir cierto tipo de ataques y fraudes.

Dependiendo del nivel de criticidad de un correo electrónico, éstos pueden ser cifrados o no. El proveedor y los usuarios pueden decidir hacer uso de certificados digitales para cifrar el contenido del mensaje, aunque en la actualidad, sólo se puede ocultar el cuerpo y adjuntos del mensaje mas no sus cabeceras, es decir: destinatarios, *cc, cco, bcc* y asunto.

Una vez definidos los conceptos que interfieren en la comunicación del envío de correos electrónicos de puede establecer el proceso para transferir este tipo de mensajes, el cual es el siguiente.

1. Suponga que José desea enviar a Juan un correo electrónico. El primer paso a dar es solicitar la dirección de correo, para lo cual requiere el contacto de Juan ya sea por alguna de sus redes sociales, páginas u otro medio.

2. Ya con el correo electrónico, José redacta un mensaje que desea enviar a Juan, una vez redactado el correo José establece diferentes atributos como lo son el asunto y el destinatario, quizás decida incluir copia oculta a otro destinatario de correo para dar constancia de su comunicación con Juan. Todo esto se hace mediante el *User Agent* de José, quien al final envía el correo.

3. Inmediatamente el *User Agent* se pone en contacto con su respectivo *Mail Transfer Agent,* el cual corresponda a su sección de red y dominio. Éste recupera el correo electrónico que ha sido enviado por José, por medio de la comunicación *SMTP* y al recibirlo comienza la búsqueda mediante consultas *DNS* sobre el dominio que usa la dirección de correo electrónico.

4. Si es necesario, el *Mail Transfer Agent* de José puede colocarse en contacto con otro *MTA* intermediario que pueda tener conocimiento sobre el destinatario, es decir, sobre la dirección de correo electrónico de Juan.

5. Suponiendo el caso de un intermediario, éste desarrollará la misma labor del anterior. El proceso deberá repetirse hasta dar con el *Mail Delivery Agent* que pueda tener la dirección de correo electrónico de Juan registrada en sus bases de datos.

6. Cuando se localiza el *MDA,* se envía el correo electrónico una vez más haciendo uso de *SMTP* y éste será asociado al buzón al que únicamente puede acceder el usuario Juan. De esta manera se habrá culminado la transferencia de correo electrónico.

En ocasiones, las consultas *DNS* o las direcciones de correo electrónico podrían no existir por lo cual el correo puede viajar durante varias horas e incluso días a través de diferentes servidores *MTA.* Sin finalmente no se encuentra cuál es el receptor del mensaje, se debe retornar un mensaje al emisor conocido como: *Delivery Message* o *Delivery Notifications,* indicando a su emisor que la dirección de correo no existe[79].

El proyecto se centra en el Servicio de *SMTP* teniendo en cuenta que es usado comúnmente para el despliegue de diferentes vectores de amenazas. Por esta razón no se ha dado mayor profundidad sobre el uso de *POP3, IMAP* y *SMAP.*

a. Funcionamiento Básico del Protocolo

El Servicio *SMTP* tiene un protocolo homónimo al mismo, por el cual se lleva a cabo la transferencia de los correos electrónicos. Este protocolo se encuentra especificado en los *RFC 821*[80]*,* para la comunicación y *RFC 822*[81] para la estructura del mensaje. Posteriormente fueron reemplazados por *RFC 2821*[82] y *2822*[83]*;* en la actualidad se trabajan en las versiones especificadas para *RFC 5321*[79] y *RFC 5322*[84]*.*

---

[79] KLENSIN. Op. cit., 101 p.

[80] POSTEL, Jonathan. Simple Mail Transfer Protocol [online]. Internet Engineering Task Force IETF ago, 1982. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc821 >

[81] CROCKER, David. Standard for the Format of ARPA Internet Text Messages [online]. Internet Engineering Task Force IETF ago, 1982. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc822 >

[83] RESNICK. Internet Message Format [online]. Internet Engineering Task Force IETF abr, 2001. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2822 >

[84] QUALCOMM INCORPORATED. Op. cit., 103 p.

Existen diferentes proveedores de servicio de *software* y *hardware* que pueden otorgar el acceso a este protocolo, es decir, el propio Servicio de *SMTP.* Comúnmente se encuentran asociados a otro tipo de soluciones como la prestación de servicios de dominio. El protocolo *SMTP* no cuenta con una versión específica; ya que en realidad no se cuenta con un cambio completo y significativo desde su aparición en el *RFC 822*[80]*.*

Como dato informativo, inicialmente *SMTP* sólo soportaba la transferencia de correos electrónicos en texto plano sin el soporte alguno de adjuntos u otro tipo de codificación diferente al *ASCII,* en la actualidad gracias a otro tipo de estándares como *MIME Types* y *ESMTP,* se han agregado las capacidades necesarias para el uso de adjuntos y multimedia. De igual forma, en sus principios únicamente se transfería el texto de forma plana y sin cifrados; lo cual hacía que el protocolo fuese bastante vulnerable. Debido a su nivel de confidencial para muchas de las compañías que lo implementan, desde el *RFC 5321*[79] se incluye la instrucción *STARTTLS* la cual agrega soporte nativo a *TLS/SSL* para el protocolo. De igual manera, es común encontrar en la actualidad implementaciones que trabajen con certificados públicos y privados que permitan el cifrado completo del cuerpo del mensaje otorgando mayor confidencialidad e integridad a la información.

El protocolo *SMTP* se encuentra basado en sesión lo cual permite la relación entre sus diferentes peticiones. Este protocolo se encuentra ubicado en la Capa de Aplicación del *Modelo TCP/IP* y *Modelo OSI,* en su mayoría de implementaciones trabaja sobre el protocolo *TCP* de la Capa de Transporte, aunque puede trabajar en cualquier otro tipo de topología que provea la transferencia de octetos de *bytes.*

En la actualidad el protocolo *SMTP* ha sido ampliado y mejorado por *ESMTP* el cual se encuentra establecido en el *RFC 1869*[85]*.* Esta nueva estandarización establece el uso de nuevos comandos para *SMTP,* expandiendo las capacidades iniciales del protocolo*.*

- Flujo básico del protocolo

A continuación, se describe de forma breve algunos de los procesos y tareas más importantes del protocolo como lo son: El inicio y finalización de sesiones, las

[85] INNOSOFT INTERNATIONAL INC, DOVER BEACH CONSULTING INC, NETWORK MANAGEMENT ASSOCIATES INC et al. SMTP Service Extensions [online]. Internet Engineering Task Force IETF nov, 1995. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc1869 >

transacciones de envío de correo electrónico, el *reenvío* de correo electrónico o *Mail Forwarding,* la verificación de nombres de buzón o correo electrónico, entre otros[86].

- *Inicio de Sesión:* La comunicación *SMTP* inicia con un mensaje que es lanzado desde el Cliente al Servidor donde se abre una conexión. En este primer mensaje inicial se debe otorgar la versión del *software* del servidor para que el cliente pueda evaluar si acaso dicha versión del *software* soporta su implementación de *SMTP.* La respuesta del servidor también puede especificar un *banner* o mensaje de bienvenida.

- *Inicio del Cliente:* Una vez se ha establecido la conexión con el servidor, el cliente normalmente envía el comando de *EHLO* el cual indica al servidor la versión de implementación de *SMTP.* Antiguas implementaciones pueden indicar el uso de *HELO.* Por medio de este comando el usuario se identifica ante el servidor indicando cuál es su dominio, en caso que pueda aplicar. El Servidor debe contestar con una lista de comandos soportados para la comunicación *SMTP.* En este punto el cliente puede escoger si llevar a cabo una comunicación cifrada por medio de *TLS/SSL* haciendo uso del comando *STARTTLS.* Para ello el Servidor debe tener soporte para este tipo de cifrado seleccionado durante la negociación.

- *Autenticación:* Una vez se ha iniciado el cliente y la sesión inicial, es posible que el usuario que se encuentra usando el protocolo *SMTP* requiera autenticación. Para ello más adelante se explican los métodos disponibles para darse de alta dentro del Servicio de *SMTP.* La autenticación se da por medio del comando *AUTH.*

- *Transacciones de Correo:* Las transacciones de envío de correo se inician con el comando *MAIL,* el cual contiene la sintaxis *MAIL FROM:<lista de correos, correo electrónico> [SP <parámetros-correo>] <CRLF>.* Esto causa que el Servidor reinicie el *buffer* de memoria destinado para guardar algunos estados de la sesión con el cliente, dicho *buffer* recibe el nombre *Tabla de Estados.* Cuando se inicia la transacción, existe un tiempo máximo de 5 minutos para culminar la transferencia del resto de parámetros y datos necesarios para el envío del correo electrónico. La transacción de *MAIL* es finalizada normalmente con el carácter del *"."* En las versiones antiguas, para las versiones nuevas se puede establecer un parámetro de finalización de transacción. Desde que es lanzado el comando *MAIL* y el Servidor responde correctamente, éste puede enviar comandos de tipo *RCPT,* el cual indica los

[86] HIGTY. Understanding the Insides of the SMTP Mail Protocol: Part 1 [online]. CodeProject dic, 2012. [citado 22 mar., 2018]. Disponible en Internet: < https://www.codeproject.com/Articles/399207/Understanding-the-Insides-of-the-SMTP-Mail-Protoco >

destinatarios del correo electrónico. La sintaxis de *RCPT* es similar a *MAIL;* siendo *RCPT TO:<dirección correo electrónico> [SP <Parámetros-RCPT>].* Cada uno de estos comandos debe ser contestado por el servidor, quien revisa la sintaxis y validez de los destinatarios. Cuando el Servidor contesta con un mensaje de confirmación se envía el comando *DATA,* quien es encargado de transferir el cuerpo y contenido del correo electrónico, teniendo en cuenta sus diferentes adjuntos. Toda la transferencia de binario se hace en este punto hasta el momento que se recibe el parámetro de finalización de la transacción *MAIL*[86]*.*

- *Forwarding:* Por estándar todos los Servidores *SMTP* deben soportar el *forwarding* para así soportar el envío de correos electrónicos a través de diferentes redes y servidores. Por razones de confidencialidad actuales, el *forwarding* debe contar con un modo silencioso el cual permita algunas ventajas en la confidencialidad de la información.

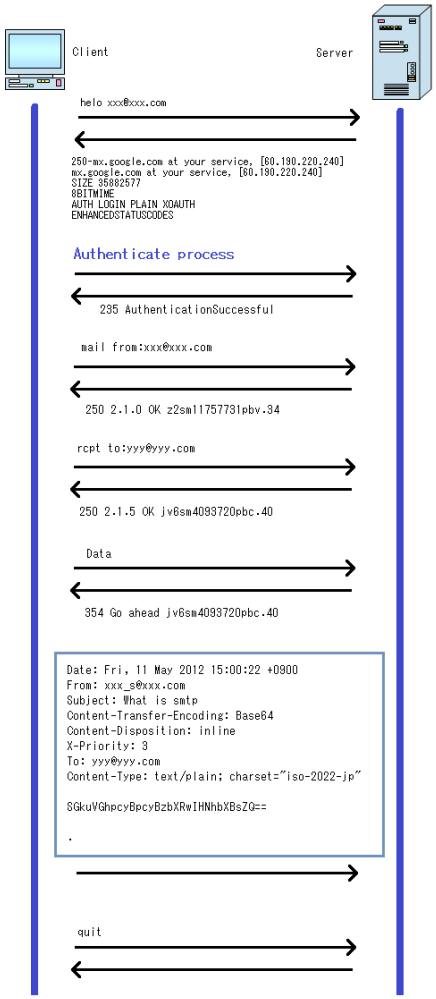
- *Verificación de Correos Electrónicos:* La verificación de los destinatarios se hace bajo los comandos *VRIFY* y *EXPN,* los cuales preguntan al Servidor de correo sobre la existencia de un correo electrónico[86]. Por estándar, las respuestas a este tipo de comandos pueden estar deshabilitadas ya que pueden significar un riesgo de seguridad en la fase de reconocimiento. De igual manera, muchos *firewalls* y mecanismos de control están configurados para no responder este tipo de mensajes. Ambos mensajes son utilizados en especial para efectos de *debugging.*

- *Relaying* y enrutamiento: *Relaying* es la capacidad de un Servidor *SMTP* de enviar transacciones de correos electrónicos que han sido generadas por usuarios o direcciones que no corresponden a su rango establecido, por ejemplo, enviar un correo desde la dirección *@usuariofalso.com* desde un Servidor configurado para el dominio *@usuario.com.* Por estándar actual, el cual se contrapone a los anteriores, no se debe permitir este tipo de comportamientos por temas de seguridad. Para ello existen los mecanismos de *Mail eXchanger.* De igual manera, cuando *SMTP* encuentre una ruta específica debe ignorarla y asegurarse de ubicar el Servidor del punto final para así garantizar la confidencialidad de los datos. Existen excepciones a estos escenarios en casos específicos donde debe ser necesario el uso de estos mecanismos para conseguir que el correo electrónico sea enviado.

- *Finalización del Cliente y la Sesión:* Una vez el cliente haya realizado sus transacciones de *SMTP* se debe enviar un código con el comando *QUIT* para finalizar la conexión. El Servidor de ninguna manera puede terminar la sesión por sus propios medios en un escenario ideal, a menos que se dé la ocurrencia de un error.

La Figura 15 muestra el flujo de *SMTP* o parte de él, que anteriormente ha sido explicado.

Figura 15 Diagrama de Flujo Básico SMTP.



Fuente: CodeProject[87]*.*

Como se puede observar primero se está realizando una petición con el código *HELO,* el cual es respondido por el Servidor *SMTP* con el mensaje de bienvenida y algunos otros parámetros importantes. Luego se aplica el proceso de autenticación, posteriormente se inicia el proceso de transacción de correo electrónico donde se envía un correo a la dirección *yyy@yyy.com* desde *xxx@xxx.com*. Luego de especificar los atributos de las cabeceras, el *Payload* o contenido binario es

[87] HIGTY. Op. cit., p. 107.

intercambiado mediante el comando de *DATA,* finalmente se finaliza la sesión por parte del cliente con el comando *QUIT.*

A continuación, se lista la estructura de los códigos de respuesta de un Servidor *SMTP*[87]*.*

- *1xx:* Indica problemas de conexión de red entre el cliente y el Servidor *SMTP.*

- *2xx:* Mensajes de confirmación desde el Servidor *SMTP.*

- *3xx:* Mensajes de respuesta para solicitar mayor cantidad de octetos binario en el cuerpo del mensaje.

- *4xx:* Este tipo de códigos se responde cuando se presentan errores del lado del Servidor, algún suceso de red o escenario inesperado que ha impedido el envío de correo.

- *5xx:* Indica problemas de sintaxis en el envío de correo electrónico.

b. Métodos de Autenticación

Inicialmente el protocolo *SMTP* era *Open Relay,* esto quería decir que cualquier usuario podría enviar correos electrónicos sin necesidad de autenticarse. Debido a la gran cantidad de *SPAM* y otro tipo de amenazas en la actualidad, se ha optado por la restricción de este tipo de capacidades. Por ello se ha asignado a *SMTP* la capacidad de realizar procesos de autenticación, de los cuales los más destacados se listan a continuación[88].

- *Texto Plano:* La autenticación sucede por medio de texto plano, es decir las credenciales de usuario se comparten sin ninguna protección. Esta puede estar basada en *ASCII* de forma pura o soportar la codificación en *Base64.* Este método de autenticación es obsoleto y corresponde un gran hueco de seguridad para muchas de las implementaciones de *SMTP.*

- *Generic Security Services Application Program Interface GSSAPI:* Otorga un estándar donde se implementan algunos de los servicios de cifrado necesarios para llevar a cabo una autenticación segura. *GSSAPI* no establece el uso de un tipo de seguridad específica pero sus diferentes

[88] KLENSIN. Op. cit., 101 p.

implementaciones deben otorgarlo siguiendo la estructura de la especificación genérica.

- *DIGEST-MD5, MD5* y *CRAM-MD5:* Métodos basados en el uso de *hashes MD5* y otro tipo de algoritmos como *HMAC-MD5* para proveer los mecanismos de autenticación contra el servicio. Se caracterizan porque ofrecen un buen nivel de seguridad contra ataques de tipo *Replay.*

- *OAUTH10A* y *OAUTHBEARER:* Implementaciones de *Open Authorization,* que permiten la interacción de una aplicación o Servicio tercero para llevar a cabo flujos de autorización y autenticación básicos. Se establece como una de las formas más seguras de acceder el día de hoy a correos electrónicos y Servicios *Web HTTP*. Actualmente se encuentran en la versión 2 especificadas en el *RFC 8252*[89]*.*

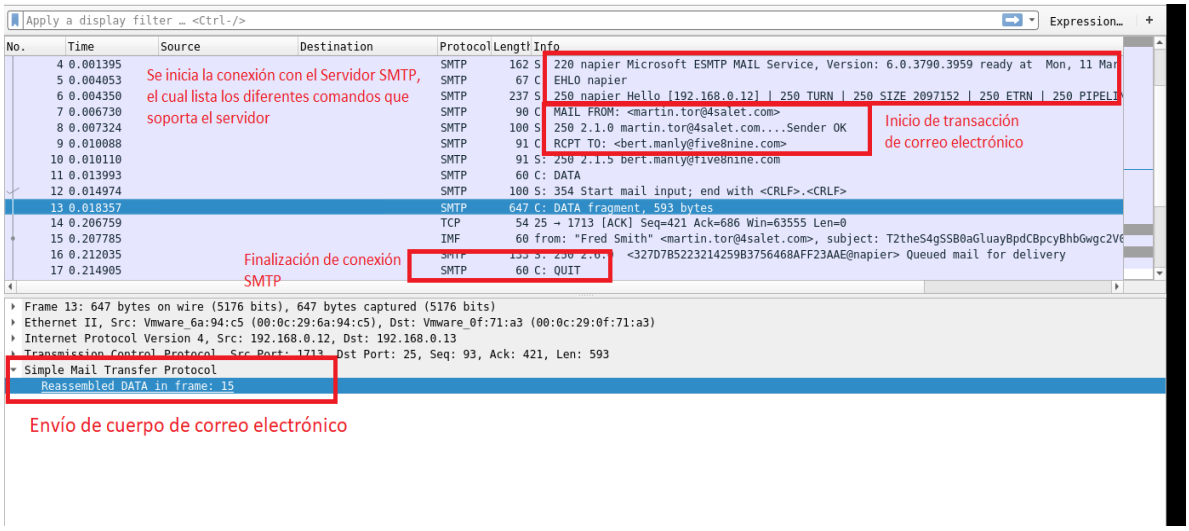
En una implementación normal de *SMTP* las credenciales de usuario suelen ser un usuario predefinido, que normalmente es el correo electrónico de quien accede al Servicio *SMTP* y una contraseña. En otras palabras, el método de acceso estaría basado en contraseñas.

c. Ejemplo de Comunicación SMTP

En la Figura 16 se puede observar una comunicación simple para el envío de un mensaje. En ella se observan los pasos de autenticación contra el Servidor *SMTP,* el inicio de la transacción de correo electrónico, un fragmento de uno de los paquetes enviados por correo electrónico y finalmente la finalización de sesión *SMTP.*

[89] GOOGLE y PING IDENTITY. OAuth 2.0 for Native Apps [online]. Internet Engineering Task Force IETF oct, 2017. [citado 22., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc8252 >

Figura 16 Ejemplo de Comunicación Simple de SMTP



Fuente: El Autor.

Como se puede observar se usa el comando *EHLO,* lo cual indica se está soportando una especificación de *SMTP* reciente. Se envía un correo desde la dirección *martin.tor@4salet.com* a *bert.manly@five8nine.com.* Luego de ello se envía otro correo desde la misma dirección de *martin,* el cual fue encolado probablemente para su envio posterior.

d. Amenazas Comunes

Debido a su antigüedad *SMTP* ha sido víctima de diferentes ataques y amenazas durante su historia. El hecho de proveer información bastante importante para las organizaciones y usuarios finales, lo convierte en uno de los objetivos preferidos durante el desarrollo de un ataque informático. Muchas son las amenazas que se han visto desarrolladas sobre diferentes proveedores de Servicios *SMTP,* a continuación, se establecen las más importantes[90].

- SPAM

[90] MICROSOFT. SMTP Security [online]. Microsoft mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://msdn.microsoft.com/en-us/library/cc505927.aspx >

Correo basura o indeseado, el cual es precisamente enviado por medio de Servidores *SMTP* que vienen configurados por defecto como *Open Relay,* es decir; que no hacen ninguna verificación sobre los usuarios que inician las transacciones de tipo *MAIL.* El *SPAM* se convierte en uno de los riesgos más grandes del Servicio *SMTP* porque otorga la capacidad de envío de correos fraudulentos, *scam* y *malware*[90]*.*

- Consecuencias

Se tienen las consecuencias del lado del servidor *SMTP* comprometido y de los receptores de los correos que hayan sido enviados. Por parte del servidor *SMTP,* el dominio adquiere baja reputación ante entes de control. Por parte de los receptores puede inducirse a una infección, fraude y engaño masivo.

- ¿Cuándo se es Vulnerable?

Cuando se utilizan configuraciones por defecto del Servidor *SMTP,* en especial cuando se dejan las opciones de *Relaying* u *Open Relay* habilitadas por defecto. Cuando el Servidor *SMTP* o uno de sus usuarios ha sido vulnerados.

- Ejemplos

Un atacante ha identificado por medio de un *scanner* de *SMTP,* que existe un Servidor que puede enviar correo electrónico sin necesidad de estar autenticado. El atacante, quien ya tiene direcciones de correo electrónico para distribuir su campaña de *SPAM,* usa esta debilidad para enviar los correos por medio del Servidor *SMTP.*

Un empleado de cierta organización ha caído bajo el engaño de una campaña de *phishing* donde ha compartido sus credenciales de acceso a correo electrónico, sin percatarse de ello y por comodidad; no cambia regularmente su contraseña de correo lo cual permite al autor de la campaña de *phishing* usar su correo electrónico para la distribución de campañas de *SPAM.*

- Captura de Credenciales, *Eaveasdropping* e Inyección

SMTP al ser tan antiguo, no soportaba aún las capacidades actuales de cifrado de comunicaciones. Esto deja vulnerables a muchos de los paquetes que se transfieren sobre canales inseguros, para que pueda observarse las credenciales de acceso de correo de los usuarios, sus transacciones de envío de correos y cualquier otra información relacionada a la comunicación *SMTP*[91].

- Consecuencias

Cualquier atacante que haya realizado con éxito el vector de ataque *MitM,* puede capturar las credenciales compartidas en los comandos *AUTH.* Además de ello puede modificar las transacciones de correo electrónico, cambiando los parámetros de los comandos *MAIL.* También es posible que pueda filtrase información relacionada a los correos electrónicos. En caso de robar credenciales de usuario, se puede incurrir en la suplantación de identidad.

- ¿Cuándo se es Vulnerable?

Cuando los métodos de autenticación no soportan ningún tipo de cifrado, o en su defecto utilizan métodos de cifrado débil. Cuando el canal de comunicación con el Servidor *SMTP* no ofrece mecanismos como *TLS/SSL,* es decir, al no usar el comando *STARTTLS.* Cuando la comunicación se hace a través de canales inseguros.

- Ejemplos

Un usuario malintencionado ha identificado que; en su red privada de la organización, la comunicación hacia el servidor de correo electrónico se hace mediante el uso de *SMTP* plano. Por ello, para filtrar algún tipo de información de algunos usuarios decide llevar a cabo un *MitM* por medio de *ARP Spoofin.* De esta manera captura todo el tráfico *SMTP* con un *sniffer* para posteriormente extraer los datos de los *Payloads* para los comandos *DATA.*

Un usuario malintencionado ha identificado por medio de *MitM* y la recolección de datos, que cierta comunicación por medio de *SMTP* utiliza un método de

[91] MICROSOFT. Op. cit., 112 p.

autenticación en *Base64.* Usando un decodificador de *Base64,* obtiene en texto plano las credenciales de acceso de algunos usuarios estando en la capacidad de autenticarse en el Servidor *SMTP* suplantando la identidad del usuario vulnerado.

- Denegación del Servicio

SMTP es un protocolo que tradicionalmente se está ejecutando bajo *TCP.* Lo cual lo hace especialmente vulnerable a los tipos de ataque *TCP SYN Flood* y *TCP RST.* De igual manera algunas implementaciones del servicio pueden ser vulnerables a *exploiting* lo que lleve a *crashear* el proceso que se encarga del servicio[91].

- Consecuencias

Pérdida de la comunicación por medio de correo electrónico para una organización. Pérdida de correos electrónicos importantes.

- ¿Cuándo se es Vulnerable?

Cuando no se utilizan mecanismos de control necesarios para filtrar los paquetes de tipo *TCP SYN,* del mismo modo cuando no se aplican actualizaciones sobre el *software* que provee el Servicio *SMTP.*

- Ejemplos

Como puede suceder con muchos de los Servicios estudiados anteriormente, cualquier atacante podría automatizar un *script* que envíe *TCP* de tipo *SYN* a la víctima. Si ésta no se encuentra debidamente protegida contra este tipo de ataques, el servicio puede denegar conexiones legítimas.

Un atacante por medio de *TCP Spoofin* puede enviar tanto a cliente como servidor un paquete de tipo *RST,* teniendo en cuenta los puertos del Servicio *SMTP.* Esto causaría que la conexión se reiniciara inmediatamente, evitando que los clientes puedan conectarse al servicio específico.

En ocasiones *SMTP* puede ser usado para transferir información confidencial por medio del uso de claves criptográficas, para así impedir que se pueda identificar la información filtrada de una organización. Sin embargo, no se lista dentro de las amenazas teniendo en cuenta que el hecho de que se observe la cabecera de los

mensajes, hace que este medio sea uno de los menos usados en la filtración de información. Otros métodos más efectivos pueden ser el uso de *VPNs* y *Tunneling* como se estudiaron en las secciones de *SSH* y *Virtual Private Network.*

5.2 MARCO CONCEPTUAL

5.2.1 Definiciones Conceptuales

a. Redes de Datos

Se trata de un conjunto de equipos o dispositivos conectados por medio de diferentes enlaces como cables, señales, ondas, entre otras. La disposición de las redes de datos es transferir información de un equipo a otro, con el fin de intercomunicar procesos o diferentes tareas que requieran datos de otros dispositivos para llevar a cabo sus objetivos[92].

b. Sistema Operativo

Conjunto de programas, aplicaciones y diferentes tipos de *software* que proveen la interfaz básica entre la comunicación del usuario final con el *hardware* de la computadora. Básicamente otorga un funcionamiento a los diferentes dispositivos informáticos. Los servicios que ofrece corresponden tanto a la coordinación de medios de *hardware* como a la interacción entre los diferentes procesos que puedan ejecutarse en el mismo y la seguridad proveída para los diferentes usuarios. Todo Sistema Operativo debe cumplir con las siguientes características básicas[93]:

- Conveniencia: Hace útil el uso de la computadora o dispositivo.
- Eficiencia*:* Permite que los recursos de una máquina sean usados de la mejor manera posible.
- Encargado de administrar el hardware: Destinado a manejar los diferentes recursos computacionales del dispositivo.

[92] TECHOPEDIA. Computer Network Definition [online]. Techopedia mar, 2018 [citado 22 mar., 2018]. Disponible en Internet: < https://www.techopedia.com/definition/25597/computer-network >

[93] WIKIPEDIA. Operating System [online]. Wikipedia mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://en.wikipedia.org/wiki/Operating_system >

- Organizar la información: Manejar los datos en los diferentes tipos de memoria que puede tener un dispositivo. Las memorias pueden ser volátiles o permanentes, en general, refiriéndose a memoria RAM y Discos Duros respectivamente.

- Administrar las comunicaciones en red: Gestionar la interconexión con otros dispositivos.

- Hacer frente a los Fallos de Usuario y Fallos de Hardware: Dar un oportuno tratamiento a los fallos generados por los usuarios y aquellos que son generados por problemas físicos del dispositivo o problemas de *hardware.*

- Relacionar dispositivos: Comunicar los diferentes periféricos que están condensados en los circuitos de un dispositivo. Permitiendo así que los procesos que utilicen varios de ellos, puedan ser compartir información de manera oportuna.

- Portable: Una de las ideas principales de un Sistema Operativo es que pueda ser reutilizado por varios dispositivos, otorgando a las aplicaciones y procesos del sistema la capacidad de desentenderse de restricciones específicas de tipo físico.

En la Actualidad los Sistemas Operativos más conocidos son: Windows, Basados en el Kernel de Linux, Android, iOS, FreeBSD y MacOS.

c. Kernel o Núcleo

Porción del Sistema Operativo ligada explícitamente a las funcionalidades necesarias para el funcionamiento del mismo. Es la parte más importante del Sistema Operativo, y otorga la interfaz de abstracción entre el *software* y el *hardware.* También está encargado de aspectos básicos de seguridad, aislamiento, administración de red y memoria. El código ejecutado por el *Kernel* es conocido como contexto privilegiado o *superadministrador,* en ocasiones se refiere al mismo como *Kernel Space* o *Kernel Mode,* dependiendo del Sistema Operativo[94].

[94] LINFO. Kernel Definition [online]. Linux Information Project mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < http://www.linfo.org/kernel.html >

d. Kernel Space o User Space

En los Sistemas Operativos basados en *Linux,* la memoria es dividida en dos regiones importantes; el espacio del *Kernel* o *Kernel Space* y el espacio del usuario *User Space.* El espacio del *Kernel,* es utilizado por el núcleo para acceder a los diferentes procesos ejecutados con contexto de *superusuario,* necesarios para el acceso a los diferentes periféricos del dispositivo; en él básicamente se encuentran todas las llamadas al sistema conocidas como *syscalls* y pueden ser accedidas por el usuario teniendo en cuenta ciertas restricciones de seguridad. Por otro lado, el espacio del usuario, es únicamente usado por las aplicaciones que son ejecutadas por el usuario final, teniendo en cuenta que muchos de los componentes en memoria no requieren exclusivamente de acceso a periféricos y tampoco tienen por qué ser ejecutados como *superusuario*[95]*.*

Estos conceptos son muy importantes debido que muchos tipos de *Malware* intentan ejecutar instrucciones maliciosas que les permitan acceder al espacio del *Kernel,* con el fin de aplicar escaladas de privilegios. Muchos de los servicios de red que serán estudiados, pueden interactuar con el *Kernel Space.*

e. User Mode y Kernel Mode

En los Sistemas Operativos *Windows,* existen dos modos básicos: el modo de usuario *User Mode* y el modo del *Kernel, Kernel Mode.* Estos modos, son una especie de metáfora del *User Space* y *Kernel Space* para los Sistemas Operativos basados en *Linux.* Su funcionamiento es bastante similar, aunque en *Windows* no se refiera directamente a la administración de espacios en memoria usados por ciertos procesos. Básicamente, cuando una aplicación está siendo ejecutada por cualquier usuario sin necesidad de privilegios; se ejecutará como *User Mode.* Una aplicación o proceso puede acceder al *Kernel Mode* siempre y cuando cuente con los permisos y privilegios necesarios.

Igualmente, estos conceptos son bastante importantes debido que muchos tipos de *Malware* intentarán aprovecharse del *Kernel Mode* para ejecutar código privilegiado en el dispositivo. Muchos de los servicios de red que serán estudiados, pueden interactuar con el *Kernel Mode* de *Windows*[96]*.*

---

[95] LINFO. Kernel Space Definition [online]. Linux Information Project mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < http://www.linfo.org/kernel_space.html >

[96] MICROSOFT. User mode and Kernel mode [online]. Microsoft abr, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode >

f. Firmware

El *firmware* ofrece una capa de abstracción de muy bajo nivel la cual se comunica directamente con el *hardware,* por esta razón se encuentra directamente ligado a los recursos físicos. La diferencia básica entre un *Firmware* y un Sistema Operativo, es que el *Firmware* no puede ser portable y está diseñado específicamente para un tipo de dispositivo[97].

g. Malware

En español definido como, programa malicioso o código maligno. Es un tipo de software utilizado para llevar a cabo acciones hostiles en una máquina, normalmente conocida como víctima. Puede ser tipificado según su funcionamiento o intención. En la actualidad los tipos más conocidos de malware son[98].

- Backdoors: Abren una puerta trasera en la víctima, permitiendo el acceso a usuarios no autorizados a máquina.
- Spyware: Espían las diferentes acciones del usuario con el objeto de extraer información sensible del mismo.
- Troyanos: Se camuflan en aplicaciones aparentemente legítimas para llevar a cabo acciones maliciosas.
- Gusanos: Tienen la capacidad de replicarse a sí mismos sin la interacción o autorización del usuario final.
- Virus: Infectan de manera intencional los espacios en memoria de otros procesos legítimos, con el objeto de ejecutar código malicioso en el contexto del proceso infectado.
- Exploit: Aprovechan huecos de seguridad de aplicaciones específicas para obtener comportamientos inesperados en la ejecución y flujo de la aplicación detectada.

[97] TECHOPEDIA. Firmware Definition [online]. Techopedia mar, 2018 [citado 22 mar., 2018]. Disponible en Internet: < https://www.techopedia.com/definition/2137/firmware >

[98] RIVERO, Marcelo. ¿Qué son los Malwares? [online]. Info Spyware oct, 2017. [citado 22., 2018]. Disponible en Internet: < https://www.infospyware.com/articulos/que-son-los-malwares/ >

- Malversiting: Inundan a la víctima de publicidad engañosa y poco deseada con el objeto de enriquecer al autor.

- Ransomware: Cifran la información privada de la víctima con el objetivo de solicitar un pago por su recuperación. Básicamente, secuestran la información de la víctima.

Algunas de estas amenazas serán profundizadas en la sección de Marco Teórico, otorgando una explicación y relación frente a las HoneyPots.

h. Zero Day y Patient Zero

Una amenaza de tipo Zero Day es aquella que utiliza técnicas de explotación o código malicioso desconocido, dirigiéndose a una vulnerabilidad que no había sido reportada con anterioridad. Es decir, una amenaza de la que no se tenía conocido, identificada en un escenario real. En español significa Amenaza de Día Zero. Por otra parte, un Zero Pacient o Paciente Zero es el primer sistema informático que se considera afectado por una amenaza de tipo Zero Day, también es conocido como Patient Z y es muy importante durante el Análisis Forense de amenazas desconocidas[99].

i. Protocolo de Comunicación de Red

Establece un conjunto de reglas que se deben cumplir para el intercambio de información entre diferentes dispositivos para una Red de Datos. Un protocolo tiene un correcto funcionamiento cuando se realiza la transferencia de información mediante estructuras específicas. Los protocolos de red pueden ser encapsulados según el Modelo OSI o Modelo TCP/IP permitiendo que diferentes secciones o estructuras de los paquetes transferidos, sean interpretados por componentes diferentes de los Hosts de origen y destino[100].

[99] ESET. ¿Qué es un 0-day? Explicando términos de seguridad [online]. We live Security feb, 2015. [citado 22 mar., 2018]. Disponible en Internet: < https://www.welivesecurity.com/la-es/2015/02/25/que-es-un-0-day/ >

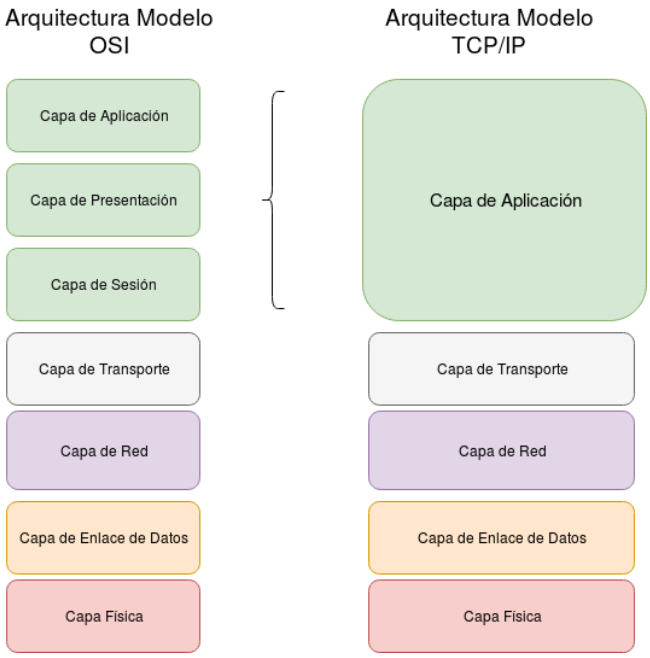
[100] International Organitation for Standardization ISO. Op. cit., 19 p.

j. Modelo OSI y el Modelo TCP/IP

El Modelo de Interconexión de Sistemas Abiertos, detallado en la ISO/IEC 7498-1 el cual es más reconocido como Modelo OSI, se trata de una arquitectura a grandes rasgos que establece la manera en que se debe transferir la información a través de Internet. Dicho modelo divide el proceso de transferencia de información entre dos dispositivos en 7 capas donde en cada una de ellas se representa una pila de protocolos de red. Una pila de protocolos de red, es un conjunto de protocolos que trabajan para un fin común[100].

Por otra parte, el Modelo TCP/IP es una descripción más breve que el Modelo OSI, aunque comparten básicamente muchos de los mismos conceptos. Su diferencia principal, es que el Modelo TCP/IP simplifica el número de capas del Modelo OSI dejando únicamente 5 capas. La Figura 17 muestra las capas del Modelo OSI junto con las capas expuestas del Modelo TCP/IP.

Figura 17 Estructura General Modelo OSI y comparación con Modelo TCP/IP



Fuente: El Autor.

A continuación, se explican las diferentes capas del Modelo OSI. Con ellas se dan por entendido el funcionamiento de las capas correspondientes al Modelo TCP/IP[100].

- Capa de Aplicación: Ofrece los servicios necesarios para establecer la comunicación de las aplicaciones ejecutadas por el usuario, tales como: Accesos a Correos Electrónicos, Accesos a Páginas Web, Resolución de Nombres, etc.

- Capa de Presentación: Se encarga de proveer la traducción necesaria de la información compartida en los paquetes de red, para que así puedan ser representadas en la aplicación correspondiente para la Capa de Aplicación. Por ejemplo: cuando se accede a una página Web la página es transferida mediante HTTP el cual dependiendo de ciertos atributos y estructura puede representar diferentes secciones de la página o portal Web, antes de mostrarse la información al usuario es necesario interpretar estos datos. Esta interpretación se hace en la capa de Presentación.

- Capa de Sesión: Tiene la responsabilidad de establecer la comunicación sobre las sesiones de diferentes dispositivos, conservando el flujo secuencial de los paquetes transferidos en red para que así puedan tener un orden lógico al momento de ser interpretados por la Capa de Presentación. Esta capa está encargada de mantener un dialogo durante la comunicación entre dos dispositivos.

- Capa de Transporte: Define la secuencia que debe interpretar la Capa de Sesión para determinar la conservación del dialogo entre dos dispositivos. Así mismo, define diferentes mecanismos de seguridad sobre la integridad de los paquetes entregados.

- Capa de Red: Determina la ruta a través de diferentes redes, que debe seguir un paquete para ser entregado. Esta capa es especialmente importante cuando la transferencia de información se hace a través de dos redes o subredes diferentes. En ella se implementan diferentes mecanismos de seguridad de bajo nivel.

- Capa de Enlace de Datos: Esta capa corresponde específicamente al tipo de arquitectura en bajo nivel, que se está usando para la transferencia de los datos. Por ejemplo: Ethernet, 802.11x, Celular, etc. Los datos aquí transmitidos corresponden específicamente a atributos determinados por los atributos de las interfaces de red.

- Capa Física: En esta capa se transfieren únicamente señales, pulsaciones, secuencias de luz, o cualquier medio específico donde se puedan transferir bits. La Capa Física corresponde al medio en que se desarrolla la comunicación de manera física, por ejemplo: Antenas Telefónicas, Cables UTP, Antenas Radiales, etc.

Los protocolos que serán objeto de estudio en este proyecto, pertenecen a la Capa de Aplicación, sin embargo; es necesario tener en cuenta los demás protocolos y capas que interactúan con estos para comprender la naturaleza de los protocolos.

k. Virtualización

Se trata de una técnica por la cual se representa una máquina por medio de la utilización de software, en lugar de hacerlo por medio de recursos físicos, disponiendo de una única máquina física. Es decir, disponer un único dispositivo el cuál por medio de tecnologías de virtualización; puede simular que contiene una o más máquinas. La virtualización provee diferentes capacidades tales como[101].

- Aislamiento: Impide que los procesos sobre los cuales ejecutan las máquinas virtuales, puedan generar fallas sobre los procesos de la máquina real.
- Independencia del Hardware: La Máquina Virtual, al trabajar sobre componentes virtualizados ofrece una independencia total del hardware.
- Encapsulamiento: La Máquina Virtual puede ser transferida entre diferentes equipos siempre y cuando se garantice el uso de un mismo hipervisor el cual es un programa encargado para el manejo de la virtualización.
- Consolidación: Permite dividir los atributos de una máquina física, o sus recursos físicos, de manera que estos puedan ser aprovechados al máximo.

El uso de la virtualización puede ayudar a que las amenazas estudiadas no comprometan el funcionamiento físico de la máquina, además pueden otorgar la capacidad de tomar snapshots para asegurar un restablecimiento del servicio más rápido.

l. Contenedores de Procesos

Los contenedores de procesos son entornos que ofrecen un total aislamiento de diferentes recursos como la CPU, la memoria, bloqueos de I/O y redes. Éstos ayudan a agregar algunas de las capacidades de las máquinas virtuales, sin embargo, no es necesario agregar una capa completa de abstracción sobre los

[101] REDHAT. El concepto de la virtualización [online]. Red Hat oct, 2016. [citado 22 mar., 2018]. Disponible en Internet: < https://www.redhat.com/es/topics/virtualization >

controladores y tampoco se requiere inicializar un Sistema Operativo completo. Puesto que no requieren un Sistema Operativo adicional, ofrecen un mayor rendimiento al minimizar el porcentaje de pérdida en la utilización de recursos de los Sistemas Operativos[102]. Algunos contenedores de procesos conocidos son: Docker, FreeBSD Jails, Solaris Zones, entre otros.

El uso de contenedores de procesos puede ayudar a que las amenazas estudiadas no comprometan el funcionamiento físico de la máquina. También por medio de sus capacidades de snapshots puede asegurar el restablecimiento más acelerado del servicio.

m. Sistemas de Prevención y Detección de Intrusos IDS e IPS

Los Sistemas de Prevención y Detección de Intrusos IDS e IPS son componentes destinados a la seguridad de redes y hosts que básicamente permiten ejecutar tres acciones principales[103].

- Monitoreo: Vigilar el tráfico para identificar la posible existencia de paquetes o datos maliciosos, sospechosos o incluso fallas de red. El tráfico puede ser archivo o puede analizarse de manera activa. Cuando es de manera local, vigilan las acciones que se realizan en el host.

- Detección: Analizar los datos recogidos en la tarea anterior, según sea por algoritmos de aprendizaje o aplicando normas definidas por el Administrador de Red. Se busca posibles paquetes o tramas que correspondan a comportamientos inusuales en la red, igualmente se buscan acciones sospechosas que puedan indicar la presencia de malware en un host.

- Respuesta ante Eventos: Responder de manera oportuna ante los eventos generados durante la fase de Detección. Es decir, si se encuentra algún tráfico o acción sospechosa ejecutar una acción específica.

Su diferencia principal se encuentra en la manera de dar respuesta a los eventos. Como indica su nombre, los IDS únicamente avisan de la presencia de una

[102] FABRIC8. Process Container [online]. Fabric8 oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://fabric8.io/gitbook/processContainer.html >

[103] SHARIFI, Ahmad; NOOROLLAHI, Akram y FAROKHMANESH Farnoosh. Intrusion Detection and Prevention Systems (IDPS) and Security Issues [online]. International Journal of Computer Science and Network Security nov, 14. [citado 22 mar., 2018]. Vol 14, no. 11. Disponible en Internet: < https://pdfs.semanticscholar.org/04da/5558cc056746a2edba2a7173ebb7c4d67807.pdf >

amenaza mientras que los IPS llevan a cabo una acción que puede neutralizar la manifestación de la amenaza completamente.

Los IDS e IPS pueden estar clasificados según su alcance básicamente, los dos tupos existentes son[103].

- Hosted Based (HIDS o HIPS): El elemento es instalado en un equipo donde se monitorean todas las acciones que se llevan a cabo en el mismo. De esta manera se identifican movimiento de archivos, escritura no autorizada de archivos, acciones sospechosas, entre otras. Por su naturaleza son llamados reactivos ya que únicamente tienen capacidad de acción en un host.

- Network-based (NIDS o NIPS): El elemento es instalado en puntos estratégicos de la red como Gateways o Proxy's donde se puede monitorear todo el tráfico de una porción de la red. Son más sofisticados que el tipo anterior ya que su estructura cuenta con capacidades de procesamiento de información en tiempo real.

Durante el desarrollo del proyecto, se usarán para identificar si alguna de las amenazas estudiadas pudo sobrepasar las barreras de las HoneyPots. Además, pueden arrojar otros datos estadísticos que puedan ser de interés.

n. Auditoría Informática

Proceso mediante el cual se verifica la capacidad de los mecanismos de control de un Sistema de Información para proteger los activos y la información misma de una organización. Busca encontrar las vulnerabilidades, amenazas y exposición de riesgos específico [104]. En ella se evalúan los diferentes controles que existen para los pilares de la Seguridad de la Información: Confidencialidad, Disponibilidad, Integridad y No Repudio. Es importante tener en cuenta este concepto durante el desarrollo del proyecto ya que el manejo de HoneyPots puede ayudar a encontrar controles que ayuden a los resultados de una auditoría informática.

o. Criptografía Simétrica y Asimétrica

La criptografía es un concepto comúnmente incluido en el manejo de redes ya que puede prestar mecanismos de comprobación de integridad y promover la

[104] ESPAÑA, GOBIERNO DE LA REPÚBLICA. Op. cit., p. 19.

confidencialidad de la información. Ésta provee por medio de algoritmos complejos y matemáticos, la capacidad de proteger la información[105]. Debido a la naturaleza de las HoneyPots propuestas, en especial la de VPN, es necesario comprender un poco cuáles son los tipos de criptografía y sus posibles aplicaciones en los medios informáticos. Los tipos de criptografía más usados en las redes son.

- Simétrica: Establece que para cifrar o descifrar un valor específico es necesario definir una llave, conocida como llave simétrica. El problema de este tipo de cifrado es que, si se adivina o se consigue la llave, se pueden descifrar los mensajes que han sido salvaguardados con ella[105]. En redes se utiliza comúnmente para la autenticación sobre Wifi, como lo es WPA2, WPA y WEP.

- Asimétrica o de Llave Pública: Establece que para cifrar o descifrar un valor específico es necesario proveer dos llaves: una publica y una privada[105]. La llave pública es usada para cifrar el mensaje, la llave privada es usada para descifrarlo. De esta manera se puede solventar el riesgo del tipo anterior, ya que siempre y cuando no se comparta la llave privada; no se podrá descifrar un mensaje. Claro es, que esto no aplica en todos los casos, teniendo en cuenta algunos estudios recientes. Es usada ampliamente para servicios de red, en especial en los protocolos de transporte TLS y SSL.

5.2.2 Ataques de Red

La red es susceptible a un sinfín número de ataques, de hecho, se considera uno de los medios más apetecidos por los diferentes atacantes ya que por naturaleza permiten la compartición de información sobre canales comunes. Durante el desarrollo del proyecto se identificarán diferentes amenazas que atañen específicamente a ataques que se llevan a cabo en la red. En general los ataques de red pueden ser clasificados *Según su Origen* y *Según su Comportamiento.*

[105] MUÑOZ, Alfonso; AGUIRRE, Jorge. Cifrado de las comunicaciones digitales: De la cifra clásica al algoritmo RSA. En: Capítulo 1 Introducción. España: 0xword, 2011 p 13 – 15. ISBN 978-84-616-3124-7.

5.2.2.1 Ataques de Red Según su Origen

a. Internos

Son aquellos generados desde la red interna, es decir desde la propia red. Comúnmente son generados por otros usuarios de la misma red[106]. Debido a que provienen de las redes internas, pueden ser bastante peligrosos ya que no existen sistemas o mecanismos de control como Firewalls que puedan mitigarlos. Para identificar este tipo de ataques es recomendado instalar NIDS.

b. Externos

Son aquellos que pueden ser generados desde una red externa, por lo general Internet. Normalmente, debido al tamaño elevado de dispositivos en red; es generado por una herramienta automatizada[106]. Debido a que pertenecen a redes externas, requieren saltarse mecanismos de control como Firewalls, Proxy's inversos, entre otros. Para identificar este tipo de ataques también se recomienda instalar un NIDS o modificar las normas de los diferentes mecanismos de control de cara a la red externa.

5.2.2.2 Ataques de Red Según su Comportamiento

a. Pasivos

Compone el conjunto de ataques que únicamente tienen intención de permanecer a la escucha del tráfico que existe en la red. Es decir, simplemente están esperando a que se transfiera cierto tipo de información sensible o que se ejecute cierta acción. Los ataques pasivos, generalmente se transforman en activos cuando se consigue suficiente información que pueda comprometer a la víctima[107].

---

[106] MICROSOFT. Security Threats [online]. Microsoft oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://msdn.microsoft.com/en-us/library/cc723507.aspx >

[107] MICROSOFT. Common Types of Network Attacks [online]. Microsoft oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://technet.microsoft.com/en-us/library/cc959354.aspx >

b. Activos

Compone el conjunto de ataques que tienen una intención agresiva u ofensiva, con el fin de corromper, modificar o tergiversar los datos de un Sistema de Información. A diferencia de los Pasivos, pueden comprometer la integridad de los activos y entorpecer el funcionamiento de la red atacada[107].

5.2.2.3 Ataques más Destacados en la Actualidad

a. Eavesdroping

Corresponde al término en inglés de escuchar en secreto o husmear. Consiste en un tipo de ataque de red pasivo, donde se encuentra a la expectativa de información sensible compartida por alguno de los elementos de red involucrados en una comunicación[107]. Normalmente la información que se desea obtener está relacionada a contraseñas, nombres de usuario, documentos, llaves públicas/privadas, entre otros. Para llevar este tipo de ataques, puede ser necesario llevar otro ataque de tipo Man in the Middle.

b. Man in the Middle o El Hombre en el Medio

Es un ataque bastante conocido en las redes, el cual consiste en la interceptación del tráfico que existe entre dos dispositivos de comunicación. Cuando el tráfico es modificado se considera un MitM activo, cuando sólo se lee información se considera pasivo. Para establecerse en el medio de la comunicación es necesario aprovecharse de ciertos huecos en los protocolos de las capas de enlace de datos, o incluso acceder de manera física, aprovechándose de la manera en que se comunican los dos hosts[107]. Por ejemplo: una manera sencilla de realizar un MitM es simplemente manipular el cable de datos físico para redireccionar todo el tráfico que existe de un Gateway a un conjunto de hosts. Otra manera puede ser aprovechándose de las capacidades de potencia de cierta antena, para generar más ruido en una red con el fin de hacerse pasar por el Gateway de la misma. Existen varios tipos de Man in the Middle, los cuales son listados a continuación.

- ICMP Redirect: Utiliza mensajes de tipo ICMP Redirect para indicar un router que existe una mejor ruta entre dos redes. Esta nueva ruta, involucra el host del atacante. Ya que requiere de un router, este ataque sólo funciona cuando se trata de diferentes redes o subredes.

- ARP Spoofin: El protocolo ARP es comúnmente usado para asociar una dirección física o MAC a una dirección IP. Este ataque busca confundir a los hosts involucrados en una red, para que consideren que una dirección MAC corresponde a una IP legítima. Es decir, el atacante selecciona una dirección IP de una máquina víctima y la dirección IP del Gateway. Ahora el atacante procede a notificar al Gateway que la IP de la víctima está asociada a la MAC de su máquina, y a la víctima que la IP del Gateway está asociado a la MAC de su máquina. De esta manera se puede hacer entre el canal de comunicación de ambos.

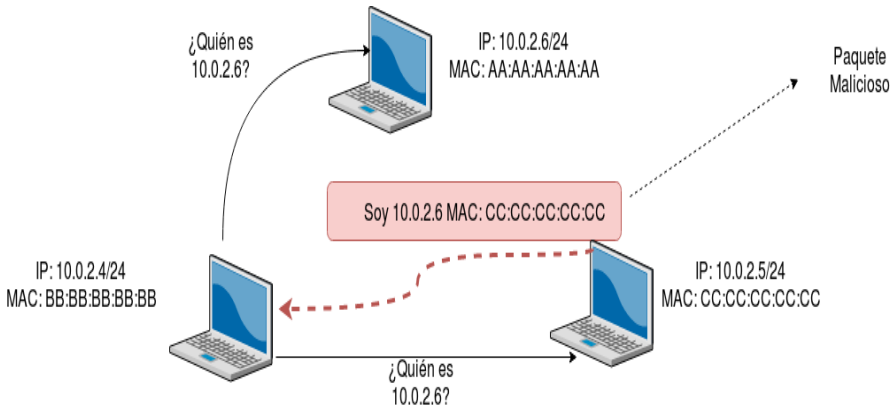
- DNS Spoofin: El protocolo DNS es aquel que se utiliza para la resolución de nombres, asociando una IP o conjunto de IP's a un dominio en específico. Este ataque consiste en suplantar una entrada DNS con la IP del atacante, de modo tal que cada vez que se realice una petición DNS a la entrada suplantada, la víctima se estaría conectando a la IP del atacante.

- Evil Twin o Rogue AP: Usando en su mayoría de veces en redes inalámbricas o Wifi, consiste en generar una copia falsa del Gateway que pueda ser tentadora para la víctima o, que teniendo en cuenta las capacidades y características de la señal, tenga mayor cobertura y potencia que un Gateway específico. El atacante es encargado de generar esta copia falsa de acceso a Internet.

- Identify Spoofin En las redes, como si se tratasen de personas, los diferentes dispositivos cuentan con identificadores únicos que permiten distinguirles de otros elementos de red. Igual que en un escenario real como la falsificación de documentos, también existen maneras de falsificar los identificadores de los dispositivos en una red[107]. De esta manera se puede hacer creer a la víctima que se está comunicando con quien desea, cuando en realidad está dejando a merced su información de un sujeto desconocido. Dependiendo de la capa del Modelo OSI o TCP/IP, se pueden llevar a cabo varios tipos de ataques Spoofin como.

- ARP Spoofin: Como se ha mencionado antes, intenta aprovechar las características del protocolo ARP para asociar una IP legítima a la MAC de un atacante. La Figura 18 muestra un ejemplo gráfico para este tipo de ataques.

Figura 18 Explicación gráfica de un Ataque ARP Spoofin

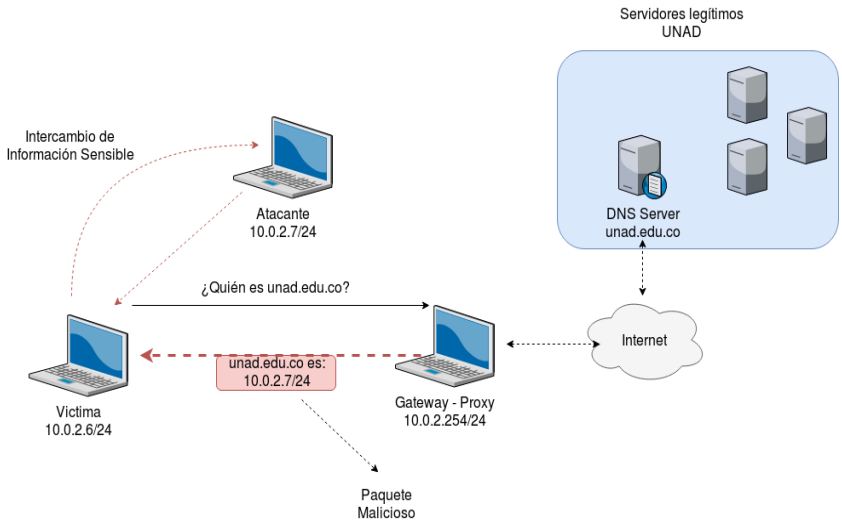


Fuente: El Autor.

Como se puede observar, se pregunta de a la dirección broadcast quien tiene una Dirección IP específica. Luego el Atacante, quien en este momento tiene la IP 10.0.2.5 dice que en realidad su MAC está asociada a la IP 10.0.2.6, lo cual es falso; pero puede engañar a quien ha lanzado la petición ARP.

- DNS Spoofin: Como se ha descrito antes, intenta reemplazar una entrada DNS legítima asociándola con una Dirección IP fraudulenta la cual, correspondería al atacante. La Figura 19 muestra de forma gráfica cómo se compone este tipo de ataque.

Figura 19 Explicación gráfica de un ataque DNS Spoofin



Fuente: El Autor.

Como se puede observar la víctima ha preguntado por el dominio unad.edu.co, el cual en un escenario normal debería viajar a través de Internet mediante los Authoritative DNS Servers y Root DNS Servers correspondientes hasta llegar a los servidores legítimos de la UNAD. Sin embargo, en este caso el Gateway quien también juega el rol de DNS Local, tiene una entrada maliciosa sobre este dominio y responde que de hecho la Dirección IP asociada a dicho nombre, se trata del atacante; causando que el tráfico de la víctima sea redirigido hacia el host del atacante.

- MAC Spoofin: Consiste en reemplazar una dirección MAC de un host, lo cual requiere de acceso local. En la mayoría de escenarios, tiene como objetivo saltarse controles impuestos por listas de acceso. Por ejemplo; en cierta red organizacional se impide la conexión a la red Wifi para aquellas MAC que sean de tipo Samsung, un MAC Spoofin sería reemplazar la dirección física de un portátil con tarjeta de red Samsung para saltarse este tipo de controles.

- IP Spoofin: Al igual que el MAC Spoofin tienen como objetivo cambiar el header del datagrama específico a la dirección IP origen para así poderse saltar puntos de control o simplemente confundir a un Servidor. En los Ataques de Denegación de Servicio DoS se una aplicación de ello en el ataque SMURF.

- TCP Spoofing: Busca suplantar el campo referido al puerto en adición a la Dirección IP, para enviar tráfico específico que pueda romper la secuencia de emisión y recepción de información por parte de dos dispositivos. Es usado especialmente con paquetes de tipo RST para causar Ataques de Denegación de Servicio DoS, reiniciando la secuencia de recepción y emisión de datos en dos dispositivos específicos.

Existen otros tipos de Spofin los cuales no están relacionados directamente con este proyecto, teniendo en cuenta las tecnologías de enlace de datos y capa física que serán usadas. En general, la idea del Spoofin es suplantar la identidad de un dispositivo en red, teniendo en cuenta el funcionamiento de un protocolo específico.

c. Password-Based Attacks

Muchos de los servicios en red, debido a su funcionamiento, requieren del uso de credenciales de usuario para acreditar que quien desea acceder al servicio, es en realidad un usuario del Sistema de Información[108]. Tal es el caso de muchos como:

[108] MICROSOFT. Op. cit., 128 p.

Kerberos, SSH, RDP, HTTP, entre otros. Como es de esperarse, estas credenciales de usuario pueden ser demasiado evidentes en ocasiones, o los mismos servicios, pueden no prestar mecanismos de protección específicos en cuanto a la tolerancia de intentos fallidos, listas negras, etc. Aún, teniendo mecanismos de canales cifrados, un servicio en red puede ser vulnerable a ciertos tipos de ataque que están configurados con la intención de intentar adivinar o predecir las credenciales de usuario. Algunos de estos ataques son.

- Ataques de Fuerza Bruta: Este tipo de ataques utiliza un conjunto de normas sintácticas para establecer una población donde teniendo en cuenta porcentajes de probabilidad, pueda encontrarse alguna de las credenciales de usuario usadas para acceder al servicio de red. Dependiendo de las reglas definidas y los recursos computacionales de la máquina del atacante, puede requerir un tiempo considerable.

- Ataques Dirigidos por Diccionario: Utiliza un conjunto de palabras conocidas como diccionario con el fin de establecer una población donde teniendo en cuenta porcentajes de probabilidad, puedan estar las credenciales de usuario usadas para acceder al servicio de red. Dependiendo de los diccionarios usados y los recursos computacionales de la máquina del atacante, puede requerir un tiempo considerable.

- Ataques en base al Tiempo de Respuesta: Puede utilizar cualquiera de los tipos anteriormente descritos, adicionando un análisis estadístico de la respuesta del servicio con el objeto de destacar aquellas credenciales de usuario con menor tiempo de respuesta. De esta manera se puede adivinar las credenciales de usuario usadas para acceder a un tipo de servicio.

d. Denial of Services DoS

Muchos de los ataques descritos anteriormente, buscan ganar acceso a un Servicio de Red. Sin embargo, a diferencia de ellos este tipo de ataque busca básicamente que ningún usuario pueda acceder a un Servicio de Red específico, de allí su nombre Denegación de Servicio[108]. Son altamente reconocidos ya que su impacto puede ser muy grave para una organización, y pueden incluir tácticas de daño permanente sobre el servicio afectado. Existen de diversos tipos, aunque en el proyecto sólo se destacan los siguientes.

- Inundación SYN (SYN Flood): Sobrecarga a un servidor específico de peticiones SYN del protocolo TCP los cuales tienen un origen suplantado falso. El Servidor intentará responder a dichas peticiones con SYN+ACK del

mismo protocolo, al quedarse a la espera de la respuesta del origen; la cual es falsa se reserva una porción de memoria para este objetivo. Si se realizan entonces muchas peticiones de tipo SYN, el servidor puede colapsar.

- Inundación ICMP (ICMP Flood): Intenta sobrecarga el ancho de banda de un Servidor mediante peticiones ICMP Echo las cuales tienen un Payload bastante grande, al intentar responder el Servidor con ICMP Echo Reply destina una cantidad proporcional al Payload enviado por el atacante. Si las capacidades del servidor son muy pocas, podrías colapsar.

- SMURF: Este ataque expande las capacidades del mencionado anteriormente. Enviando una petición al broadcast de la red, con un falso origen el cual apunta hacia el Servidor víctima. Todos los hosts de la red entonces responderán con el ICMP hacia el Servidor, causando que se colapse el mismo si no se tiene capacidad computacional necesaria para hacer frente a grandes cantidades de información.

- Inundación UDP (UDP Flood): Genera un alto tráfico de UDP contra la víctima, la cual de no tener suficientes capacidades computacionales; podría colapsar.

- Explotación de Bug: Poco conocido y difícil de explotar en un ambiente real, intenta aprovecharse de una vulnerabilidad del software que está ejecutando el Servicio de Red específico, de esta manera se puede generar un paquete con un Payload adecuado para cuasar el rompimiento del proceso correspondiente al servicio.

Cuando este tipo de ataques es llevado a cabo por una red hosts, se les conoce como Ataques de Denegación de Servicio Distribuidos DDoS.

e. Compromised-Key Attack and Downgrading Protocol

Es el conjunto de ataques destinados específicamente a comprometer la implementación criptográfica de los servicios y protocolos de red. Estos ataques buscan comprometer el funcionamiento de los algoritmos criptográficos, llevándolos a revelar características primordiales de los mismos o simplemente a disminuir su versión de aplicación. Este tipo de amenazas saca gran provecho del funcionamiento de muchos de los protocolos actuales, aunque es un poco complicado llevarlos a cabo[109]. En una comunicación cifrada, por ejemplo, es usual que Servidor y Cliente se pongan de acuerdo en los algoritmos que se desean

[109] MICROSOFT. Op. cit., 128 p.

implementar; un atacante que en lo general ya debe de haber realizado un MitM puede básicamente confundir tanto al Servidor y el Cliente para llevar a cabo implementaciones de algoritmos criptográficos vulnerables, con el objetivo de descifrar la información. Otro ataque conocido, es intentar encontrar el momento específico en que dos dispositivos involucrados en una comunicación cifrada; comparten componentes claves como: llaves privadas o llaves públicas que sean débiles.

5.3 MARCO LEGAL

5.3.1 Ley 1273 de 2009[110]

Por medio de la cual se establecen las directivas de protección de Sistemas de Información y la protección de los datos. En esta ley se establece una serie de acciones que pueden ser interpretadas como delitos informáticos. Comportamientos como: Accesos no autorizados, filtración de información, suplantación, violación de datos personales; entre otros, se tratan dentro de esta ley. Esta ley aplica para la República de Colombia únicamente.

5.3.2 Ley Estatutaria 1266[111]

La cual establece y define cómo se debe administrar y captar la información personal del lado de las empresas. Son las disposiciones generales del *Habeas Data* para la regulación de Bases de Datos personales, incluyendo en especial los sectores financieros, crditicio, comercial y de servicios.

Se relaciona con el proyecto ya que es necesario recolectar información, la cual puede ser considerada como información personal. Se debe garantizar que el despliegue y configuración de los *HoneyPots* no colecte información considerada como personal, ya que se podría incurrir en la quebrantación de esta ley.

---

[110] CONGRESO DE LA REPÚBLICA DE COLOMBIA. Ley 1273 de 2009 Nivel Nacional [online]. Alcaldía de Bogotá ene, 2009. [citado 21 mar., 2018]. Disponible en Internet: < http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=34492 >

[111] COLOMBIA, CONGRESO DE LA REPUBLICA. Ley 1266. Bogotá. (Diciembre 31 de 2008). Diario Oficial 47.219 de diciembre 31 de 2008. p. 1-15.

5.3.3 Federal Wiretap Act y Electronic Communication Privacy Act[112]

Ambas constituyen dos leyes de Estados Unidos cuya posición es adoptada a nivel mundial. Trata sobre la relación de los datos privados y personales y cómo deberían ser administrados por los Sistemas de Información. El *Wiretap Act* establece básicamente que se debe considerar ilegal la captación de datos en tiempo real, a menos que la persona sea consciente que su información está siendo guardada para su seguimiento. Únicamente existe una excepción a esta ley es usar los datos para un fin o bien propio, excepción conocida como *Service Provider Protection.*

Por otro lado, la ley *Electronic Communicaction Privacy Act,* establece el manejo de las autorizaciones sobre la captación de información cableada, oral o que viaje por medio de medios electrónicos; definiendo qué tipo de comportamientos se pueden considerar como actos ilegales.

Los *HoneyPots* se relacionan con estas leyes ya que permiten recoger información de los usuarios, aunque podrían considerarse bajo una excepción debido que los datos recolectados serán u9sados para un bien necesario y a manera de defensa propia.

5.3.4 Reglamento General de la Protección de Datos GDPR[113]

Se trata de un conjunto de normas establecidas para las organizaciones que establecen la manera en que deben tratar los datos e información personal, así como los niveles de seguridad que deban tener. Se divide, en resumen, en una estructura de diferentes medidas (organizativa, técnica y a nivel de los datos). La directiva debe ser aplicada a todos los Sistemas de Información que se alojen en Europa. Este reglamento ha sido establecido por la Unión Europea.

Se relaciona con los *HoneyPots* ya que en algunas secciones refiere al tratamiento y niveles de seguridad que debe mantener un sistema cuya información pueda ser considerada como personal.

---

[112] U.S.A GOVERNMENT. 18 U.S. Code Chapter 119 – Wire and Electronic Communications Interception and Interception of Oral Communications [online]. Legal Information Institute, 2015. [citado 21 mar., 2018]. Disponible en Internet: < https://www.law.cornell.edu/uscode/text/18/part-I/chapter-119 >

[113] UNIÓN EUROPEA. Reglamento General de Protección de Datos [online]. ESET feb., 2018. [citado 21 mar., 2018]. Disponible en Internet: < https://gdpr.eset.es/ >

5.3.5 Legalidad de los HoneyPots en Colombia

Las *HoneyPots,* a pesar de ser usadas para un bien investigativo o de protección de activos, irónicamente pueden ofrecer cualidades y capacidades cuya finalidad se podría interpretar como maliciosas. El objetivo de estas soluciones informáticas es ofrecer una variedad de atributos que permitan recoger la información de un atacante en el momento justo donde se lleva un delito informático, sin embargo, precisamente este acto de monitoreo probablemente esté llevando al usuario o beneficiario de los sistemas de señuelo a cometer un delito. Esta interpretación es similar a cuando se observan en las noticias novedades acerca de un ladrón que ha sido víctima de un atroz maltrato por parte de quienes en un principio iban a ser robados. En Colombia es bien conocido el término de la *defensa legítima* y la manera en que puede ser usado por el mismo delincuente para salirse con las suyas. En todo esto surge una pregunta principal *¿Cómo considerar si una HoneyPot es legal o ilegal en Colombia?* En primer lugar, considerar la legalidad de uno de estos mecanismos hace necesario indagar en tres hitos importantes[114]

- Inducción al Delito: Se sabe que una *HoneyPot* es básicamente un sistema de engaño para cualquier usuario, teniendo en cuenta que sus únicos usuarios vendrían siendo los delincuentes. Sin embargo, es necesario preguntarse ¿Es acaso este sistema un engaño que va mucho más allá de fines investigativos o defensivos? ¿Quiere el autor inculpar a alguien más sobre un delito que el mismo ha inducido a realizar? Un sistema de señuelo instalado y configurado con el objeto de inculpar a alguien más podría fácilmente considerarse en un fraude o un *complot* contra el delincuente. En algunos gobiernos como en Estados Unidos se recomienda colocar a estos sistemas avisos específicos sobre la intención que tendrá el sistema de señuelo, aunque claramente esto reduciría su usabilidad y seguro impactaría de forma negativa a los estudios realizados.

- Privacidad: El objetivo principal de los sistemas de engaño es recoger información, conocer cómo opera el criminal y qué herramientas puede usar para lograrlo; para ello es necesario extraer información sobre sus actividades. Pero ¿Qué información está relacionada a la intimidad del atacante? Ciertamente es una pregunta difícil de contestar, lo cual da cierto grado de ambigüedad a los aspectos legales de una *HoneyPot.* Bien se podría desplegar un sistema que simplemente observe las acciones maliciosas del atacante, como lo hacen las *HoneyPots de baja interacción,* o también es posible que se quiera incluso identificar algunos atributos o propiedades en red del atacante generando un perfil informático, a esto se le

[114] ALVARES, Carlos. Aspectos penales relativos al uso de “Honeypots”. Legal Legis. Colombia: dic 2003. 20 p.

conoce como *HoneyPots de alta interacción.* Quizás la legalidad se tambalea un poco en este segundo tipo de *tracking* donde es posible que, al generar el perfil del delincuente, se esté obteniendo información de más; como números de teléfono, correos electrónicos, direcciones IP públicas y geolocalización. En este punto cualquier criminal podría ejercer un derecho a la intimidad y a proteger su información personal, argumentando que al acceder al sistema señuelo todos sus datos han sido vulnerados; este hecho podría agravarse si estos datos son colocados en público. Al igual que el punto anterior, sería recomendado colocar al menos un aviso donde se notifique al usuario final o delincuente que sus acciones van a ser monitoreadas en todo momento; aunque esto evidentemente incapacita a la *HoneyPot* de su propiedad de pasar desapercibida.

Por fortuna en la actualidad algunas soluciones de *HoneyPots* son conscientes de esto y presentan datos generales que no comprometen a los usuarios finales.

- Responsabilidad: ¿Qué sucedería si en realidad se está estudiando un atacante que ya ha tomado control sobre otra máquina? En redes de computadoras es muy común el tipo de ataques laterales, es decir, aquellos delincuentes que primero toman un servidor o dispositivo como base para desplegar a partir de allí sus actividades maliciosas. Podría llamársele un tipo extraño de *proxy,* lo cual finalmente dificultaría la identificación de la fuente de origen. En caso que la *HoneyPot* sea instalada con fines defensivos, esto puede tener un conflicto, sencillamente quienes estén encargados en la infraestructura de red para una organización podrían terminar contratacando a alguien que es ajeno al escenario delictivo. Por otro lado, suponiendo por ejemplo que la *HoneyPot* sea con fines investigativos; es posible que los investigadores estén incurriendo en la recolección de datos falsos que; aunque no tiene implicaciones legales, podría estropear un estudio completo. En este punto es recomendable preguntarse si vale la pena de fiarse en los datos y contratacar. Por fortuna, igualmente existen maneras de identificar cuándo un dispositivo está a cargo de un criminal que no derechos de propiedad sobre el mismo.

5.4 ANTECEDENTES Y ESTADO DEL ARTE

Muchos de los sistemas de control a día de hoy trabajan sobre firmas y comportamientos conocidos de *malware; Sistemas de Detección y Prevención de Intrusos (IDS-IPS)* y *Antivirus* en general actualizan una considerable cantidad de Bases de Datos con la información necesaria para la búsqueda de amenazas. El problema, básicamente, es que los mismos proveedores o desarrolladores que

mantienen el *software* deben conocer esta amenaza en específico para posteriormente agregarlas a sus Bases de Datos; esto impide y obstaculiza en muchos casos la investigación autónoma de las empresas, como se expone en *"Deception Techniques, Methods, Honeypots, Honeynets and Usage"*[115]. Frente a esta necesidad, bastante conocida en el siglo pasado, emergen las publicaciones "*The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*"[116] y "*An evening with Berferd In Wich a Cracker is Lured*" [117] a principios de los años noventa, fue en esta época donde puntualmente se conoció el término de *HoneyPot,* que en pocas palabras quiere decir *"Sistema de carnada o señuelo".* El concepto sería nuevamente abordado en *"The Honeynet Project"* en 1999 donde se exaltaría la importancia de las *HoneyPots* y las *HoneyNets* otorgando su relación con la *Seguridad Informática y de la Información.*

En el trabajo *"Deception Techniques, Methods, Honeypots, Honeynets and Usage"*[115] dan una breve exposición de simulación de servicios mediante Sistemas Operativos basados en *Unix,* para capturar las acciones de los atacantes y procurar identificar su localización; en primera instancia aplicando un concepto bastante básico sobre los *sistemas de señuelo*. Aunque, se parte de la premisa de un ataque desarrollado por un humano, algo contrastado al trabajo de *"Honeypots Detecion in Advanced Botnet Attacks"*[118] donde se determina la construcción de una *HoneyPot* para la detección de *botnets.* Estos dos tipos de trabajo, como base primordial nos exponen un Análisis específico para la detección de actividades maliciosas. Mientras un humano puede cometer errores que le delaten fácilmente, quizás una *botnet* más elaborada contiene maneras de evasión que pueden ser difícilmente detectadas. Diego Jurado, en su *"Trabajo de Fin de Grado: Análisis y Estudio de Honeypots Complejos: Honeynets"*[119] expone este problema, refiriéndose a la necesidad de evaluar el nivel de exposición de las *HoneyPots,* implementando *software* que no sea conocido por el atacante para prevenir su posible explotación. En estas tres investigaciones se observan algunos problemas e inconvenientes, ¿Hacia qué fin dirigir una *HoneyPot?* ¿Qué criterios permiten la implementación de una *HoneyPot* ser segura y confiable? y finalmente ¿Cuánto trabajo y costos puede significar mantener alguna?

---

[115] TANLI, Ilker; KOLCALAR, Turgut. Deception Techniques, Methods, Honeypots, Honeynets and Usage. MontClair. Inglaterra: jun, 2017. 10 p.

[116] STOLL, Cliff. Op. cit., p. 27.

[117] CHESWICK Bill. Op. cit., p. 27.

[118] WANG, Ping; WU, Lei; CUNNINGHAM, Ryan et al. Honeypot detection in advanced botnet attacks. International Journal of Computer Science and Network Security feb, 2010. Vol. 4, issue 1. P 30 – 51.

[119] JURADO PALLARÉS, Diego. Trabajo de Fin de Grado, Análisis y Estudio de Honeypots Complejos: Honeynets. Madrid: Universidad Autónoma de Madrid. Escuela Politécnica Superior. Departamento de Ingeniería Informática. 2016. 93 p.

En la práctica, existen diversos tipos de *sistemas de señuelo* disponibles para su uso libre gracias a su licenciamiento, un ejemplo es la publicación *"Building a Honeypot to Research"*[120] donde se implementa *Kippo* para el servicio *SSH*. Algunos otros tipos de soluciones son abordados en "*A Practical Guide to Honeypots*"[121] en el cual se pueden observar un listado breve de proyectos vigentes en la actualidad como *Honeyd, HoneyBOT* y *Specter*. Algunos de ellos aplicados por los trabajos de Ilker y Turgut[115]*.*

La mayoría de estos trabajos, recolectados hasta la fecha; muestran que a pesar de aplicarse el concepto base de una *HoneyPot* para realizar tácticas de *engaño,* sus conclusiones en la mayoría de los casos únicamente atañen sobre un tipo de servicio de red. Como se explica en la tesis *"Trabajo de Fin de Grado: Análisis y Estudio de Honeypots Complejos: Honeynets"*[119]; existen diferentes soluciones para la instalación de estos *sistemas de señuelo,* aunque no se hayan consolidados hasta la fecha. Las organizaciones no cuentan con un servicio específico, pueden exponer diferentes vulnerabilidades sobre distintos Servicios y Protocolos que puedan ser base para el desarrollo de vectores de ataques, por lo tanto, se hace necesario la consolidación de los Servicios comúnmente asociados a un entorno empresarial para su estudio bajo el término de *HoneyPot,* sin limitarse a comparar una situación deseada a una situación actual, sino que además otorgando una serie de pautas para lograr un proceso de *hardening* en los diferentes sistemas; de esta forma se logra aplicar un concepto más amplio de la investigación autónoma de amenazas para las compañías basados en los *sistemas de señuelo.* Finalmente, el ambiente adecuado para la realización de este proyecto es Internet; donde se encuentra en actual circulación todo tipo de actividad maliciosa.

Otro trabajo importante donde se alude el diseño de red seguro mediante el uso de implementaciones de *HoneyPots* es el de *Daniel Gustavo*[122]*.* Se resalta su importancia ya que es impartido por una universidad ecuatoriana, exaltando la investigación de las instituciones latinoamericanas en base a esta temática. Aquí se aborda un análisis por medio de *HoneyPots* que permite posteriormente mejorar el esquema de seguridad de la Universidad de Guayaquil.

---

[120] BELL, Simon. Building a HoneyPot to Research Cyber-Attack Techniques. Reino Unido: Faculty of Computer Systems and Software Engineering, University College of Engineering and technology mar, 2014. 69 p.

[121] PETER, Eric y SCHILLER, Tood. Op. cit., 41 p.

[122] ESTRELLA GUIJIJE, Gustavo. Diseño del Prototipo de una HoneyPot Virtual que permitirá mejorar el Esquema de Seguridad en Redes de la Carrera Ingeniería en Sistemas Computacionales y Networking de la Universidad de Guayaquil. Ecuador: Universidad de Guayaquil sept, 2011. 370 p.

6 CONGURACIÓN DE HONEYPOTS USANDO TÉCNICAS DE VIRTUALIZACIÓN

Cada *HoneyPot* se configura aprovechando las ventajas de la virtualización, por medio de un único Servidor expuesto el cual contiene los diferentes servicios y *software* implementado para el despliegue de los *HoneyPots.* En este capítulo se observa la descripción del ambiente virtualizado y la configuración de cada uno de los *softwares* configurados para simular el comportamiento de los servicios LDAP, HTTP, SSH y SMTP.

6.1 DESCRIPCIÓN DE AMBIENTE VIRTUALIZADO

La virtualización del entorno de despliegue de los *HoneyPots* se realizó por medio del uso de un *Virtual Private Server,* esta tecnología consta del uso de un único Servidor Físico el cual puede contener varias instancias de Máquinas Virtuales o Contenedores que puedan ser utilizados como servidores independientes y virtualizados. Cada una de estas instancias es administrada por medio de un *software* que en la mayoría de los casos se conoce como *Hipervisor.*

Durante el desarrollo del proyecto se opta por el uso del servicio de *VPS Hostinger*[123] el cual utiliza el concepto de virtualización y *Containerzation,* por medio del *software OpenVZ*[124]*.* La Figura 20 muestra las características del *VPS* adquirido, el cual en adelante podrá ser referido por el nombre *Prometeo*.

[123] HOSTINGER COLOMBIA. Términos y Condiciones de Uso del Servicio [online]. Hostinger mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://www.hostinger.co/terminos-uso >

[124] VIRTUOZZO CONTAINERS. OpenVZ Get Started and Instalation [online]. Virtuozoo Containers mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://openvz.org/Main_Page >

Figura 20 Características del Virtual Private Server de Hostinger

Fuente: El Autor.

Los términos de uso y condiciones de *Hostinger* no especifican incumplimiento sobre el manejo de *HoneyPots* a la fecha[123], por lo cual se concluye que se puede usar el servicio de manera legal.

La Tabla 2 describe otras características importantes a tener en cuenta en el *VPS.*

Tabla 2 Características adicionales del VPS

Característica	Valor
Sistema Operativo	Debian Jessie 8
Versión del Kernel	Compilación personalizada por el VPS Kernel 2.8
IP Reversa/Pública versión 4	185.201.8.66
IP Reversa/Pública versión 6	2a02:4780:3:3:ab2e:6b18:8fe4:4a38
Copias de Seguridad	No
Método de Administración	SSH

Fuente: El Autor.

Las siguientes consideraciones son usadas para la administración del *VPS,* para así evitar que el servidor sea comprometido completamente:

m. Se eligen claves fuertes para los diferentes usuarios del sistema.

n. Se deshabilita la autenticación por medio de contraseñas en el Servicio SSH, escogiendo la autenticación por medio de cifrado de clave pública.

o. El puerto de conexión de SSH se cambia por uno no estándar, el cual sólo lo conocen los usuarios del sistema.

p. Se aplica *Port Knocking* para que el puerto de SSH no esté expuesto completamente.

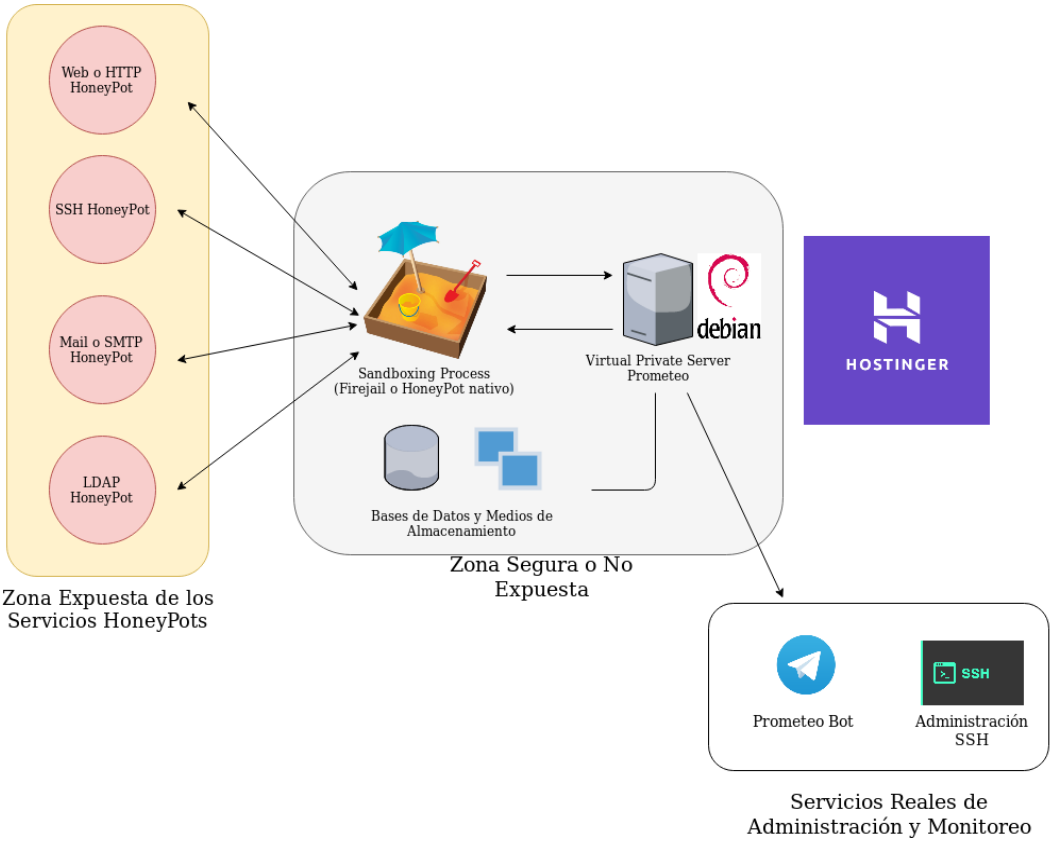
q. El servidor cuenta con un *bot* personalizado en *Telegram,* el cual sirve para monitorear el estado del servidor mismo y los *HoneyPots.* De esta manera se vigila constantemente el estado del servidor por medio de un canal rápido de mensajería instantánea.

No se entra a detalle sobre este tipo de configuraciones ya que se escapa del alcance del proyecto. El proveedor del *VPS,* aunque no permite copias de seguridad, permite el uso de *Snapshots* las cuales son configuradas con regularidad cada tanto se hace una nueva configuración sobre el servidor. Esto evita que durante el desarrollo del proyecto la integridad del servidor se vea comprometida completamente.

6.1.1 Esquema de Configuración de los HoneyPots

La Figura 21 muestra la estructura general propuesta para la configuración de los *HoneyPots.* En ella se puede diferenciar los componentes que serán expuestos y qué otros elementos se encargarán del funcionamiento del servidor *VPS,* su monitorización y recolección de la información.

Figura 21 Estructura general de la Configuración de los HoneyPots en el VPS.



Fuente: El Autor.

Como se puede apreciar, constantemente el *VPS* podrá ser administrado y monitoreado por medio de los servicios de *Prometeo Bot* el cual está integrado a *Telegram*[125] y la Administración de *SSH.* Los únicos servicios que serán expuestos son los cuatro *HoneyPots* correspondientes a los protocolos objetos de estudio. Los procesos de los servicios de los *HoneyPots* serán enjaulados en una *sandbox*[126], la cual durante la configuración de cada servicio se decidirá si es o no compartida. La *sandbox* tiene como objetivo mitigar el acceso de los usuarios malintencionados a los recursos reales de la máquina, algo que protegerá en primera instancia al *VPS.*

[125] TELEGRAM. Bots: And Introduction for developers [online]. Telegram mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://core.telegram.org/bots >

[126] MOZILLA. Sandbox Architecture of Firefox [online]. Mozilla mar, 2018. Security/Sandbox/Process model. [citado 22 mar., 2018]. Disponible en Internet: < https://wiki.mozilla.org/Security/Sandbox/Process_model >

6.2 CONFIGURACIÓN DE HONEYPOT WEB GLASTOPF (PROTOCOLO HTTP)

En los siguientes pasos se expone la forma en que es configurado el *HoneyPot Glastopf* para proveer un sistema de monitoreo almacenando la información en las Bases de Datos de *MongoDB.*

6.2.1 Instalación

a. En primer lugar, se deben instalar todas las dependencias de *Glastopf* esto se logra con los siguientes comandos en la terminal de *Debian*[127]*.*

```
apt-get update
apt-get install python python-openssl python-gevent libevent-dev python-dev build-essential make
apt-get install python-argparse python-chardet python-requests python-sqlalchemy python-lxml
apt-get install python-beautifulsoup mongodb python-pip python-dev python-setuptools
apt-get install g++ git php7 php7.0-dev liblapack-dev gfortran
apt-get install libxml2-dev libxslt-dev
apt-get install libmysqlclient-dev
pip install --upgrade distribute
```

Durante el desarrollo del proyecto se encontró la necesidad de migrar de *PHP 5.0* a *PHP 7.0,* una guía muy detallada de cómo hacerlo es ofrecida por *Nixcraft*[128]*.*

b. El siguiente paso es instalar un *PHP Sandbox* con el cual el software puede simular algunas funcionalidades de *Zend.* Para ello el mismo autor de la herramienta ha creado un repositorio conocido como *BFR (Better Function*

[127] MUSHORG. Glastopf Installation – Debian Squeeze [online]. Github mar., 2016. [citado 22 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/glastopf/blob/master/docs/source/installation/installation_debian.rst >

[128] NIXCRAFT. How to install PHP 7 on Debian Linux 8.7/7.x Jessie/Wheezy [online]. Cyberciti feb., 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://www.cyberciti.biz/faq/installing-php-7-on-debian-linux-8-jessie-wheezy-using-apt-get/ >

Replacer based on APD)[129] este repositorio debe ser clonado y compilado, una vez se haya actualizado el *PHP.* Para ello se utiliza los siguientes comandos[127].

```
cd /opt
git clone git://github.com/mushorg/BFR.git
cd BFR
phpize
./configure --enable-bfr
make &&  make install
```

Luego se debe añadir la siguiente línea al archivo de configuración *php.ini.*

```
zend_extension=/usr/lib/php/20151012/bfr.so
```

La ubicación de este archivo se puede conocer con el comando *php --ini.*

Nota: la ubicación de bfr.so puede variar dependiendo de la versión descargada de *BFR.*

Una vez se haya habilitado esta línea, se debe ejecutar *php --version* donde se puede observar en la salida del comando el siguiente renglón:

```
with Better Function Replacer (BFR) v0.1, Copyright (C) 2015,
by Lukas Rist
```

Si es así significa que la extensión *BFR* ha sido instalada correctamente.

c. Finalmente se debe instalar *Glastopf,* en este caso se realiza por medio del repositorio ya que la instalación por *pip* tiene problemas con *distribute.* Para ello se ejecutan las siguientes líneas de comandos en la terminal[127].

```
cd /opt
git clone https://github.com/mushorg/glastopf.git
git clone https://github.com/client9/libinjection.git
git clone https://github.com/mushorg/pylibinjection.git
cd glastopf
python setup.py install
```

[129] MUSHORG. Better Function Replacer base don APD [online]. Github oct., 2015. [citado 22 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/BFR >

6.2.2 Configuración de Sistema de Almacenamiento en MongoDB

a. Una vez instalado el *HoneyPot* se debe crear un directorio cualquiera en este caso se selecciona la ruta */opt/HoneyPots/WebHoney.* Allí se debe ejecutar el comando *glastopf-runner.* Una vez ejecutado el comando debe detenerse inmediatamente.

   Este comando lo único que hace por el momento es crear un directorio completo del *HoneyPot* donde el archivo más destacado, por ahora, será *glastopf.cfg.*

b. Se procede a iniciar el proceso de *MongoDB* con el comando *systemctl start mongodb,* una vez iniciado se procede a la creación de una Base de Datos destinada para *glastopf* con los siguientes comandos[130].

```
use glaspot
db.createUser({user:”glaspot”, pwd:”contra”,
roles:[“readWrite”, “dbAdmin”]})
```

c. Luego se procede a editar el archivo *glastopf.cfg* donde anteriormente se ejecutó el comando *glastopf-runner.* En dicho archivo se busca la sección *main-database* y se comenta el connection_string actual el cual usa un SQL Lite. Luego se coloca el siguiente.

```
connection_string =
mongodb://glaspot:contra@localhost:27017/glaspot
```

   Se recomienda bloquear el acceso externo a los puertos 27017, ya que esta base de datos debería ser accedida únicamente por el *HoneyPot.*

d. Una vez terminada y comprobada la configuración se recomienda habilitar el servicio de *MongoDB* con el comando *systemctl enable mongodb.*

6.2.3 Configuración de *Glastopf* como Servicio

a. Se puede configurar *Glastopf* como servicio haciendo uso de *systemd.* Esto trae las facilidades de integración con el resto del sistema *Debian* y el

[130] MONGODB. User Management Methods, db.createUser() [online]. MongoDB mar., 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.mongodb.com/manual/reference/method/db.createUser/ >

daemon principal del Sistema Operativo. Para ello se debe crear un archivo llamado *glastopf.service* el cual contenga la siguiente estructura.

```
[Unit]
Description=Glastopf HoneyPot
After=network.target
Conflicts=apache2.service httpd.service

[Service]
ExecStart=/usr/local/bin/glastopf-runner --workdir
/opt/HoneyPots/WebHoney

[Install]
WantedBy=multi-user.target
```

b. Una vez terminado el archivo se debe copiar a la ruta */etc/systemd/system/glastopf.service.* Luego con los comandos *systemctl start glastopf.service* y *systemctl status glastopf.service* se comprueba su estado.

c. Finalmente se recomienda habilitar el servicio para arrancar en conjunto con el Sistema Operativo, para ello se usa el comando *systemctl enable glastopf.service.*

Nota: El puerto de la *HoneyPot* es por defecto el 80, el mismo puerto que el servicio Web para *HTTP.* Si se desea modificar se debe cambiar la entrada *port* de *webserver* en el archivo *glastopf.cfg.*

6.2.4 Pruebas de Despliegue

a. Para hacer las pruebas se debe asegurar que tanto los servicios de *MongoDB* y *Glastopf* se encuentran en ejecución. Para ello se puede usar el comando *systemctl status.*

b. Una vez se garantice que los dos servicios se están ejecutando se procede a verificar el puerto, comprobando que se puede acceder al mismo. Para ello se usa el comando *nmap* y se abre la página por medio de un navegador *web.* La Figura 22 expone el resultado de la prueba con el navegador web.

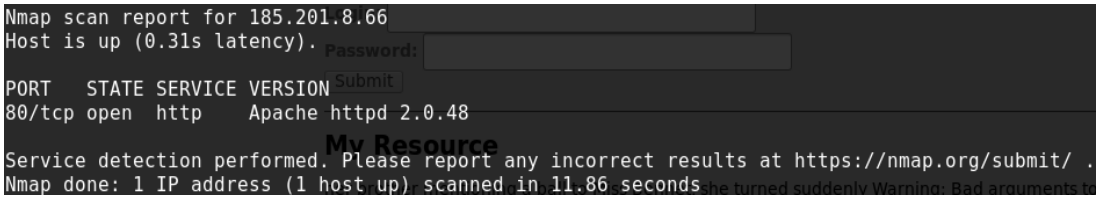
Figura 22 Prueba de HoneyPot Glastopf desde el Navegador Web

Fuente: El Autor.

Se ejecuta la siguiente instrucción con *nmap.* Los resultados se observan en la Figura 23.

```
nmap -sV 185.201.8.66 -p 80
```

Figura 23 Prueba HoneyPot Glastopf con Nmap



Fuente: El Autor.

6.3 CONFIGURACIÓN DE HONEYPOT SSH COWRIE (PROTOCOLO SSH)

Cowrie es un *HoneyPot* basado en *Kippo SSH* el cual ofrece un nivel de interacción media con el atacante. Esto lo hace simulando de manera precisa el servicio de

administración remota de *SSH* y *Telnet*[131]*,* imitando incluso un sistema de archivos falso el cual se apoya en el uso de *Twisted* quien permite enjaular el acceso del atacante a ciertos directorios específicos donde se encuentre instalada la herramienta. En los siguientes pasos se observa cómo configurar *Cowrie* para crear un *HoneyPot* destinado al protocolo *SSH.*

6.3.1 Instalación

a. Es necesario instalar todas las dependencias necesarias para ejecutar el *HoneyPot.* Para ello se debe ejecutar la siguiente instrucción en la terminal[132].

```
apt-get install git python-virtualenv libssl-dev libffi-dev
build-essential libpython-dev python2.7-minimal authbind
```

b. Antes de proseguir es necesario aclarar que, si se desea una mejor simulación del servicio SSH real, sería conveniente que el puerto donde escuche el *HoneyPot* sea el mismo dispuesto como estándar para *SSH,* es decir, el puerto 22. Por esta razón se debe verificar si el servidor tiene algún servicio de *SSH* legítimo para así cambiar el puerto estándar, en este caso ya se había reemplazado el puerto del servicio, en caso que se desee cambiar se deben seguir los siguientes pasos[132].

```
#Abrir el siguiente archivo
vi /etc/ssh/sshd_config

#La entrada Port 22 se debe cambiar por el número por el puerto
alternativo
```

La entrada a editar se puede ver con mayor claridad en la Figura 24.

[131] MICHELOOSTERHOF. Cowrie HoneyPot [online]. Github ene., 2018. [citado 26 mar., 2018]. Disponible en Internet < https://github.com/micheloosterhof/cowrie >

[132] TAKHION. Use the Cowrie SSH Honeypot to Catch Attackers on your Network [online]. Null byte ene., 2018. [citado 26 mar., 2018]. Disponible en Internet: < https://null-byte.wonderhowto.com/how-to/use-cowrie-ssh-honeypot-catch-attackers-your-network-0181600/ >

Figura 24 Cambio de entrada Port en archivo sshd_config para configuración de puerto alternativo para SSH

Fuente: El Autor.

```
#reiniciar el servicio ssh
systemctl restart ssh
```

c. Se procede a la adición de un nuevo usuario destinado para el *HoneyPot* para así garantizar una correcta segregación de permisos de usuario. Para ello se ejecutan los siguientes comandos[132].

```
adduser --disabled-password cowrie

su – cowrie # Se autentica como el usuario cowrie
```

d. Una vez se ha creado el usuario se clona el repositorio de *cowrie* en el directorio *home* del mismo, luego de esto con el comando *virtualenv* se crea el entorno de *Python* necesario para la ejecución aislada del *HoneyPot* con *twisted*[132].

```
git clone https://github.com/micheloosterhof/cowrie

cd cowrie

virtualenv cowrie-env
```

Nota: El repositorio puede ser creado en el directorio *home* por defecto cuando se crea el usuario *cowrie.* En este caso se usa el directorio */opt/HoneyPots/SshHoney*

e. Con el comando *source* se accede al nuevo entorno virtual de *Python* y luego se procede a instalar el resto de dependencias del proyecto.

```
#Accede al entorno virtual de Python
source cowrie-env/bin/activate

#Instala las dependencias adicionales de Cowrie
pip install --upgrade pip && pip install –upgrade –r
requirements.txt
```

6.3.2 Configuración de puerto estándar SSH e instalación como servicio de Systemd

a. Luego de haberse instalado todas las dependencias se procede a editar el archivo de configuración principal del *HoneyPot* el cual es *cowrie.cfg.* Para ello se copia el archivo del proporcionado como ejemplo en el repositorio.

```
cp cowrie.cfg.dist cowrie.cfg
```

Una vez se haya copiado el archivo se puede configurar varios atributos del *HoneyPot* como lo son el *hostname,* el puerto de escucha del servicio, *plugins,* directorios donde se suban los archivos por medio del *SSH* falso, llaves públicas y privadas, el banner del *HoneyPot,* entre otros. En este caso sólo se editará el puerto y el *hostname.*

El puerto se puede configurar en las entradas *listen_port* y *listen_endpoints,* se recomienda configurar ambos, aunque se hace claridad que el primero se encuentra obsoleto. La Figura 25 expone las entradas del archivo descritas anteriormente.

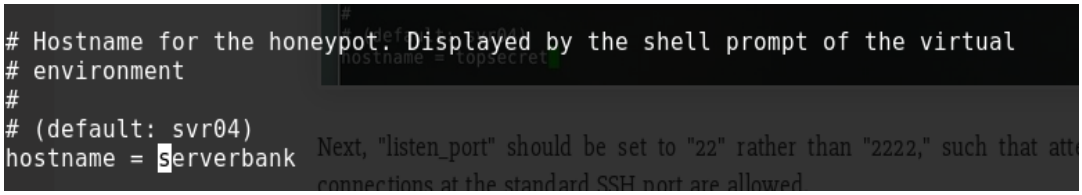
Figura 25 Configuración de puerto de escucha para HoneyPot Cowrie

```
# Port to listen for incoming SSH connections.
# (DEPRECATED: use listen_endpoints instead)
#
# (default: 2222)
listen_port = 2222

# Endpoint to listen on for incoming SSH connections.
# See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
# (default: listen_endpoints = tcp:2222:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
# listen_endpoints = systemd:domain=INET:index=0
# For both IPv4 and IPv6: listen_endpoints = tcp6:2222:interface=\:\:
listen_endpoints = tcp:2222:interface=0.0.0.0
```

Fuente: El Autor.

Por otro lado, el *hostname* se configura en la entrada homónima; en este caso se ha optado por el uso de un *hostname* tentador, algo que haga creer al atacante que se encuentra en una máquina importante como *ServerBank.* Esto se puede ver reflejado en la Figura 26.

Figura 26 Configuración de Hostname del HoneyPot Cowrie



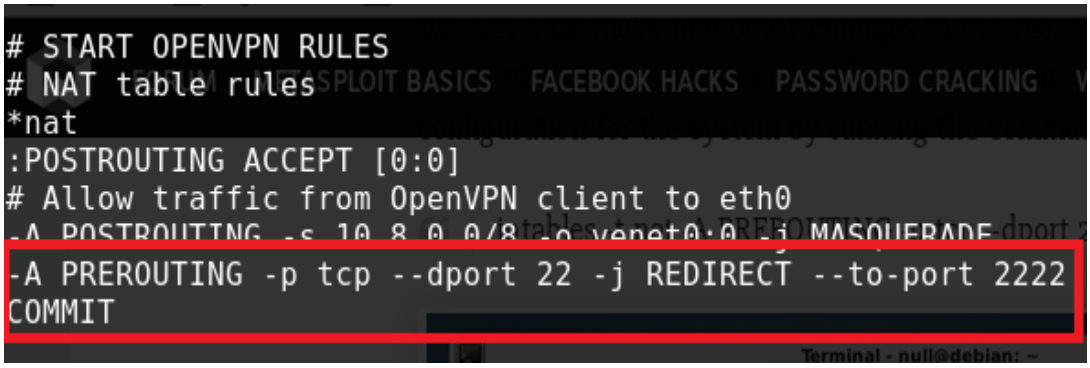
Fuente: El Autor.

Nótese que en las instrucciones anteriores se ha notificado que el puerto usado será el estándar, sin embargo, se ha dejado el puerto 2222. Esto es porque se debe ser el usuario *root* si se quiere usar el puerto 22, algo que traería un riesgo elevado en el despliegue del *HoneyPot*. En este caso se puede usar *Port forwarding* para redirigir el tráfico desde el puerto 22 al puerto 2222.

```
iptables –t nat –A PREROUTING –p tcp --dport 22 –j REDIRECT --top-port 2222
```

La instrucción anterior se encarga de hacer la redirección del tráfico, pero este cambio será volátil si no se tiene en cuenta el gestor de *firewall* de la distribución. En este caso se trata de un *UFW,* por lo cual se escribe la regla en el archivo */etc/ufw/before.rules* como se puede ver en los Figura 27.

Figura 27 Regla de Iptables para redirección de tráfico desde el puerto 22 al puerto de Cowrie HoneyPot



Fuente: El Autor.

Nota: Las instrucciones para la edición del archivo */etc/ufw/before.rules* e *iptables* deben ser ejecutadas como usuario *root.*

b. En este punto el *HoneyPot* ya está configurado completamente y puede ser desplegado desde el directorio *cowrie* como usuario homónimo con el comando.

```
bin/cowrie start
```

Sin embargo, lo ideal sería poderlo integrar al sistema de administración de servicios *Systemd* tal y como se hizo con *glastopf.* Para ello, el mismo repositorio ofrece un archivo que puede ser copiado directamente a */etc/systemd/system.*

```
cp cowrie/doc/systemd/cowrie.service /etc/systemd/system
```

Una vez copiado el archivo se procede a re configurar los servicios con el comando *systemctl daemon-reload.* Posteriormente se habilita el servicio para el arranque del sistema y se ejecuta.

```
systemctl enable cowrie.service
```

```
systemctl start cowrie.service
```

6.3.3 Pruebas de Despliegue

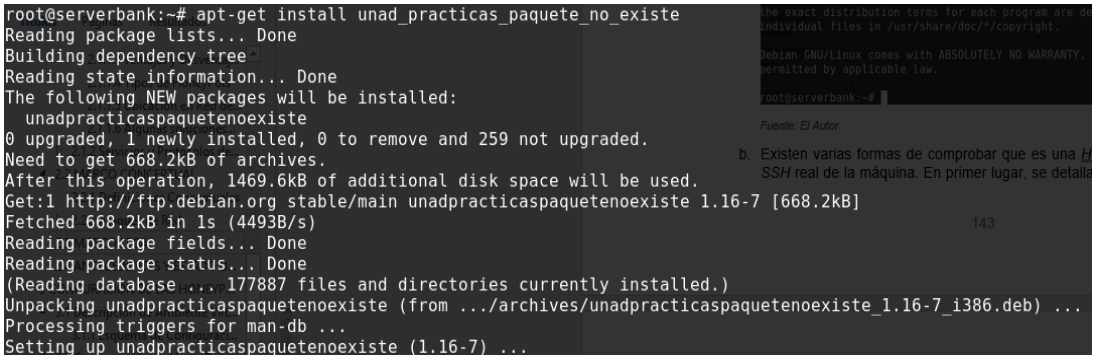
a. Para las pruebas de despliegue simplemente se procede a conectarse al servicio *SSH* del servidor, el cual en realidad sería atendido por el *HoneyPot Cowrie.* El *HoneyPot* responderá con una falsa autenticación, que normalmente al tercer intento otorgará la entrada al *HoneyPot* como se observa en la Figura 28.

Figura 28 Conexión con el usuario root al HoneyPot Cowire

Fuente: El Autor.

b. Existen varias formas de comprobar que es una *HoneyPot* y no el servicio *SSH* real de la máquina. En primer lugar, se detalla el *hostname,* el cual no corresponde a la máquina *Prometeo.* En segundo lugar, si se instala cualquier software por medio del comando *apt-get install* aunque el software no existe el *HoneyPot* dirá que es exitosa su instalación, como se ve en la Figura 29. Finalmente el archivo */etc/passwd* que contiene todos los usuarios del sistema muestra datos que no corresponden a ningún usuario real del servidor *Prometeo.*

Figura 29 Instalación de paquete inexistente por medio de HoneyPot Cowrie SSH



Fuente: El Autor.

6.4 CONFIGURACIÓN DE HONEYPOT SHIVA (PROTOCOLO SMTP)

Shiva es la abreviación de *Spam HoneyPot with Intelligent Virtual Analyzer.* Se trata básicamente de una *HoneyPot* que permite recoger correos *SPAM* simulando un *Open Relay.* Tiene la capacidad de almacenar toda la información que es enviada a través del servicio, identificando posibles estafas, ataques de *phishing,* campañas de *malware,* entre otros. Se divide en dos componentes bastante importantes los cuales son *Receiver* quien captura toda la información y *Analyzer* quien guarda todos los eventos sucedidos[133]. A continuación, se expone la manera en que se instala *Shiva* en el laboratorio haciendo uso de *MySQL*

[133] SHIVA-SPAMPOT. Shiva HoneyPot [online]. Github jun., 2016. [citado 26 mar., 2018]. Disponible en Internet: < https://github.com/shiva-spampot/shiva >

6.4.1 Instalación

a. Como sucede con las anteriores *HoneyPots* el primer paso es instalar las dependencias que requiere el proyecto, para ello se ejecuta la siguiente instrucción[134].

```
apt-get install python-dev exim4-daemon-light g++ python-virtualenv libmysqlclient-dev libffi-dev libfuzzy-dev mysql-server mysql-client
```

Nota: Quizás algunas dependencias ya estén instaladas, por lo tanto, el gestor de paquetes notificará de ello y obviará la instalación de dicho paquete.

Es necesario destacar que con la instrucción anterior se instalan los paquetes *mysql-server* y *mysql-client* los cuales corresponden al Sistema de Gestión de Bases de Datos.

b. El siguiente paso consiste en clonar el repositorio y guardarlo en una ruta para compilarlo y culminar la instalación apropiadamente, en este caso se guarda en la ruta */opt/HoneyPots/SmtpHoney*[134]*.*

```
git clone https://github.com/shiva-spampot/shiva.git shiva-installer
```

c. Este paso es opcional, aunque es altamente recomendado. Como el caso de la *HoneyPot* anterior, se creará un entorno virtual de *Python* por lo tanto se recomienda que dicho entorno esté ejecutado como un usuario diferente. Para ello se repite el proceso de crear un usuario sin tener habilitada las credenciales de usuario y contraseña.

```
adduser --disabled-password shiva
```

d. Luego de clonar el repositorio, el último paso para la instalación es ejecutar el *script install.sh* el cual se encuentra en la ruta *Shiva-installer*[134]*.*

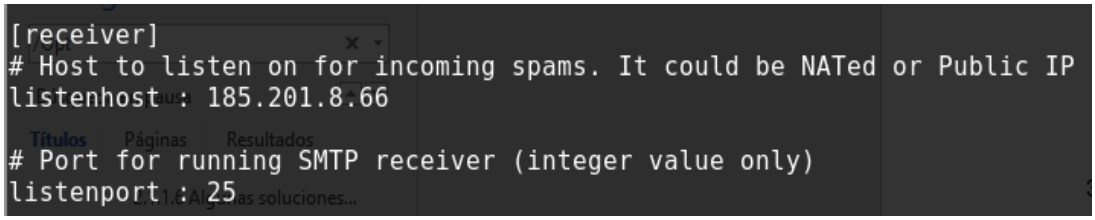
```
 cd shiva-installer
./install.sh
```

[134] PRITCHARD, Chris. Shiva the Spam HoneyPots: Tips and Tricks for getting it up and running [online]. Pentest Partners jun., 2015. [citado 26 mar., 2018]. Disponible en Internet: < https://www.pentestpartners.com/security-blog/shiva-the-spam-honeypot-tips-and-tricks-for-getting-it-up-and-running/ >

6.4.2 Configuración Global y MySQL

a. El archivo de configuración más importante para *Shiva* es *Shiva.conf* el cual se puede encontrar en la ruta *shiva-installer/shiva/shiva.conf.* En este archivo se ubica la configuración tanto del *receiver* como de *analyzer.* En primer lugar, se cambia la IP y puerto de escucha del *receiver* esto se hace en las entradas *listenhost* y *listenport*[134] como se puede ver en la Figura 30*.*

Figura 30 Configuración de IP y Puerto de escucha Receiver Shiva HoneyPot



Fuente: El Autor.

Como se puede observar la IP debe corresponder a alguna de las interfaces del servidor, en este caso se coloca la interfaz de IP pública de *Prometeo.*

Nota: Es posible que el puerto sea bloqueado por el ISP o por el mismo proveedor de VPS, para evitar esto se recomienda usar un puerto que no sea estándar.

b. Una vez se ha configurado el puerto se debe asegurar que únicamente se usará el medio de almacenamiento deseado, en este caso sólo se requiere que la información se guarde en *MySQL* por ello se colocan las entradas *enabled* de *hpfeeds* y *notification* en *False.*

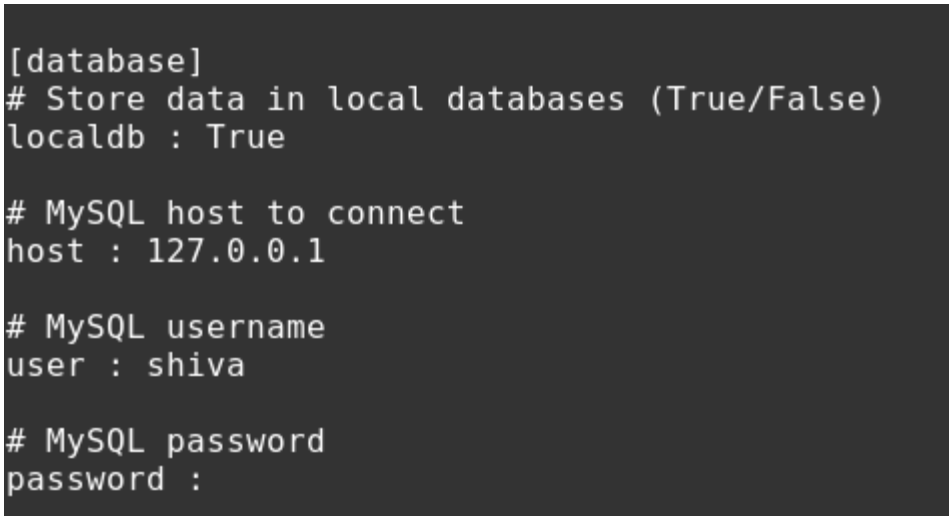
c. Se procede a crear un usuario en *MySQL* para el acceso de la *HoneyPot* hacia el motor de Bases de Datos, para ello se autentica frente al servicio del motor como *root* y se ejecutan las siguientes instrucciones *SQL.*

```
CREATE USER ‘shiva’@’localhost’ IDENTIFIED BY ‘password’;

GRANT ALL PRIVILEGES ON Shiva.* TO ‘shiva’@’localhost’;

GRAN ALLT PRIVILEGES ON ShivaTemp.* TO ‘shiva’@’localhost’;
```

Nota: Se recomienda usar una contraseña fuerte, en esta bitácora no se muestra la contraseña real usada en producción.

d. Se regresa de nuevo a la edición del archivo *Shiva.conf* donde se deben configurar las credenciales de acceso del nuevo usuario creado. Para ello se busca la entrada *database* cambiando los valores de *localdb, host, user* y *password,* como demuestra la Figura 31.

Figura 31 Configuración de Bases de Datos MySQL para Shiva HoneyPot



Fuente: El Autor.

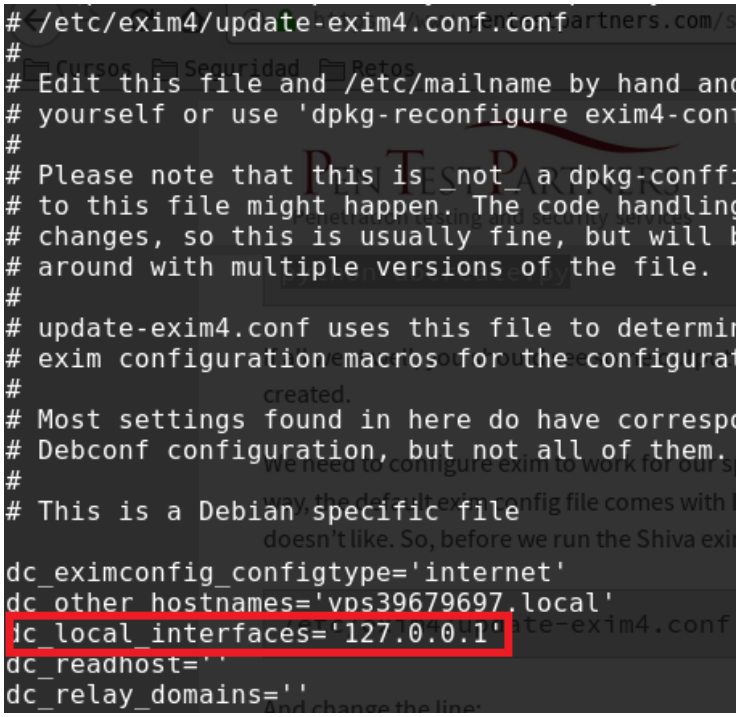
e. Luego se dirige al directorio *shiva-installer/shiva* y en él se ejecuta el *script* de *Python dbcreate.py.* Si todo sale correctamente se debería visualizar un mensaje que notifica que las Bases de Datos *Shiva* y *ShivaTemp* han sido creadas.

```
python dbcreate.py
```

e. *Shiva* tiene dificultades de despliegue y compatibilidad con *IPv6* por ello es altamente recomendado deshabilitar el componente de *IPv6* de *exim4,* uno de los elementos que compone a *Shiva*[135]*.* Para ello se debe editar el archivo */etc/exim4/update-exim4.conf.conf* en la línea *dc_local_interfaces,* quitando la dirección *::1* correspondiente a *IPv6,* como demuestra la Figura 32. En pocas palabras, la única dirección *IP* que debería quedar en esta entrada de configuración es la *localhost* de *IPv4.*

[135] PRITCHARD, Chris. Op. cit., 157 p.

Figura 32 Desactivación de IPv6 para exim4 Shiva HoneyPot.



Fuente: El Autor.

c. Una vez hecho esto se puede ejecutar el comando *sh setup_exim4.sh* el cual se encuentra en el directorio *shiva-installer/shiva.*

6.4.3 Configuración como servicio en Systemd

a. Hasta este punto el *HoneyPot* ya estaría configurado y listo para desplegarse, sin embargo, es un poco tedioso su despliegue ya que requiere varias instrucciones y además se trata de dos componentes diferentes. Para simplificar los pasos de arranque, detención del *HoneyPot* se recomienda colocar los siguientes archivos en la ruta */etc/systemd/system.*

```
#Archivo para la configuración de shiva_receiver.service

[Unit]
Name=Shiva Receiver
Description = Shiva Receiver HoneyPot
After=multi-user.target
```

```
[Service]
Type=idle
User=root
WorkingDirectory=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaReceiver/receiver
ExecStart=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaReceiver/bin/lamson start
ExecStop=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaReceiver/bin/lamson stop

[Install]
WantedBy=multi-user.target

#Archivo para la configuración de shiva_analyzer.service

[Unit]
Name=Shiva Analyzer
Description = Shiva Analyzer Data
After=multi-user.target

[Service]
Type=idle
User=shiva
Group=shiva
WorkingDirectory=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaAnalyzer/analyzer
ExecStart=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaAnalyzer/bin/lamson start
ExecStop=/opt/HoneyPots/SmtpHoney/shiva-
installer/shiva/shivaAnalyzer/bin/lamson stop

[Install]
WantedBy=multi-user.target
```

b. Una vez guardado los dos archivos se procede a ejecutar las siguientes instrucciones que permitirán habilitar el arranque automático de *Shiva.*

```
systemctl daemon-reload

systemctl enable exim4

systemctl enable mysql

systemctl enable shiva_receiver
```

```
systemctl enable shiva_analyzer
```

c. Ahora los componentes *Receiver* y *Analyzer* de *Shiva* pueden ser inicializados por medio de *systemctl* siempre y cuando se estén ejecutando los servicios *exim4* y *mysql.*

6.4.4 Pruebas de Despliegue

a. El servicio *Shiva Receiver* se encuentra en ejecución y corriendo bajo el puerto estándar 25 para *SMTP.* Una manera bastante sencilla de comprobarlo es usando un cliente de *telnet.* Para ello se escribe la dirección y el puerto, luego se ejecutan los siguientes comandos.

```
EHLO prometeosec.com
MAIL FROM:zero_bat@gmail.com
RCPT TO: noexiste@unad.co
DATA
Hello this is a probe

.
```

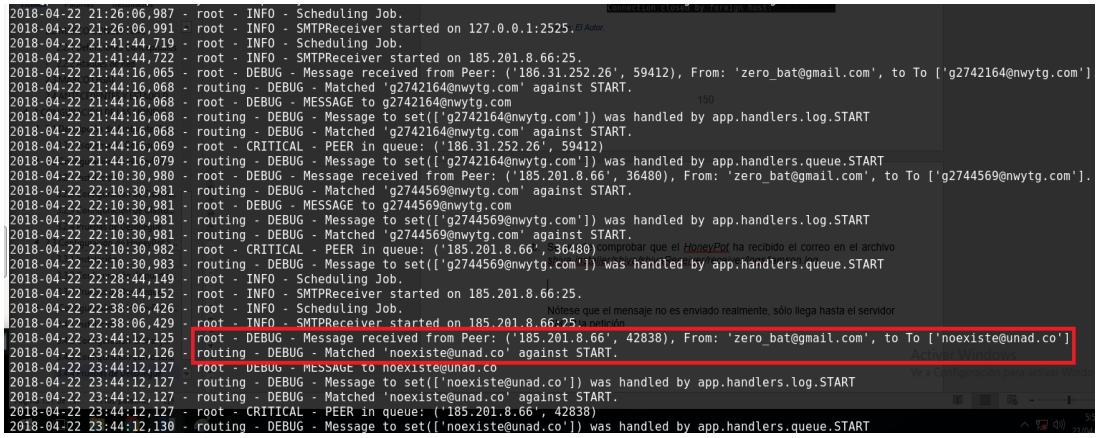
Los resultados de la ejecución de estos comandos se observan en la Figura 33.

Figura 33 Prueba de SMTP con Telnet a Shiva HoneyPot

Fuente: El Autor.

b. Se puede comprobar que el *HoneyPot* ha recibido el correo en el archivo *shiva-installer/shiva/shivaReceiver/receiver/logs/lamson.log.* La salida del archivo se puede ver en la Figura 34.

Figura 34 Evidencia de envío del mensaje SMTP al Shiva HoneyPot



Fuente: El Autor.

El mensaje será enviado al destinatario siempre y cuando cumpla las condiciones establecidas en el archivo de configuración *shiva.conf.*

Nota: También se recomienda observar el archivo de log de *exim4* en caso que se desee comprobar que el mensaje ya fue enviado, este archivo se encuentra en */var/log/exim4/mainlog*

6.5 CONFIGURACIÓN DE HONEYPOT OPENLDAP (PROTOCOLO LDAP)

A la fecha no se consiguió ningún software que prestase el servicio especializado de *HoneyPot* para *LDAP.* Sin embargo, es posible configurar un servicio real de *LDAP* y monitorearlo siempre y cuando sus datos no correspondan a un ambiente productivo real. De esta manera se puede simular el comportamiento de un *HoneyPot* de alta interacción. A continuación, se describe la manera de realizar una instalación básica sobre el *software OpenLDAP* quien ofrece las capacidades y acceso bajo el protocolo *LDAP,* para así capturar las actividades maliciosas.

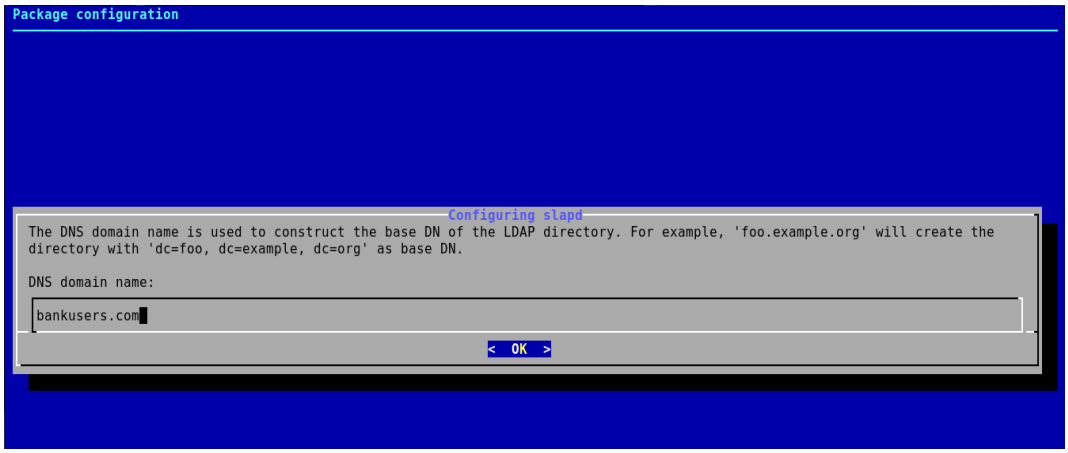
6.5.1 Instalación

a. La instalación de *OpenLDAP* es bastante sencilla a comparación de los demás servicios de *HoneyPots,* debido a que se trata de un paquete ya disponible para las distribuciones debían. Únicamente se debe instalar los paquetes *slapd* y *ldap-utils*[136]*.*

```
apt-get install slapd ldap-utils
```

b. Una vez instalados los paquetes, se procede a reconfigurar los atributos del servicio. Para ello se ejecuta el comando *dpkg-reconfigure slapd.* Con este comando se configura el nombre de dominio, el nombre del *DN,* la clave de administrador o *root* de *LDAP* y otros atributos[136]. Para mayor claridad se presenta la Figura 35.

Figura 35 Configuración de DNS Domain para servicio OpenLDAP



Fuente: El Autor.

Nota: Es recomendable definir una contraseña fuerte para el administrador, debido que se trata de un servicio real. También como se puede observar se opta por la utilización de un nombre de dominio llamativo como *BankUsers.com.*

De igual manera se escoge el sistema de almacenamiento, en este caso una Base de Datos *MDB* y se habilita el protocolo *LDAPv2.*

---

[136] BOUCHERON, Brian. How to Install and Configure OpenLDAP and phpLDAPadmin on Ubuntu 16.04 [online]. Digital Ocean jun., 2017. [citado 27 mar., 2018]. Disponible en Internet: < https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-a-basic-ldap-server-on-an-ubuntu-12-04-vps >

6.5.2 Configuración de Loggingc con rsyslog

a. Una vez se ha creado el servicio, es necesario recolectar la información de los usuarios que se conecten a *OpenLDAP* para ello se puede usar la siguiente instrucción[137].

```
ldapsearch -Y external -H ldapi:/// -b cn=config
"(objectClass=olcGlobal)" olcLogLevel -LLL > slapdlog.ldif
```

Este comando produce un archivo llamado *slapdlog.ldif* con la siguiente estructura.

```
dn: cn=config
olcLogLevel: none
```

b. Se edita el archivo *slapdlog.ldif* para agregar las siguientes líenas.

```
dn: cn=config
changeType: modify
replace: olcLogLevel
olcLogLevel: stats
```

c. Finalmente se puede cargar el archivo con el comando *ldapmodify,* esto modificará la entrada del *CN* Config habilitando el nivel de logs a *stats.*

```
ldapmodify -Y external -H ldapi:/// -f slapdlog.ldif
```

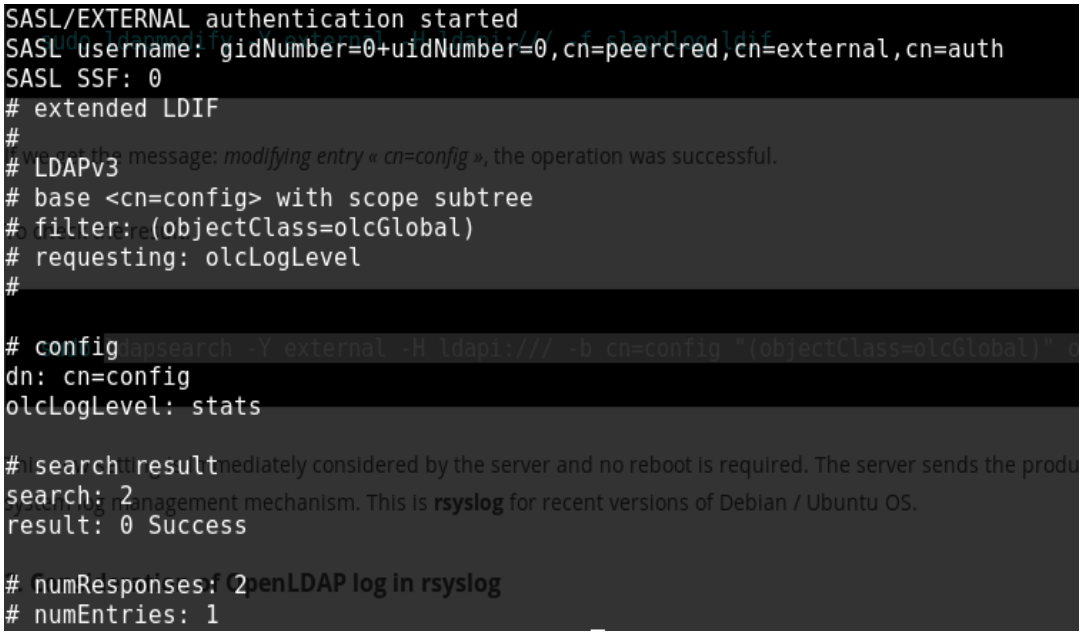
La modificación se puede comprobar con la siguiente instrucción.

```
ldapsearch -Y external -H ldapi:/// -b cn=config
"(objectClass=olcGlobal)" olcLogLevel
```

Como se expone en la Figura 36 se puede ver que se ha realizado un seguimiento del comando ejecutado anteriormente.

[137] TUTORIELS MEDDEB. Enable the production of Openldap Log File [online]. Tutoriels Meddeb sept., 2017. [citado 27 mar., 2018]. Disponible en Internet: < http://tutoriels.meddeb.net/openldap-tutorial-log/ >

Figura 36 Verificación de habilitación de Logs en Producción HoneyPot OpenLDAP



Fuente: El Autor.

d. Posteriormente es requerido configurar *rsyslog* para separar los eventos recogidos por el *HoneyPot* OpenLDAP. Esto se hace agregando una entrada en el directorio */etc/rsyslog.d/10-slapd.conf* [137]. En este archivo se debe colocar la siguiente instrucción.

```
$template                    slapdtmpl,"[%$DAY%-%$MONTH%-%$YEAR%
%timegenerated:12:19:date-rfc3339%]                    %app-name%
%syslogseverity-text% %msg%\n"
local4.*    /var/log/slapd.log;slapdtmpl
```

e. Una vez se ha guardado el archivo anterior se reinicia el servicio de *rsyslog* con la instrucción *systemctl restart syslog.*

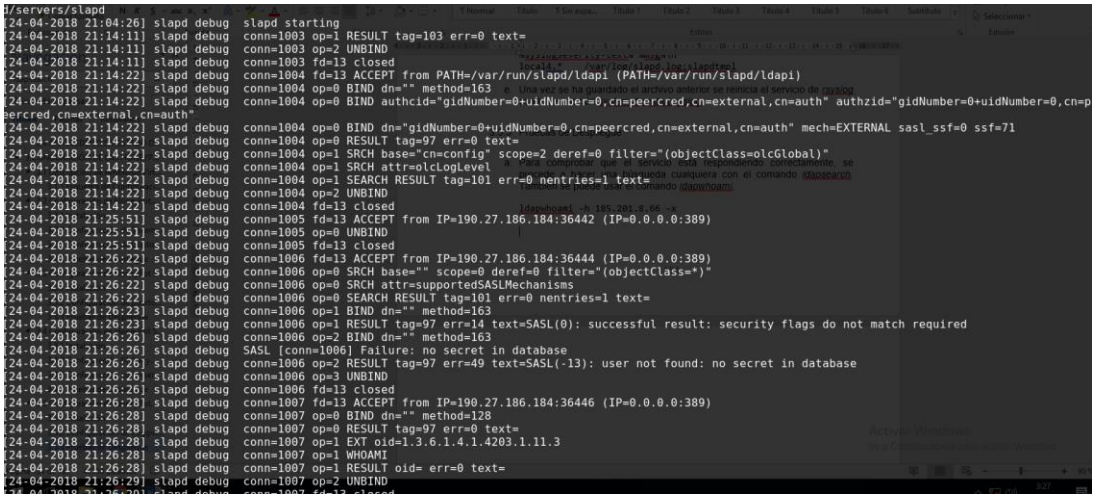
6.5.3 Pruebas de Despliegue

a. Para comprobar que el servicio está respondiendo correctamente, se procede a hacer una búsqueda cualquiera con el comando *ldapsearch.* También se puede usar el comando *ldapwhoami.*

```
ldapwhoami –h 185.201.8.66 –x
```

Se comprueba que se las acciones han sido registradas correctamente en el log de *Slapd.* Para ello se debe observar el log */var/log/slapd.log.* La salida de este archivo se muestra en la Figura 37.

Figura 37 Registro de actividades de OpenLDAP en archivo slapd.log



Fuente: El Autor.

7 HERRAMIENTAS DE SOFTWARE LIBRE PARA MONITORIZACIÓN DE ATAQUES Y ATRAVÉS DE LOS HONEYPOTS

Una vez configurados los *HoneyPots* se hace necesario instalar y configurar otro tipo de sistemas de monitorización que permitan asegurar que el ambiente no ha sido comprometido. Cada *HoneyPot* es un tipo de *software* o aplicación, como todo programa informático puede ser susceptible a ciertas vulnerabilidades. El objetivo principal de las herramientas de monitorización, será la constante vigilancia del proyecto. Además de ello, estos sistemas de vigilancia pueden identificar otro tipo de amenazas que podrían ser analizadas en el siguiente capítulo.

Para la instalación del *software* de monitorización, se opta por herramientas libres ya que son gratuitas y disponen de bastante documentación en línea lo cual es bastante conveniente para el desarrollo de proyectos de bajo costo.

7.1 SNIFFER TCPDUMP

TcpDump es una poderosa herramienta que puede ayudar al análisis y captura de tráfico de red para el diagnóstico e identificación de problemas en un *host* específico. La herramienta originalmente se encuentra escrita en *C/C++* y se apoya de las librerías *libpcap* las cuales le permiten filtrar y capturar el tráfico sobre las diferentes interfaces de red de un *host*[138]*.*

Originalmente la herramienta imprime los tipos de eventos sucedidos sobre una o varias interfaces de red, teniendo en cuenta una expresión *booleana* la cual está basada en la sintaxis de *libpcap.* Los mensajes impresos por la herramienta pueden ser precedidos por una marca de tiempo la cual puede tener una estructura definida por el usuario[138]. Del mismo modo, esta herramienta puede volcar todo el tráfico capturado hacia uno o varios archivos permitiendo al analista persistir la evidencia deseada para así analizarla posteriormente mediante otro tipo de herramientas.

TcpDump por sí solo puede ofrecer una interacción con el usuario meramente por línea de comandos, a pesar de ello existen otro tipo de herramientas como *Wireshark*[138] y otros *sniffers* más elaborados que lo implementan dentro de su funcionamiento. Adicionalmente, se puede asegurar que *TcpDump* viene preinstalado en la mayoría de los Sistemas Operativos basados en el *kernel* de Unix, sólo requiere de privilegios de administrador para ejecutarlo.

[138] TCPDUMP. TcpDump Man Page [online]. TcpDump. Feb., 2017 [citado 1 abr., 2018]. Disponible en Internet: < https://www.tcpdump.org/tcpdump_man.html >

La licencia de esta herramienta está sobre la cláusula 3 de *BSD* por lo cual se considera *Open Source.*[138]

La selección de un *sniffer* ayudará a corroborar los datos arrojados por las *HoneyPots* y además identificar falsos negativos y falsos positivos.

7.1.1 Instalación y Configuración para Monitoreo de los HoneyPots

a. Generalmente *TcpDump* viene instalado por defecto en las distribuciones basadas en *Debian.* Sin embargo, si no se encuentra instalado se puede realizar la instalación por medio del siguiente comando.

```
apt-get install tcpdump
```

b. Una vez instalada la herramienta se procede a configurar un comando que permita monitorear el tráfico de los puertos de los *HoneyPots.* Para ello hay que tener en cuenta que los puertos son los siguientes:

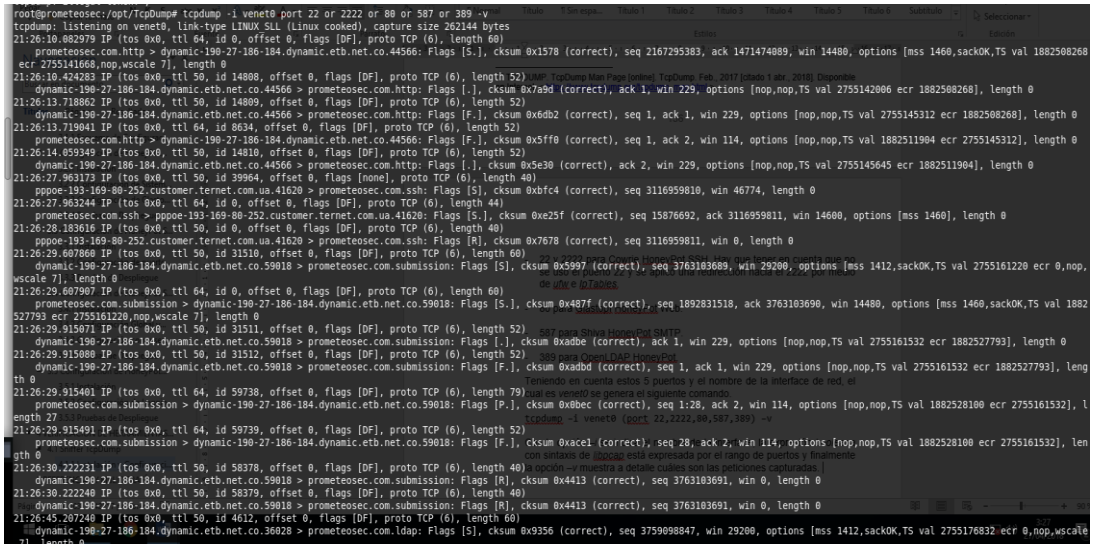
- 22 y 2222 para Cowrie HoneyPot SSH. Hay que tener en cuenta que no se usó el puerto 22 y se aplicó una redirección hacia el 2222 por medio de *ufw* e *IpTables.*
- 80 para Glastopf HoneyPot Web.
- 587 para Shiva HoneyPot SMTP.
- 389 para OpenLDAP HoneyPot.

Teniendo en cuenta estos 5 puertos y el nombre de la interface de red, el cual es *venet0* se genera el siguiente comando.

```
tcpdump –i venet0 port 22 or 2222 or 80 or 587 or 389 –v
```

Con la opción *–i* se indica el nombre de la interface, la expresión *booleana* con sintaxis de *libpcap* está expresada por el rango de puertos y finalmente la opción *–v* muestra a detalle cuáles son las peticiones capturadas. La salida del comando anterior se observa en la Figura 38.

Figura 38 Sniffer de los puertos de los HoneyPots con TCPDump



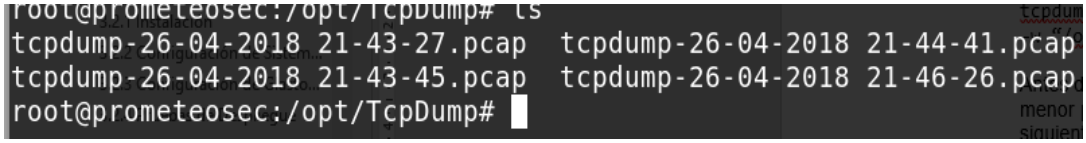
Fuente: El Autor.

c. El siguiente paso es definir una carpeta donde estarán guardados todos los archivos generados por el *sniffer,* en este caso será la carpeta */opt/TcpDump.* La idea es guardar un archivo cada seis horas, por lo cual se usa la opción –*G* que permite definir el tiempo rotatorio de archivos en segundos, también se define el formato de fechas con la estructura estándar de Unix[139], esto se define en la opción –*w* la cual indica el nombre del archivo a guardar.

```
tcpdump –i veneto port 22 or 2222 or 80 or 587 or 389 –G 28800
–w “/opt/TcpDump/tcpdump-%d-%m-%Y %H-%M-%S.pcap”
```

Antes de iniciarlo con los 28800 segundos, se recomienda colocar un umbral menor para verificar que la nomenclatura de los archivos es correcta. En la siguiente imagen se observa la estructura generada con la opción –*G* de 10 segundos. Como se expone en la Figura 39.

Figura 39 Archivos generados por TCPDump con la opción –G 10



Fuente: El Autor.

[139] NIXCRAFT. How to Format Date for Display or Use in a Shell Script [online]. Nixcraft, mar., 2016 [citado 1 abr., 2018]. Disponible en Internet: < https://www.cyberciti.biz/faq/linux-unix-formatting-dates-for-display/ >

d. Probablemente se desee mejorar el rendimiento de las tramas recogidas, por ello es necesario agregar las opciones *–K* y *–n* las cuales indicarán a la herramienta que no se desea convertir ninguna dirección, puerto o demás a un nombre (*-n*) y que no se desea verificar el *checksum* en especial de los paquetes de *TCP* (*-K*).

```
tcpdump –i venet0 port 22 or 2222 or 80 or 587 or 389 –G 28800 –w “/opt/TcpDump/tcpdump-%d-%m-%Y %H-%M-%S.pcap” –z gzip –K –n
```

También se ha agregado la opción *–z gzip* la cual permite comprimir los archivos una vez se ha terminado el proceso de rotación, es decir, cada vez que se haya completado los 28800 segundos.

e. Finalmente se puede enviar a *background* el proceso integrándose a un *script* de *Systemd.*

```
[Unit]
Description=TCP Dump for HoneyPots
After=network.target

[Service]
Type=idle
User=root
Group=root
WorkingDirectory=/opt/TcpDump
ExecStart=/usr/sbin/tcpdump -i venet0 port 22 or 2222 or 80 or 587 or 389 -G 28800 -w "tcpdump-%%d-%%m-%%Y-%%H-%%M-%%S).pcap" –z gzip -K -n
ExecStop=/bin/kill -15 $(/usr/bin/pgrep -f "tcpdump -i venet0 port 22 or 2222 or 80 or 587 or 389")

[Install]
WantedBy=multi-user.target
```

Nota: Se debe ser cuidadoso con el formato de la fecha, ya que *Systemd* requiere que se escape el carácter *%* agregando *%%.*

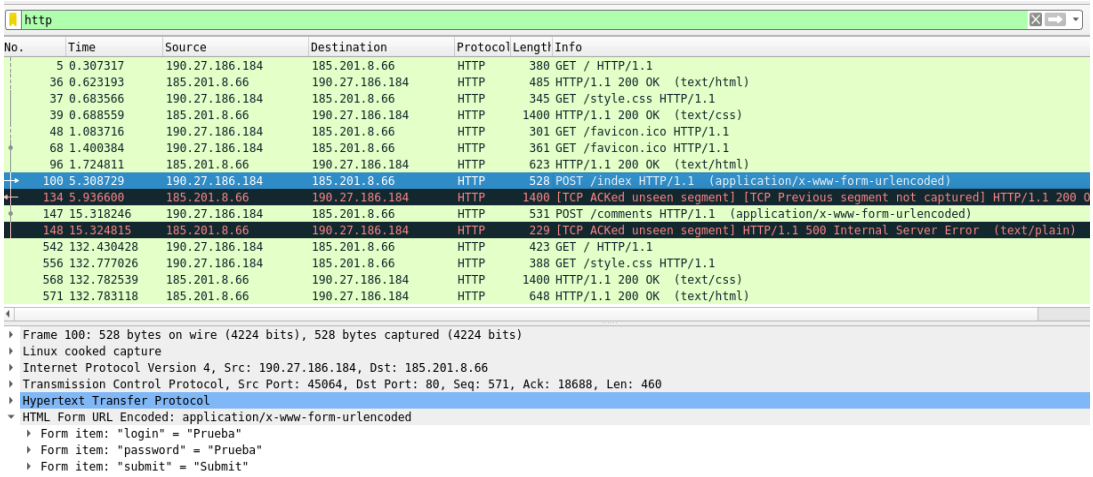
f. Una vez se ha creado el *script* se puede controlar fácilmente con los comandos *systemctl start tcpdump* y *systemctl stop tcpdump.* En este caso no se recomienda habilitarse al inicio para no saturar el proceso de arranque del servidor.

7.1.2 Verificación de las tramas capturadas

a. Se procede a iniciar el servicio de *TcpDump* y a generar tráfico de manera aleatoria para cada uno de los protocolos a estudiar. Luego de generar el tráfico se detiene el servicio *TcpDump* y se extraen los archivos de la carpeta */opt/TcpDump* hacia una máquina donde puedan ser estudiados. El archivo se abre con *Wireshark.*

b. Primero se inicia con las peticiones hacia el *HoneyPot Glastopf* para ello se selecciona un filtro *HTTP.* Se puede observar una de las peticiones *POST* que se hizo durante las pruebas, esta petición se hace hacia la *URI /index.* La Figura 40 demuestra lo anterior.

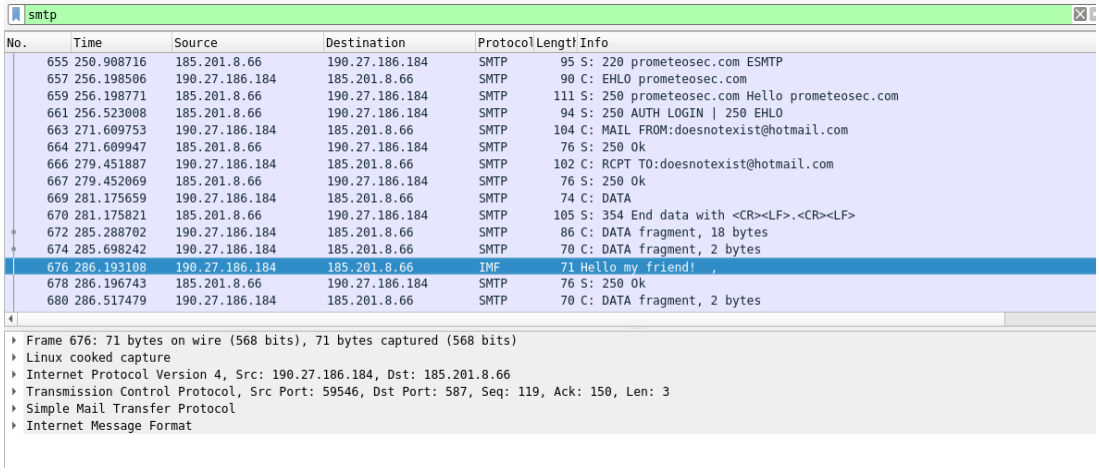
Figura 40 Verificación de Tráfico recogido hacia HoneyPot Glastopf por Tcp Dump.



Fuente: El Autor.

c. El siguiente paso es verificar el tráfico generado hacia el *HoneyPot Shiva.* En este caso se selecciona el filtro *SMTP.* Se puede observar el tráfico generado en las pruebas, el correo enviado desde *doesnotexist@hotmail.com* hacia él mismo como se ve en la Figura 41.

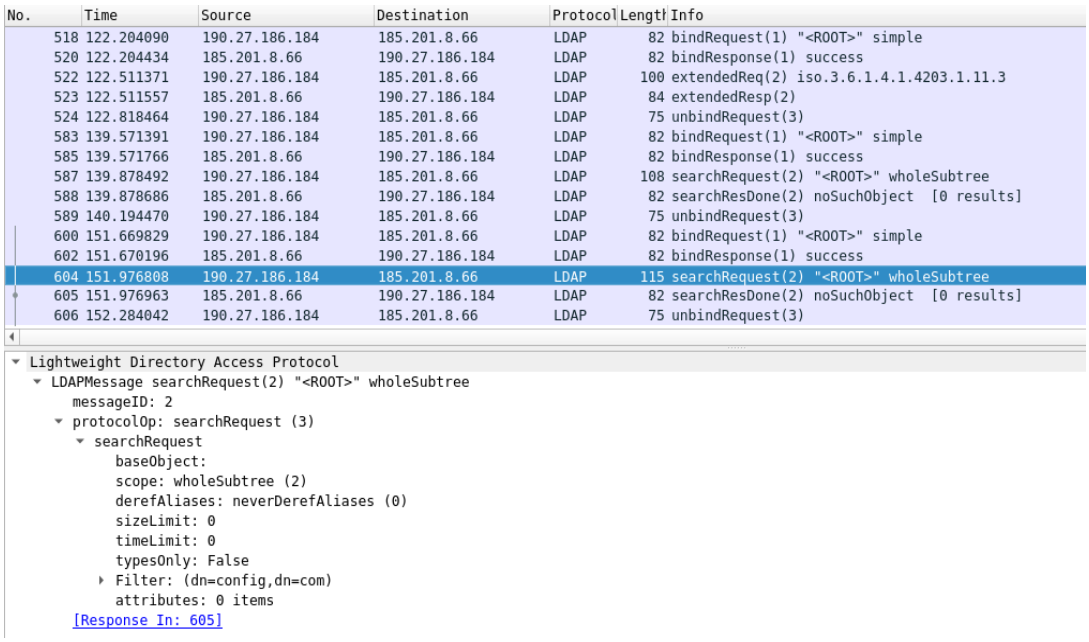
Figura 41 Verificación de Tráfico recogido hacia HoneyPot Shiva por Tcp Dump



Fuente: El Autor.

d. Se procede a verificar el tráfico generado hacia el *HoneyPot LDAP.* En este caso se selecciona el filtro *LDAP.* Se puede observar la búsqueda *"dn=config, dn=com"* el cual fue generado por el comando *ldapsearch.* Lo anterior se puede observar en la Figura 42.

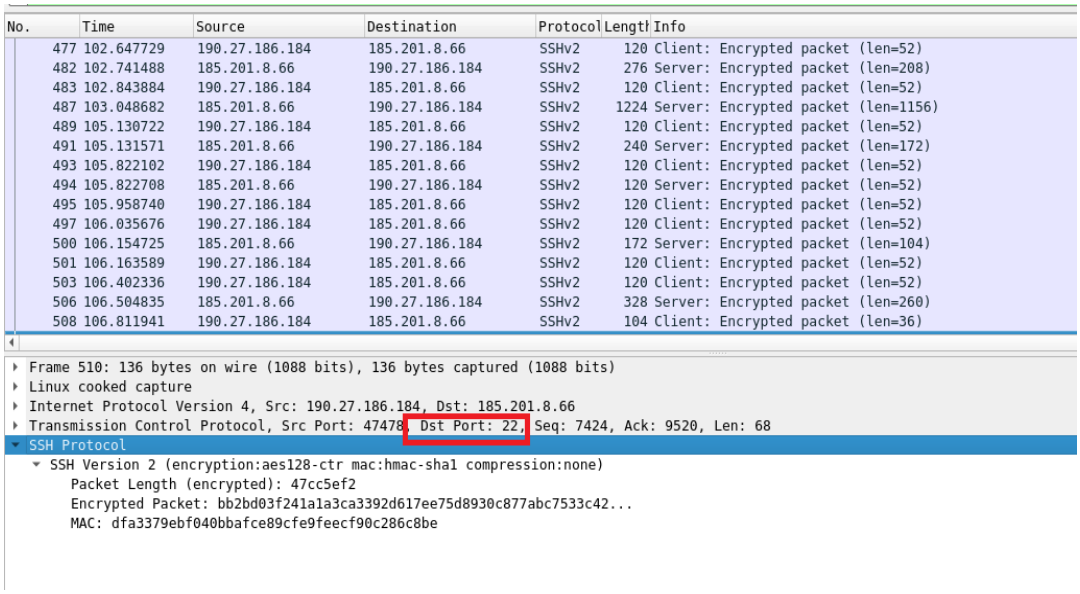
Figura 42 Verificación de Tráfico Recogido hacia HoneyPot LDAP por Tcp Dump



Fuente: El Autor

e. En cuanto al protocolo *SSH* aun contando con las llaves públicas y privadas, no se puede descifrar el tráfico generado; por lo cual únicamente se puede observar en el archivo de la captura que hubo un intercambio entre el cliente y servidor que involucra a la *HoneyPot Cowrie,* esto se observa por el protocolo *TCP* ya que el destino de los paquetes es el puerto 22, el cual hace *Port Forwarding* al puerto 2222. Al igual que en los casos anteriores, se selecciona el filtro *SSH,* el resultado se puede observar en la Figura 43.

Figura 43 Verificación de Tráfico Recogido hacia HoneyPot Cowrie SSH por Tcp Dump



Fuente: El Autor.

7.2 LINUX MALWARE DETECT LMD O MALDET

Linux Malware Detect, como su nombre lo indica, es una herramienta que ofrece las capacidades de búsqueda de *malware, rootkits* y otro tipo de amenazas que pueden afectar a un *host* en específico que ejecute un Sistema Operativo basado en Linux. Linux Malware Detect, abreviado como *LMD* basa su funcionamiento en el uso de firmas sobre amenazas ya conocidas, una técnica bastante usada en los diferentes antivirus. Cuenta con una comunidad abierta de investigadores que constantemente

actualizan las Bases de Datos para así garantizar la mayor precisión de la detección de amenazas[140].

Entre algunas de las características más importantes de esta herramienta se encuentra la detección por medio de *hash MD5,* análisis estático sobre amenazas que puedan contener código ofuscado, la garantía de encolar y almacenar amenazas en un entorno seguro donde no tengan permisos, la garantía de restaurar falsos positivos con sus respectivos propietarios y permisos, integración con otras herramientas como *ClamAV*, monitorear los eventos causados por *inotify* para vigilar el acceso a ciertos archivos específicos, opciones de escaneo de todo en uno con una simple línea de comando, opciones de ignorar ciertos directorios, ofrecer un escaneo en *background* que no necesariamente sea atendido, aplicar bitácoras sobre las amenazas detectadas y más[140].

Actualmente *Maldet* o *LMD* cuenta con una licencia de *GNU GPLv2* lo cual lo ubica en la categoría de *Software Libre*[140]*.*

Elegir una herramienta para el monitoreo a nivel de *host* puede ayudar a prevenir un *Takeover* del *host,* es decir, la pérdida completa del control del Servidor donde se alojan los *HoneyPots* porque una amenaza se haya materializado realmente.

7.2.1 Instalación y Configuración

a. Inicialmente se debe descargar la última versión de *Maldetc*[141]*.* Esto se hace con los siguientes comandos. Actualmente se encuentra en la versión *1.6.2.*

```
cd /tmp/
curl -O http://www.rfxn.com/downloads/maldetect-current.tar.gz
tar –zxvf maldetect-current.tar.gz
```

b. Luego de ello se ejecuta el *script* de instalación. Este *script* se encargará de instalar el servicio de la herramienta y actualizar las firmas de virus, durante la actualización se puede observar la sincronización de elementos importantes como las firmas de *hashes MD5, Yara Rules,* identificación de binarios por medio de hexadecimales, entre otros.

---

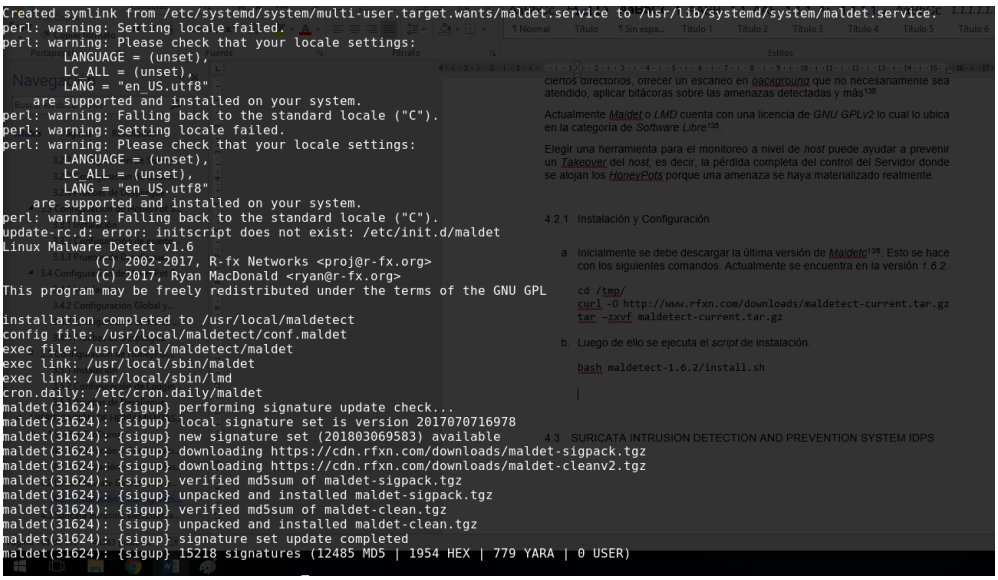
[140] R-FX NETWORKS. Linux Malware Detect [online]. R-fx Networks, mar., 2017. [citado 1 abr., 2018]. Disponible en Internet: < https://www.rfxn.com/projects/linux-malware-detect/ >

[141] RAJ. Install Linux Malware Detecto on Debian / Ubuntu / LinuxMint – A Malware Scanner for Linux Operating System [online]. IT'z Geek, nov., 2017 [citado 1 abr., 2018]. Disponible en Internet: < https://www.itzgeek.com/how-tos/linux/debian/install-linux-malware-detect-on-debian-ubuntu-linuxmint-a-malware-scanner-for-linux-operating-system.html >

```
bash maldetect-1.6.2/install.sh
```

La salida de este comando se expone en la Figura 44.

Figura 44 Salida de Script de Instalación Maldetect



Fuente: El Autor

.

c. *LMD* funciona mucho mejor en compañía de la herramienta *CalmAV,* el cual es un antivirus *Open Source* que puede detectar otro tipo de amenazas que se escapan del alcance inicial de *LMD.* Por ello se recomienda instalar *CalmAV* con el siguiente comando.

```
apt-get -y install clamav clamav-daemon clamdscan
```

d. El archivo de configuración más importante de *LMD* se encuentra en la ruta */usr/local/maldetect/conf.maldet.* Muchas de las opciones recomendadas, como la integración con *ClamAV,* monitorización de *inotify,* notificaciones por correos electrónicos, entre otras; son configuradas en este fichero. Por el momento todas las configuraciones por defecto se dejan intactas ya que se considera que proporcionan el nivel requerido de seguridad.

e. El módulo de *inotify* requiere la especificación de un arreglo de directorios los cuales se vigilarán constantemente, esto no quiere decir que se puedan prevenir las amenazas en tiempo real, pero si se monitoreará movimientos sospechosos. Para indicar a *LMD* que monitoree algunos directorios importantes se debe cambiar el archivo */usr/local/maldetect/monitor_paths.*

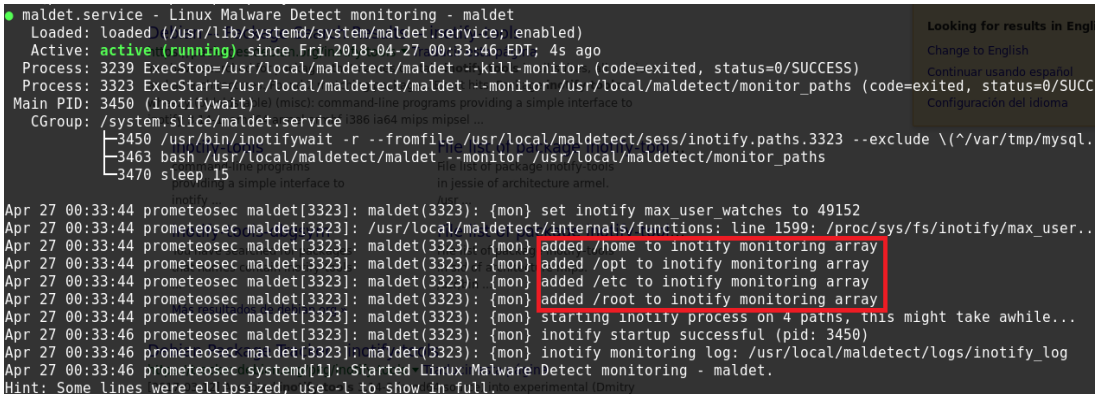
En esta práctica se agregan los *paths* correspondientes a los *HoneyPots,* la configuración y el directorio *home* de los usuarios.

f. Finalmente, luego de ello se procede a iniciar el servicio para ello primero se debe desenmascarar el servicio de *maldet* y después de ello se debe iniciar.

```
systemctl unmask maldet && systemctl start maldet
```

El resultado de este comando se aprecia en la Figura 45.

Figura 45 Monitoreo de algunos directorios específicos con Maldet o LMD



Fuente: El Autor.

f. Durante la instalación se puede notar que se ha configurado un *cron job* de manera diaria en el archivo */etc/cron.daily/maldet.* Este archivo realiza actualizaciones automáticas sobre las firmas de virus y escanea algunos de los directorios más importantes y preferidos por los atacantes.

g. Todos los eventos de *Maldetect* quedan registrados en la carpeta */usr/local/maldetect/logs,* en este directorio se encuentran tres archivos: *clamscan_log* el cual contiene todos los eventos sucedidos por *Clam Scan, event_log* el cual contiene cualquier evento provocado por la herramienta e *inotify_log* quien contiene todas las alertas que se observan sobre el módulo *INotify* [142].

[142] RAJ. Op. cit., 175 p.

7.2.2 Verificación de LDM con archivos EICAR

a. *EICAR* es una organización dispuesta para mejorar la seguridad en Internet. Es reconocida por ofrecer archivos para el testeo de *malware,* es decir, para comprobar que una herramienta de detección de amenazas realmente funciona. En este caso se usa la herramienta para comprobar que *LMD* puede detectar las amenazas[143].

   Con el siguiente comando se descargan dos archivos *EICAR* y se colocan en la carpeta */tmp.*

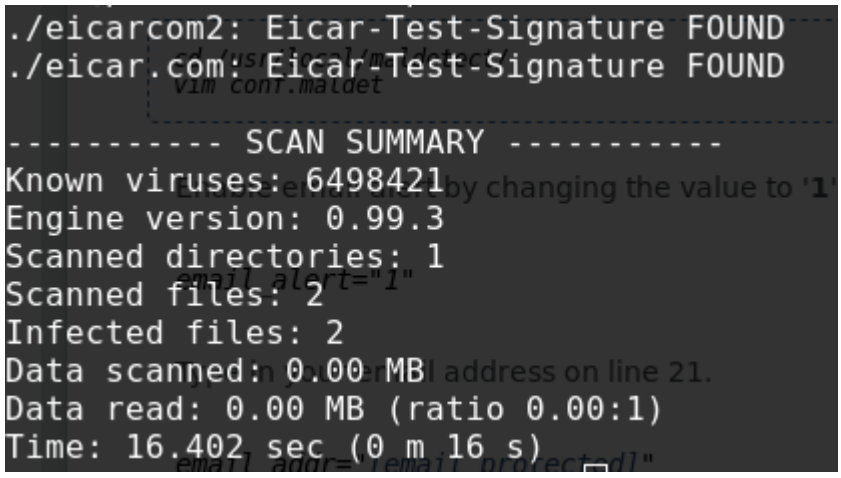
```
wget      http://www.eicar.org/download/eicar_com.zip      -O
/tmp/eicar_com.zip
wget      http://www.eicar.org/download/eicarcom2.zip      -O
/tmp/eicarcom2.zip
```

b. Se ejecuta la siguiente instrucción sobre el directorio */tmp* para comprobar que el escáner de malware puede identificar los archivos *EICAR.*

```
maldet –a /tmp
```

   La Figura 46 define la salida del comando.

Figura 46 Prueba de archivo EICAR para LMD o Maldet



Fuente: El Autor.

[143] EICAR. European Expert Group For IT-Security [online]. Eicar mar., 2018. [citado 1 abr., 2018]. Disponible en Internet: < http://www.eicar.org/ >

7.3 SNORT INTRUSION DETECTION AND PREVENTION SYSTEM IDPS

Snort es un Sistema de Detección y Prevención de Intrusos, abreviado como *IDPS,* el cual se caracteriza por tener la capacidad de monitorear el tráfico entrante y saliente de una o más interfaces de red. Esto se logra verificando un conjunto de reglas predefinidas, las cuales pueden ser descargadas por el usuario o desde fuentes oficiales, quienes indican a *Snort* cómo se clasifican las amenazas[144].

Entre sus capacidades se encuentran el hecho de ser multiplataforma, ya que ofrece soporte para *Windows,* sistemas basados en *Unix,* basados en *MAC* y el *Linux.* Cuenta con una gran comunidad de desarrolladores e investigadores que constantemente están actualizando las firmas y reglas con las cuales se alimenta *Snort,* también tiene un foro oficial donde se puede encontrar una amplia información sobre el manejo y administración de la herramienta, finalmente la herramienta no necesita de un *hardware* especializado para ejecutarse.

Snort es liviano, en comparación de algunos otros Sistemas de Prevención y Detección de Intrusos y además de ello es *OpenSource.* Se puede instalar una versión gratuita, la cual se encuentra disponible en su página oficial y con ella se puede acceder a actualizaciones y demás, o se puede optar por su oferta comercial la cual otorga ciertos beneficios en soporte y participación en foros. Cualquiera puede instalar este producto sin costo alguno en sus redes empresariales.

El proceso de instalación y configuración de *Snort* puede ser bastante extenso, como lo es el objetivo del proyecto describir a detalle el manejo de un *IDPS* se describen los pasos más importantes.

7.3.1 Compilación e Instalación

a. Se debe instalar las dependencias que necesita *Snort* para su correcto funcionamiento. Esto se puede hacer con el siguiente comando.

```
apt-get install flex bison build-essential checkinstall
libpcap-dev libnet1-dev libpcre3-dev libnetfilter-queue-dev
iptables-dev libdumbnet-dev zlib1g-dev
```

[144] SNORT. Snort Documents [online]. Snort official page, ene., 2018. [citado 1 abr., 2018]. Disponible en Internet: < https://www.snort.org/documents >

Luego de ello se debe crear una carpeta donde se pueda compilar el proyecto. En *Linux* cuando se desea compilar algún proyecto desde cero se recomienda usar la ruta */usr/src/snort_src*[145]*.*

```
mkdir -p /usr/src/snort_src
cd /usr/src/snort_src
```

b. En la página oficial de *Snort*[146] se pueden encontrar dos proyectos importantes, el primero de ellos es *DAQ* quien se encarga del procesamiento de las reglas, el segundo es el proyecto principal de *Snort* encargado de monitorear las interfaces de red. En la sección *Source* se descargan los dos archivos con extensión *tar.gz* más recientes. Se deben guardar en la carpeta creada anteriormente.

```
wget -O /usr/src/snort_src
https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz

wget -O /usr/src/snort_src
https://www.snort.org/downloads/snort/snort-2.9.11.1.tar.gz
```

Se descomprimen ambos archivos con el comando *tar.*

```
tar xvzf daq-2.0.6.tar.gz && tar xvzf snort-2.9.11.1.tar.gz
```

c. El primer proyecto a compilar es *DAQ* ya que es prerrequisito para instalar *Snort.* Por ello se ejecutan los siguientes comandos ubicados en la carpeta donde se ha descomprimido el proyecto.

```
./configure; make; make install
```

Este comando configurará, compilará e instalará el proyecto respectivamente.

d. Luego es necesario dirigirse hacia la raíz del proyecto *Snort,* la carpeta creada al descomprimir las fuentes, en ella se ejecutan una serie de comandos similares al anterior; aunque varían algunos *flags* para el proyecto.

```
./configure --enable-non-ether-decoders --enable-sourcefire;
make; make install
```

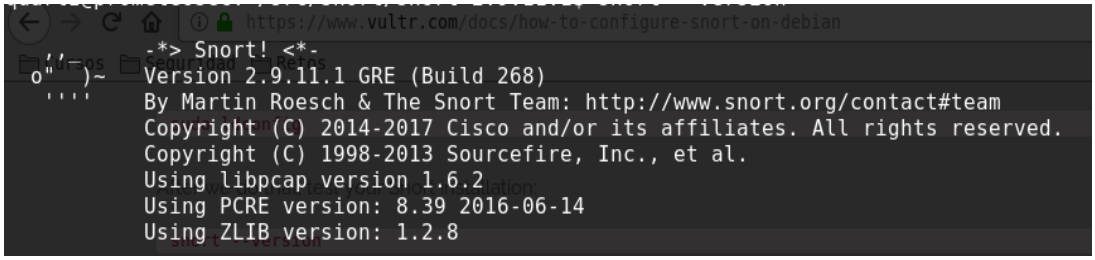
[145] VULTR. How to Configure Snort on Debian [online]. Vultr Docs, ene 2016. [citado 1 abr., 2018]. Disponible en Internet: < https://www.vultr.com/docs/how-to-configure-snort-on-debian >

[146] SNORT. Snort Documents [online]. Snort downloads, ene., 2018. [citado 1 abr., 2018]. Disponible en Internet: < https://www.snort.org/downloads >

La opción *--enable-non-ethet-decoders* proporciona al proyecto compilado la capacidad de monitorear otro tipo de interfaces de red que no sean específicamente de *Ethernet.* Algo útil para el proyecto ya que el sistema operativo en un *Debian* sobre *OpenVZ.* Los demás comandos compilan e instalan el proyecto. Este proceso puede requerir algunos minutos.

e. Con el comando *snort --version* se puede comprobar la versión que se ha instalado de *Snort.* Además, esto mostrará si la compilación e instalación han sido correctas. La Figura 47 define el resultado de la verificación.

Figura 47 Verificación de Compilación e Instalación de Snort



Fuente: El Autor.

f. Para prevenir posibles vectores de amenaza que intenten explotar el entorno del proceso de *Snort* es bastante recomendado ejecutarlo como un usuario que no sea *root.* Es por ello que se requiere realizar la configuración de un nuevo usuario, el cual no tenga habilitado el *login* pero que pueda ofrecer permisos sobre los directorios de *snort.*

```
# Se crea el usuario snort

groupadd snort
useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort

#Se crean los directorios para la configuración de snort

mkdir /etc/snort
mkdir /etc/snort/rules
mkdir /etc/snort/preproc_rules
touch                          /etc/snort/rules/white_list.rules
/etc/snort/rules/black_list.rules
/etc/snort/rules/local.rules

#Se crea el directorio para el logging de snort
```

```
mkdir /var/log/snort

 #Se crea el directorio donde snort puede añadir reglas
dinámicas

mkdir /usr/local/bin/snort_dynamicrules

#Se asignan los permisos al usuario snort sobre los directorios
creados

chmod -R 5775 /etc/snort
chmod -R 5775 /var/log/snort
chmod -R 5775 /usr/local/lib/snort_dynamicrules
chown -R snort:snort /etc/snort
chown -R snort:snort /var/log/snort
chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

7.3.2 Configuración Inicial

a. Una vez se ha configurado el esquema de directorios y usuarios requeridos por *Snort* se procede a copiar los archivos de configuración de ejemplo, los cuales vienen con el código fuente de *Snort.* Se debe ubicar la carpeta *etc* del código fuente y ejecutar los siguientes comandos.

```
cp /usr/src/snort_src/snort*/etc/*.conf* /etc/snort
cp /usr/src/snort_src/snort*/etc/*.map /etc/snort
/usr/src/snort_src/snort*/etc/*.config /etc/snort
```

b. El archivo principal de configuración de *Snort* se encuentra en la ruta */etc/snort/snort.conf.* Es necesario configurar varios aspectos en este archivo, en primer lugar, verificar las redes *HOME_NET* y *EXTERNAL_NET*[147] las cuales deben ser coherentes a la red interna y la red externa correspondientemente. En este caso ambas se dejan *any* ya que no se tiene conocimiento de las redes internas del *VPS.*

```
ipvar HOME_NET any
ipvar EXTERNAL_NET any
```

Luego es importante configurar las rutas donde se encuentran las reglas que tomará *Snort,* por ello se debe colocar las siguientes líneas en el archivo de

[147] VULTR. Op. cit., p. 180.

configuración. Es importante verificar que ninguna de estas líneas esté duplicada.

```
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

c. Se debe comentar todos los archivos de reglas ubicados al final del fichero, excepto la siguiente línea.

```
include $RUEL_PATH/local.rules
```

Luego se crea el archivo *local.rules* en la ruta */etc/snort/rules.* Con el siguiente contenido.

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)
```

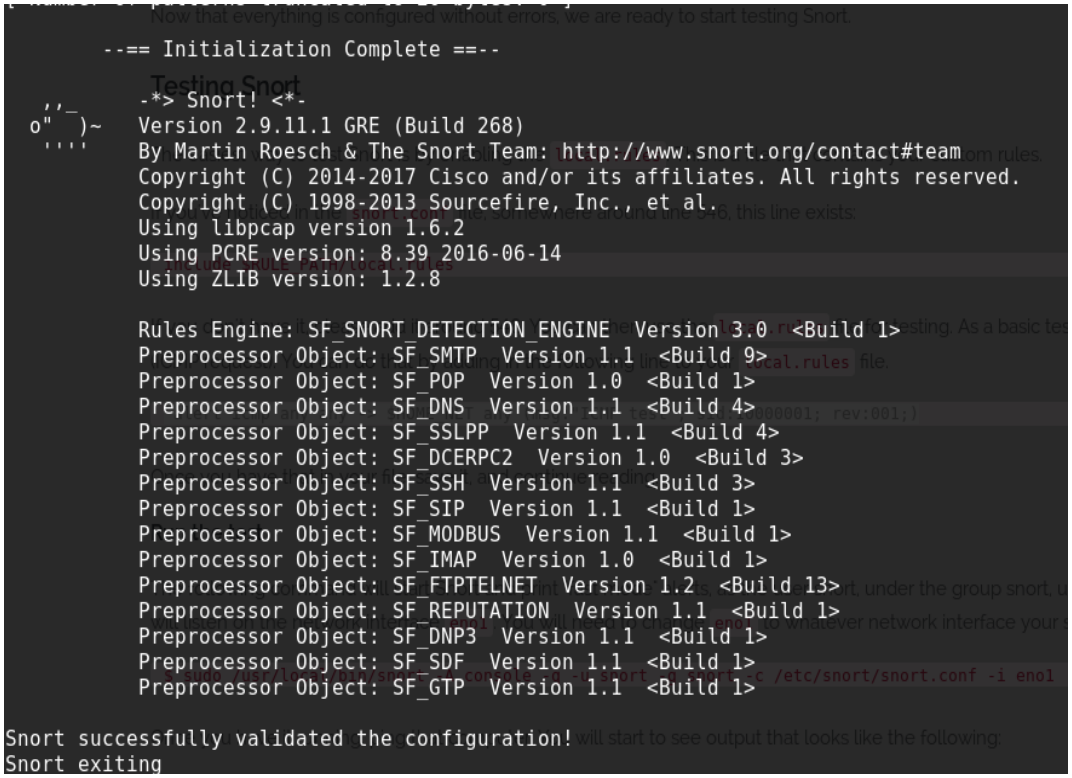
Esto indica a *Snort* que alerte cualquier mensaje de *ICMP* entrante desde la red *HOME_NET,* en este caso cualquier red.

d. El comando *snort –T –c* puede ayudar a depurar el archivo de configuración. Este comando es bastante importante para identificar errores de sintaxis en dicho archivo.

```
snort –T –c /etc/snort/snort.conf
```

La salida del comando anterior se observa en la Figura 48.

Figura 48 Comprobación de Archivo de Configuración con Snort



Fuente: El Autor.

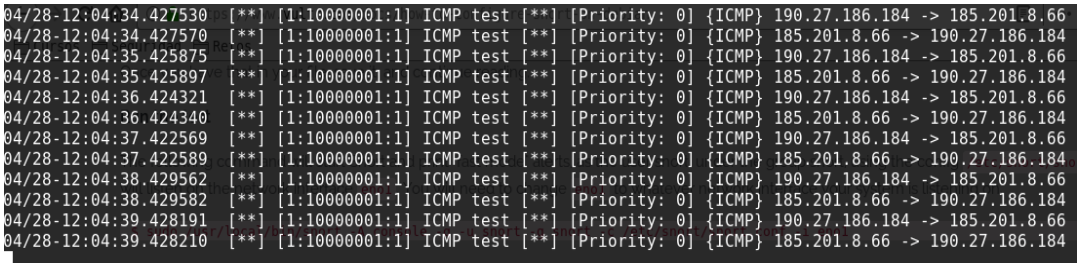
Si la configuración ha sido exitosa se debe visualizar el mensaje *Snort successfully validated the configuration.*

e. Para comprobar que hasta el momento se está ejecutando correctamente *Snort,* se ejecuta el modo de consola con el siguiente comando. Esto imprimirá los mensajes de alerta en la consola.

```
snort -A console -q -u snort -g snort -c /etc/snort/snort.conf
-i venet0:0
```

Desde cualquier otra máquina se debe realizar *ping* hacia el *host* para comprobar si se muestran las alertas sobre el mensaje *ICMP.* La Figura 49 demuestra el resultado de la prueba anterior.

Figura 49 Alertas generadas por Snort sobre los mensajes ICMP



Fuente: El Autor

.

Si se puede visualizar estas alertas, significa que la configuración inicial de *Snort* ha sido exitosa.

7.3.3 Descarga y Configuración de Reglas Oficiales

a. Para descargar las reglas de *Snort* es necesario tener configurada una cuenta de usuario en la comunidad. Luego de ello en la página https://www.snort.org/downloads#rules se pueden observar las reglas, cada una de ellas en paquetes de *tar.gz.* En este caso se dirige a la sección *Snort v 2.9* y se descarga el *Snapshot* más reciente el cual es snortrules-snapshot-29111.tar.gz.

b. Hecho esto se procede a descomprimir el archivo descargado con el siguiente comando.

```
tar xvzf snortrules-snapshot-29111.tar.gz
```

Como se puede observar las reglas están dispuestas en una estructura de directorio bastante similar a la creada anteriormente, estos son los directorios */preproc_rules, rules* y *so_rules.* Además, el paquete viene con un directorio */etc* el cual puede contener modificaciones en la configuración de *Snort.*

c. Se procede a la copia de las reglas en los directorios correspondientes.

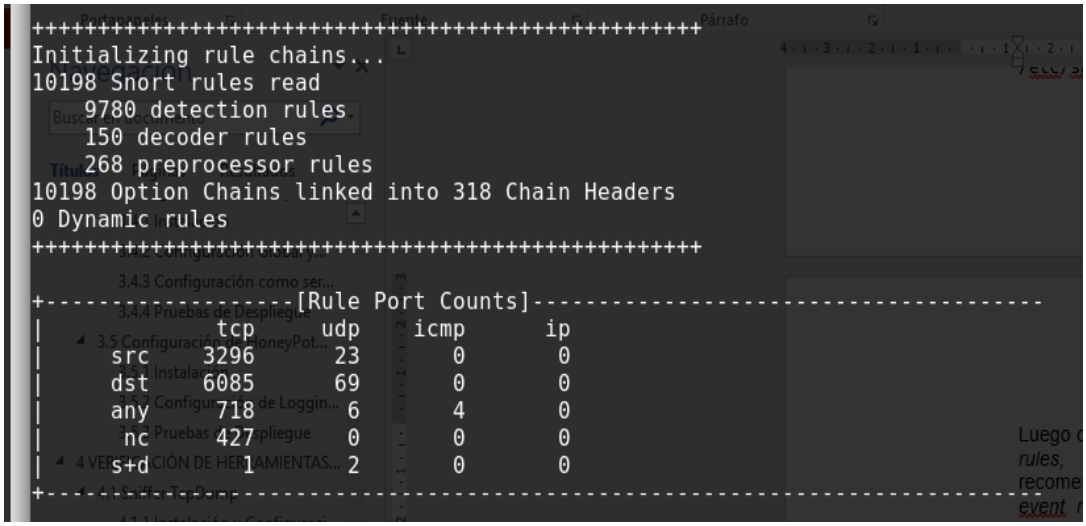
```
cp rules/* /etc/snort/rules
cp so_rules/* /etc/snort/so_rules
cp preproc_rules/* /etc/snort/preproc_rules
#Reglas personalizadas por S.O en este caso es Debian-8
cp            so_rules/precompiled/Debian-8/x86-64/2.9.11.1/*
/etc/snort/preproc_rules
```

Luego de ello se debe quitar comentarios a las líneas de la sección *specific rules,* los cuales se habían comentado inicialmente. También es recomendable quitar comentarios a las líneas *decoder and preprocessor event rules.* Finalmente se puede comprobar que las reglas han sido cargadas de forma exitosa con el siguiente comando.

```
snort –T –c /etc/snort/snort.conf
```

En la salida se debe observar un cuadro que determine cuántas reglas se han cargado, además de indicar al final el logo de Snort con el mensaje “Snort succesfully validated the configuration”. Esta verificación se ve relejada en la Figura 50.

Figura 50 Cuadro con el Resumen de las Reglas cargadas por Snort



Fuente: El Autor.

7.3.4 Configuración de Snort en Systemd

a. Se puede generar un archivo *snort.service* con el siguiente contenido.

```
[Unit]
Description=Snort NIDS
After=syslog.target network.target

[Service]
Type=simple
```

```
Environment=SNORT_CONFIG=/etc/snort/snort.conf
Environment=SNORT_IFACE=venet0:0
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c
${SNORT_CONFIG} -i ${SNORT_IFACE}

[Install]
WantedBy=multi-user.target
```

Dicho archivo se debe colocar en la ruta */etc/systemd/system.* Ahora se puede usar *systemctl* para iniciar y detener el servicio.

7.3.5 Verificación de Snort IDPS

a. Para verificar el correcto funcionamiento de *Snort* se puede explotar una de las vulnerabilidades de alguna de los *HoneyPots.* En este caso se intenta hacer un *RFI* de *Glastopf,* por medio de la siguiente ruta.

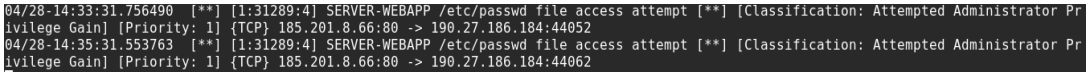
   http://185.201.8.66/BetaBlockModules/PopularTagsModule/PopularTagsModule.php?path_prefix=../../../../../etc/passwd

   Antes de acceder a la página es necesario ejecutar *Snort.*

   ```
   snort -A console -q -u snort -g snort -c /etc/snort/snort.conf
   -i venet0:0
   ```

   Luego de acceder a la página se puede observar en la consola que ha existido un intento de ganar privilegios como administrador, como se aprecia en la Figura 51.

Figura 51 Alerta por intento de ganar privilegios como Administrador Snort



Fuente: El Autor.

b. En el archivo */var/log/snort/snort.log* los eventos generados por *Snort* en el formato *Unified2.* Para leerlo es necesario instalar la herramienta *IDSTools,* la cual es una librería en *Python.*

   pip install idstools

Luego se genera el siguiente *Script.*

```
from idstools import unified2

reader = unified2.SpoolRecordReader("/var/log/snort",
    "snort.log", follow=True)
for record in reader:
    if isinstance(record, unified2.Event):
        print("Event:")
    elif isinstance(record, unified2.Packet):
        print("Packet:")
    elif isinstance(record, unified2.ExtraData):
        print("Extra-Data:")
    print(record)
```

Finalmente se observa en la salida cuáles han sido los paquetes, eventos y datos extra que guarda el log en *Unified2.* Esto se puede apreciar en la Figura 52.

Figura 52 Salida del archivo /var/log/snort/snort.log

Fuente: El Autor.

8 ATAQUES DETECTADOS A TRAVÉS DE LOS HONEYPOTS A LOS SERVICIOS DE LOS PROTOCOLOS HTTP, LDAP, SSH Y SMTP

Una vez recolectada la información es necesario analizar los datos, para ello se revisan las salidas de las herramientas de monitorización y los *HoneyPots.* De esta forma, teniendo en cuenta las amenazas a las cuales están expuestos cada uno de los Servicios; se aplica una sistematización breve para así poder concluir qué acciones se pueden llevar a cabo para prevenirlas.

Ya que se está hablando de una gran cantidad de datos como insumo, es necesario utilizar otro tipo de herramientas de analítica para discriminar comportamientos y observar las diferentes amenazas.

8.1 RESULTADOS GENERALES DE HERRAMIENTAS DE MONITOREO

8.1.1 Snort IDS

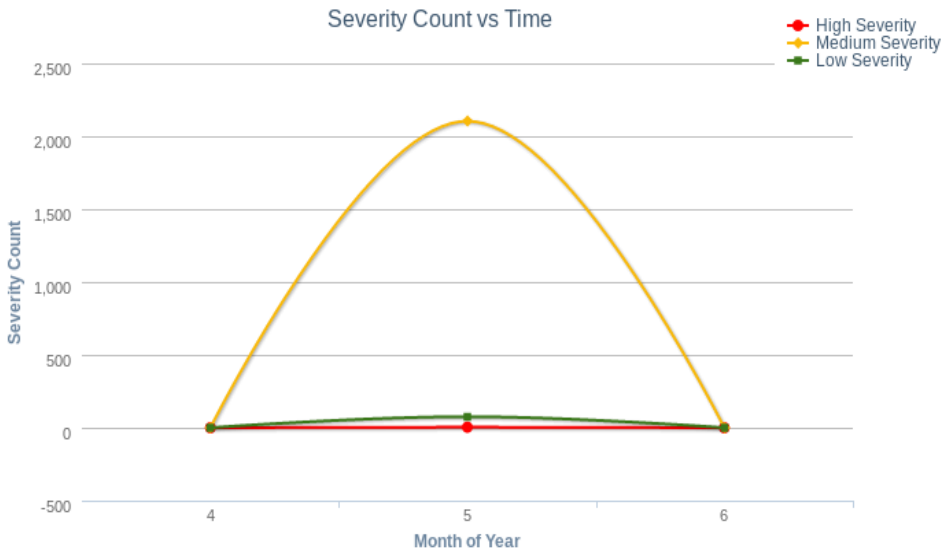
En un total se identifican 4 eventos diferentes de severidad alta, 2104 de severidad media y 76 de severidad baja. Hay que tener en cuenta que un mismo evento pudo haber sido explotado en varias ocasiones. La Figura 53 refleja estos resultados.

Figura 53 Resumen General de Eventos identificados por Snort

Fuente: El Autor.

Evidentemente, debido al periodo de exposición de los *HoneyPots* y el servidor, todos estos eventos ocurrieron entre el mes de abril y mayo del presente año, como se ve en la Figura 54.

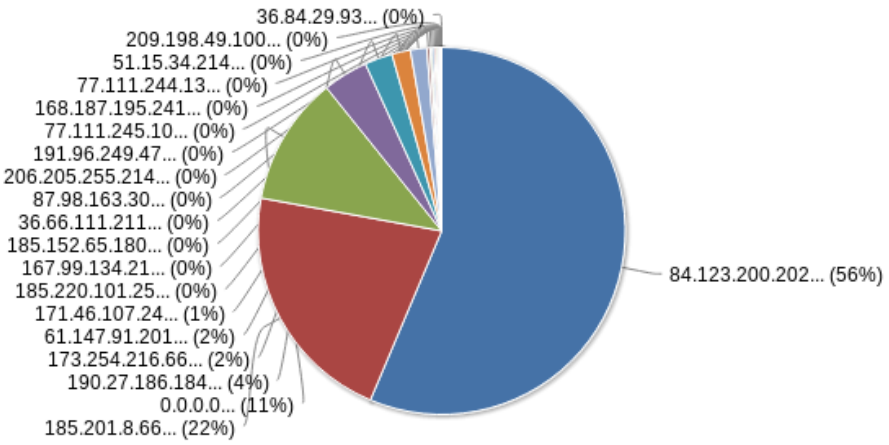
Figura 54 Número de eventos según su nivel de severidad vs Tiempo de Exposición



Fuente: El Autor.

Según el Sistema de monitoreo existió una clara tendencia sobre el origen que generó los diferentes eventos, el origen más destacado para la generación de tráfico fue la *IP 84.123.200.202,* tal cual y se observa en la Figura 55.

Figura 55 Direcciones IP que generaron los diferentes eventos registrados por Snort



Fuente: El Autor.

8.1.2 TcpDump

La Figura 56 muestra las estadísticas generales de todos los paquetes capturados, se puede observar que en promedio existió un flujo de intercambio de *bytes* bastante moderado, es decir es poco probable que se hayan generado amenazas de forma masiva o se haya abusado de los recursos de la máquina.

Figura 56 Estadísticas generales de las tramas capturadas TcpDump visualizado desde Wireshark

Measurement	Captured	Displayed
Packets	1372740	755765 (55.1%)
Time span, s	740916.322	197412.055
Average pps	1.9	3.8
Average packet size, B	785.5	761.5
Bytes	1078469675	575429289 (53.4%)
Average bytes/s	1455	2914
Average bits/s	11 k	23 k

Fuente: El Autor.

La imagen expuesta en la Figura 57 expone el tráfico generado por cada uno de los protocolos, se puede observar en él que los servicios a los cuales se realizaron más peticiones fueron *SSH* y *HTTP.*

Figura 57 Estadísticas de uso de protocolos capturados por TcpDump

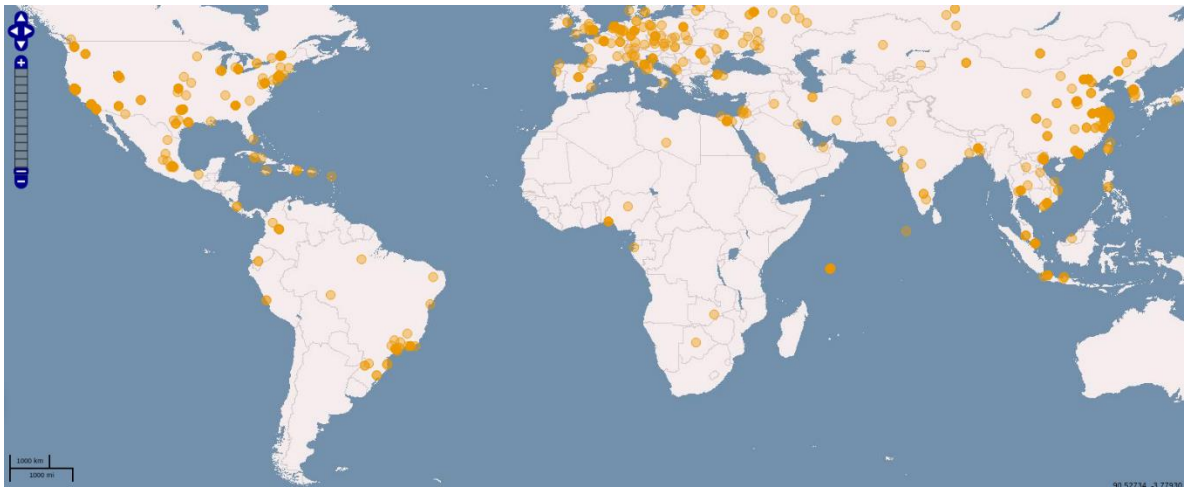
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Frame	100.0	755765	100.0	575429289	23 k	0	0
Linux cooked-mode capture	100.0	755765	2.1	12092240	490	0	0
Internet Protocol Version 6	0.4	2795	0.0	111800	4	0	0
Internet Protocol Version 4	99.6	752970	2.6	15059400	610	0	0
User Datagram Protocol	0.0	48	0.0	384	0	0	0
QUIC (Quick UDP Internet Connections)	0.0	30	0.0	3600	0	30	3600
EtherNet/IP (Industrial Protocol)	0.0	2	0.0	48	0	0	0
Malformed Packet	0.0	2	0.0	0	0	2	0
Connectionless Lightweight Directory Access Protocol	0.0	16	0.0	829	0	16	829
Transmission Control Protocol	99.6	752922	95.0	546636949	22 k	653503	465172996
SSH Protocol	1.3	9878	0.4	2123453	86	9875	2119301
Simple Mail Transfer Protocol	6.1	45765	10.1	57907303	2346	45366	57735420
Internet Message Format	0.1	399	9.9	56923487	2306	399	56923487
Malformed Packet	0.0	96	0.0	0	0	96	0
Lightweight Directory Access Protocol	0.0	5	0.0	200	0	5	200
Hypertext Transfer Protocol	5.7	42759	53.2	306031080	12 k	25574	8712970
Line-based text data	2.2	16648	51.1	294196715	11 k	16648	295771491
HTML Form URL Encoded	0.1	478	0.0	15200	0	478	15200
eXtensible Markup Language	0.0	8	0.0	7626	0	8	8898
Action Message Format	0.0	4	0.0	3432	0	0	0
Data	0.1	970	0.0	247501	10	970	247501

Fuente: El Autor.

Así mismo también se observa la predilección por el uso de *IPv4* sobre *IPv6,* como protocolo usado en la Capa de Enlace de Datos. La mayoría de tráfico generado no tuvo problemas de integridad.

Por último, se observa el origen de las conexiones en el mapa expuesto en la Figura 58. Esta imagen relaciona las direcciones *IP* con su ubicación geográfica, la herramienta que genera este gráfico es *Wireshark.*

Figura 58 Origen de las peticiones generadas a los HoneyPots Wireshark - TcpDump

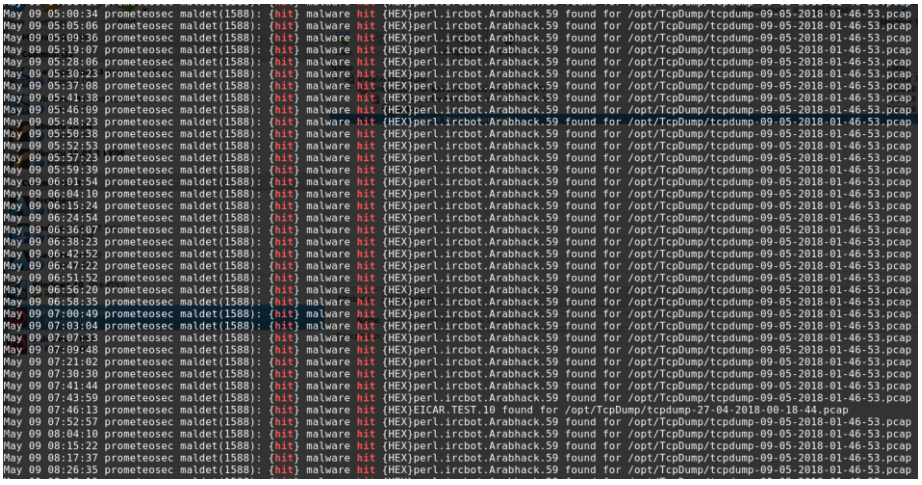


Fuente: El Autor.

8.1.3 Maldet

Maldet también detecta diferentes amenazas, en especial en los paquetes recogidos por *TcpDump* y los archivos generados por los *HoneyPots Cowrie* y *Shiva.* Sus eventos son recolectados en el archivo *event_log.* La salida de este archivo se expone en la Figura 59.

Figura 59 Visualización de event_log y amenazas detectadas por Maldet



Fuente: El Autor.

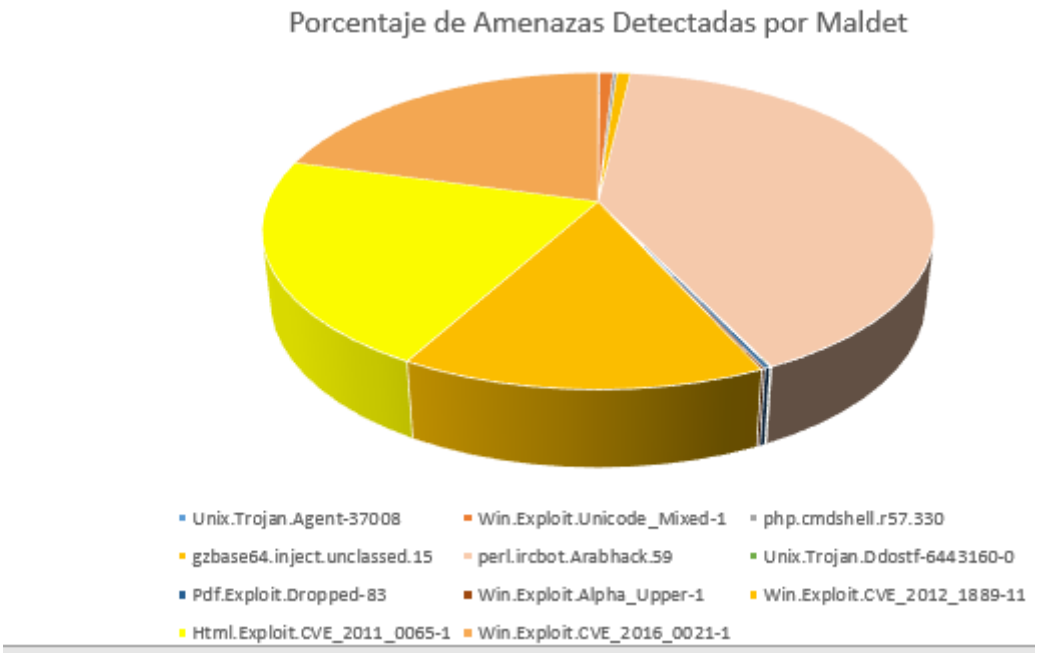
En especial se identifican dos tipos de amenazas sobresalientes los cuales son *Win.Exploit.Unicode_mixed* y *perl.ircbot.Arabhack.* La Tabla 3 relaciona el porcentaje de amenazas identificadas por *Maldet.* La Figura 60 otorga una exposición gráfica de estos datos.

Tabla 3 Porcentaje de amenazas detectadas por Maldet

Amenaza Detectada	Número de Eventos	Porcentaje
Unix.Trojan.Agent-37008	1	0,07698229
Win.Exploit.Unicode_Mixed-1	10	0,76982294
php.cmdshell.r57.330	3	0,23094688
gzbase64.inject.unclassed.15	10	0,76982294
perl.ircbot.Arabhack.59	532	40,9545804
Unix.Trojan.Ddostf-6443160-0	1	0,07698229
Pdf.Exploit.Dropped-83	3	0,23094688
Win.Exploit.Alpha_Upper-1	2	0,15396459
Win.Exploit.CVE_2012_1889-11	192	14,7806005
Html.Exploit.CVE_2011_0065-1	276	21,2471132
Win.Exploit.CVE_2016_0021-1	269	20,7082371
Total Amenazas Detectadas	1299	

Fuente: El Autor.

Figura 60 Gráfica de porcentajes de amenazas detectadas por Maldet



Fuente: El Autor.

Hay que tener en cuenta que el presente trabajo no se centra en el estudio de los virus y amenazas de este tipo, es posible que algunos de ellos sean falsos positivos.

8.2 ATAQUES HACIA HONEYPOT GLASTOPF (PROTOCOLO HTTP)

Debido que se trata de uno de los servicios más populares en el mundo de las redes, ha presentado un alto nivel de ataques. La información sobre los vectores de amenaza hacia este servicio fue recogida gracias al *HoneyPot Glastopf,* en conjunto con las herramientas *TcpDump* y *Snort.*

8.2.1 Ataques Registrados

8.2.1.1 Parameter Injection

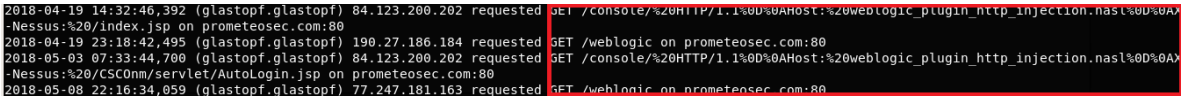
En repetidas ocasiones se realizaron peticiones de tipo *GET* las cuales pretendían inyectar código en algunos de los parámetros. Llama la atención que en la mayoría

de estos casos se asumía que el servidor web era un *JBoss,* por lo cual se intentó adjuntar código en *Java.* En una de las peticiones se puede observar que se intenta explotar la vulnerabilidad de *Web Logic* la cual apenas se dio a la luz en enero[148]. Esto se refleja en las Figura 61 y Figura 62.

Figura 61 Parameter Injection registrado por Glastopf

Fuente: El Autor.

Figura 62 Parameter Injection registrado por Glastopf – Parte 2

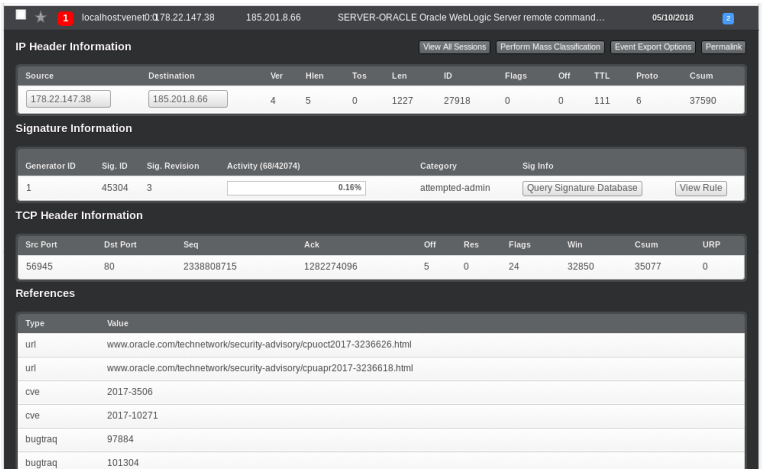


Fuente: El Autor.

Snort detectó y reportó la amenaza como una alerta grave como se ve reflejado en la Figura 63.

[148] COMMON VULNERABILITIES AND EXPOSURES CVE. Vulnerabilities Details CVE-2017-10271 [online]. Common Vulnerabilities and Exposures CVE feb., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.cvedetails.com/cve/CVE-2017-10271/ >

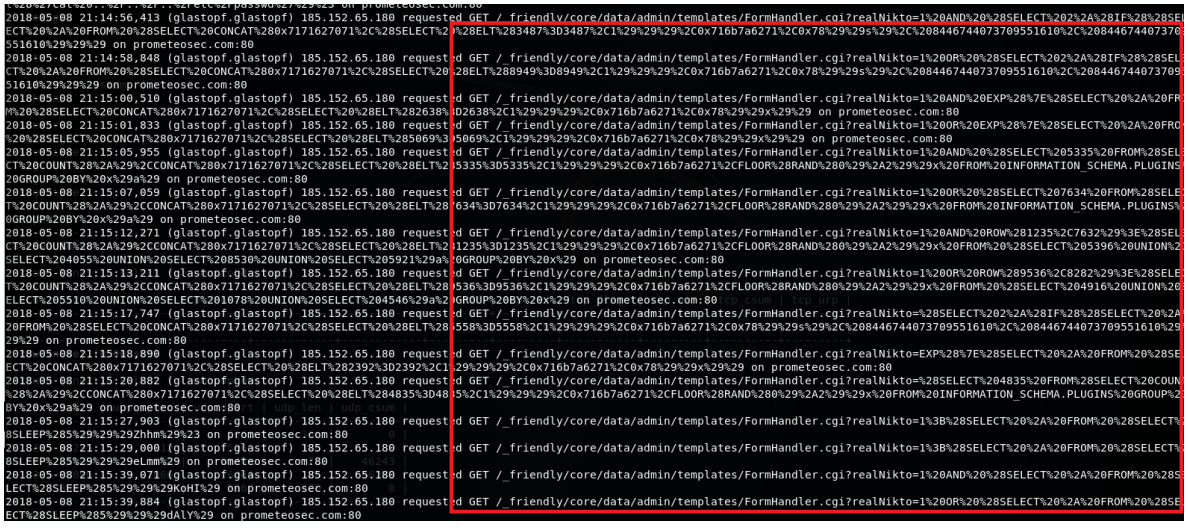
Figura 63 Snort Parameter Injection – Oracle WebLogic RCE



Fuente: El Autor.

Otro tipo de inyección se llevó a cabo por medio de *SQL Injection* (**referenciar SQL**) el cual en su mayoría fue desarrollado por medio de peticiones *GET* hacia el *HoneyPot.* En la Figura 64 se observa por ejemplo cómo se intentó inyectar el *SQL* por medio del parámetro *realNikto.*

Figura 64 Parameter Injection SQL Injection Glastopf



Fuente: El Autor.

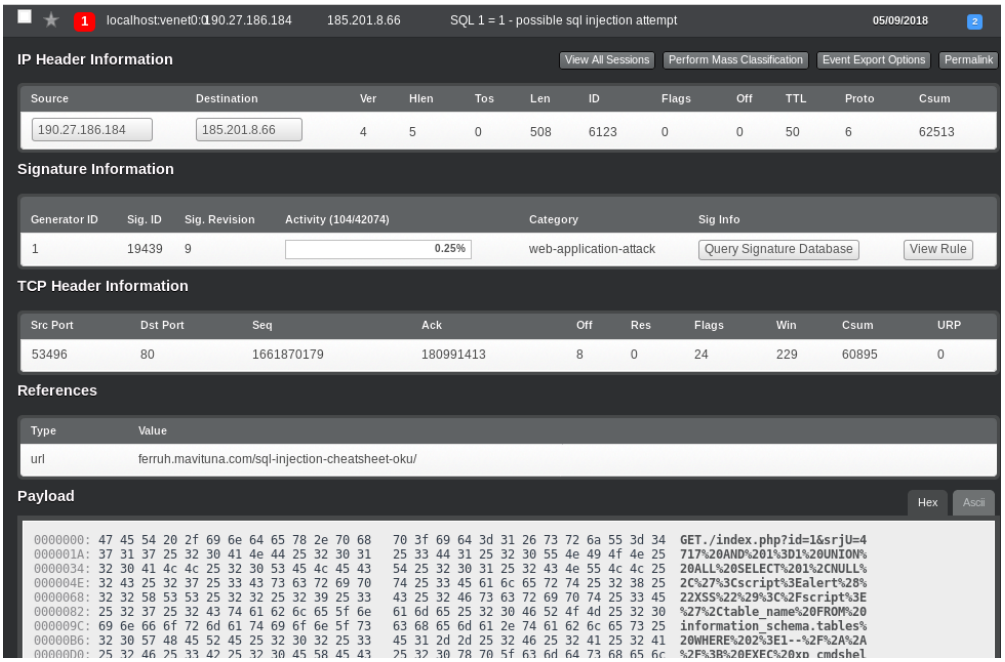
Si por ejemplo se decodifica una de las *URL* se puede observar claramente que se estaba intentando realizar la siguiente consulta.

```
(SELECT 4835 FROM(SELECT COUNT(*),CONCAT(0x7171627071,(SELECT (ELT(4835=4835,1))),0x716b7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
```

Claramente se puede observar por la referencia a *INFORMATION_SCHEMA* que se intentó reconocer si la Base de Datos era de tipo *MySQL*[149]. Por otro lado, el uso de caracteres especiales da un indicio que el sujeto o fuente del ataque conocía sobre técnicas de evasión de *Web Application Firewalls*[150] y de *IDPS.*

La amenaza también fue identificada por *Snort* como se evidencia en la Figura 65. Su clasificación fue de severidad alta.

Figura 65 Snort Parameter Injection - SQL Injection



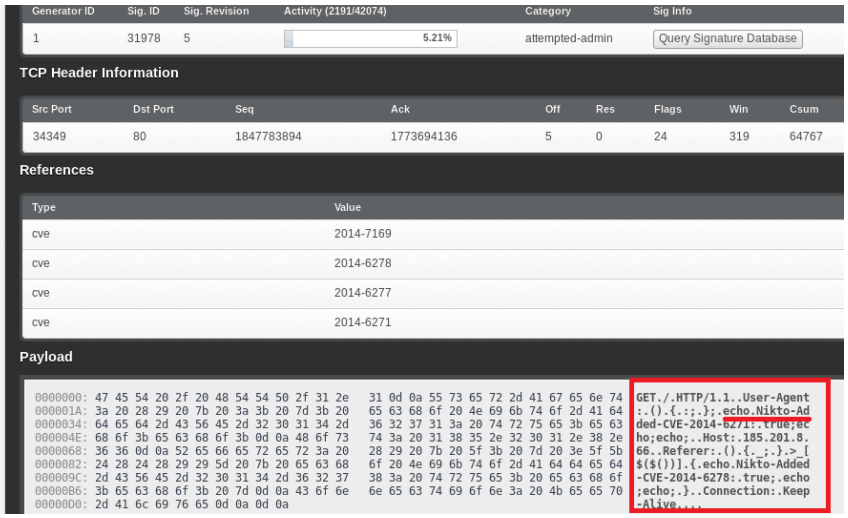
Fuente: El Autor.

Incluso se registraron ataques que iban destinados a la inyección de código en las cabeceras de las peticiones *HTTP.* Se puede ver en la Figura 66 claramente que quien desarrolla este tipo de ataque es el escáner *Nikto.*

[149] ORACLE. Chapter 21 Information_Schema Tables [online]. MySQL Development ene., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://dev.mysql.com/doc/refman/5.6/en/information-schema.html >

[150] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Web Application Firewall WAF [online]. OWASP ene, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Web_Application_Firewall >

Figura 66 Snort Parameter Injection – CGI Bash



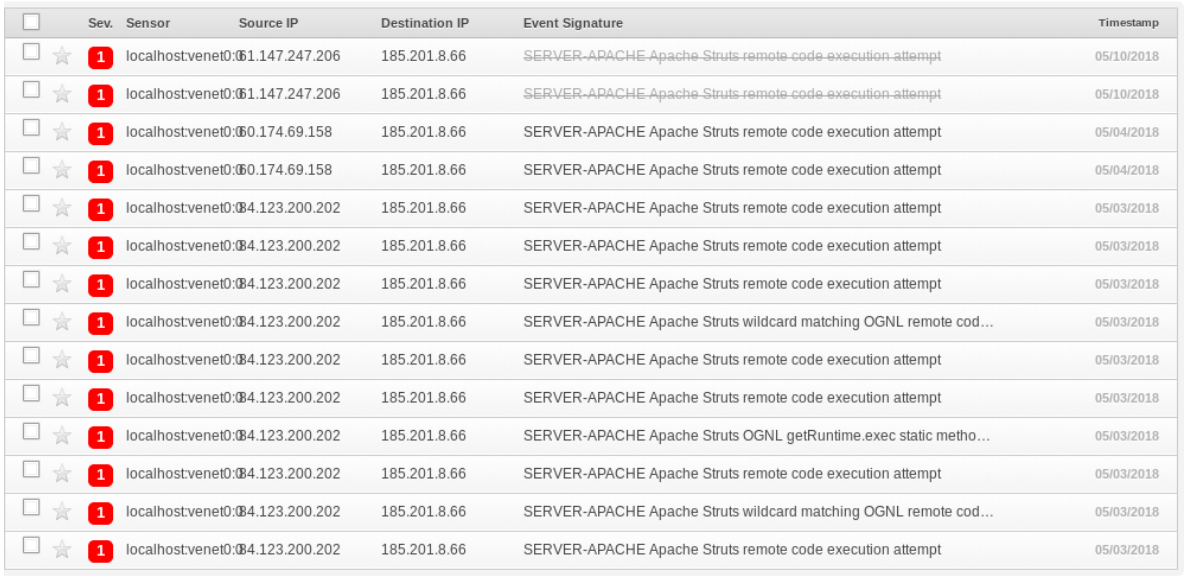
Fuente: El Autor.

En este caso *Glastopf* no registró el ataque ni en sus eventos de Bases de Datos ni tampoco en su archivo de log. Se tuvo conocimiento de este ataque gracias a *Snort,* quien lo clasifica como una amenaza grave. La vulnerabilidad que se intenta explotar también tendría relación con la Ejecución de Código Remota, *RCE.*

No se intenta explotar vulnerabilidades únicamente de servidores desarrollados en *Java,* también *Apache* no es extenso de algunos tipos de ataques que intentaban llevar a cabo un *RCE. Snort* identifica una serie de intentos para la explotación de *Apache Struts* una vulnerabilidad que lleva no más de un año desde que se ha descubierto[151]. Como se evidencia en la Figura 67.

[151] TREND MICRO. New Apache Stryts Vulnerability Could Be Worse tan Poodle [online]. Trend Micro sept., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/new-apache-struts-vulnerability-could-be-worse-than-poodle >

Figura 67 Snort Parameter Injection - Explotación de RCE para Apache



Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
1	localhost:veneto:0:0	61.147.247.206	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/10/2018
1	localhost:veneto:0:0	61.147.247.206	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/10/2018
1	localhost:veneto:0:0	60.174.69.158	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/04/2018
1	localhost:veneto:0:0	60.174.69.158	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/04/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts wildcard matching OGNL remote cod…	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts OGNL getRuntime.exec static metho…	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts wildcard matching OGNL remote cod…	05/03/2018
1	localhost:veneto:0:0	84.123.200.202	185.201.8.66	SERVER-APACHE Apache Struts remote code execution attempt	05/03/2018

Fuente: El Autor.

Un tipo de inyección bastante común en las Aplicaciones Web es el *Server Side Request Forgery SSRF.* Este consiste en realizar una petición web usando como cliente el propio servidor que se desea atacar. Las consecuencias de este tipo de ataques pueden variar desde un *RCE* hasta *Remote File Inclusion RFI.* Se observa en el log de *Glastopf,* expuesto en la Figura 68, varias referencias hacia *URLs* externas las cuales son colocadas en los parámetros de las peticiones *GET.*

Figura 68 Parameter Injection – Server Side Request Forgery SSFR Glastopf

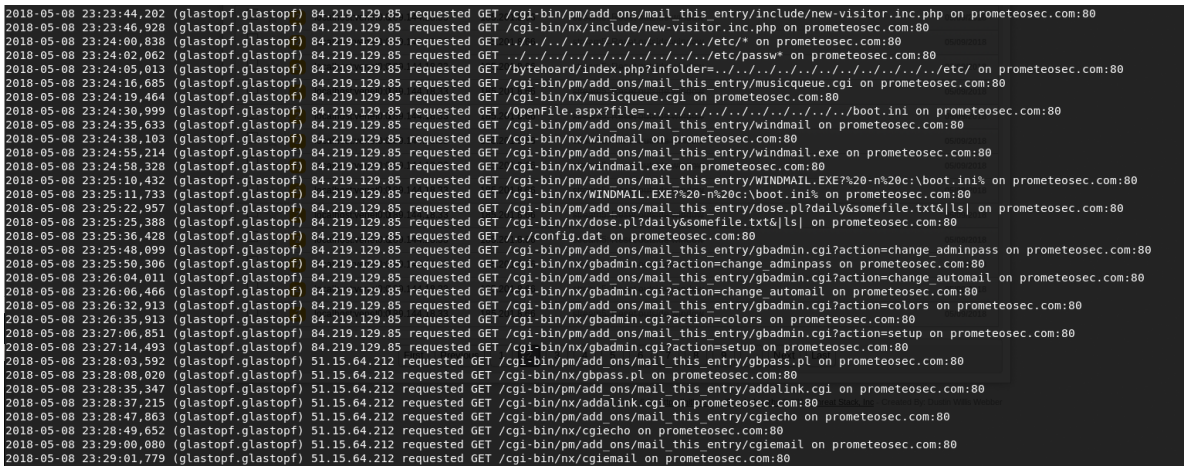
Fuente: El Autor.

8.2.1.2 Exposición de Datos Sensibles

Los sensores de *Glastopf* indican que existieron varios intentos de acceso hacia archivos locales de la máquina, a este tipo de amenazas en concreto se le conoce como *Path Transversal*[152]. Destaca el intento de acceso hacia los archivos */etc/passwd* y */etc/shadow* los cuales son muy importantes en los Sistemas Operativos Linux y Unix ya que almacenan información primordial sobre los usuarios. También se intenta acceder al archivo *boot.ini* uno de los ficheros más importantes para el arranque de Windows. Ambos ataques se ven reflejados en la Figura 69.

[152] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Path Transversal [online]. OWASP jun., 2015. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Path_Traversal >

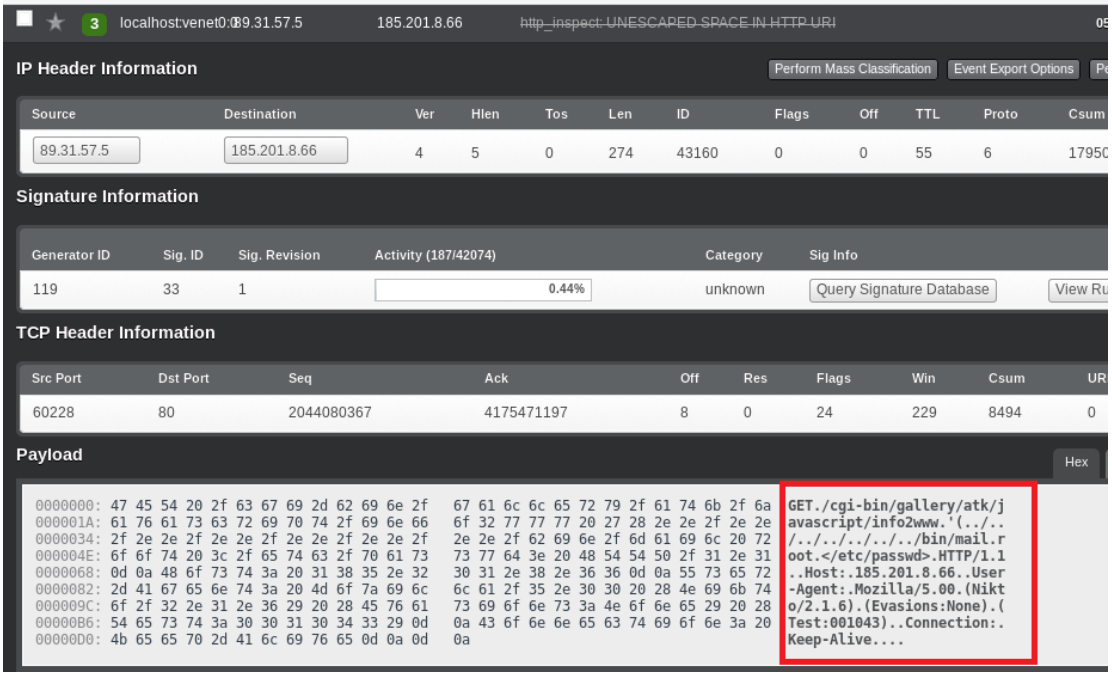
Figura 69 Exposición de Datos Sensibles – Path Transversal Glastopf



Fuente: El Autor.

Desafortunadamente muchas de estas amenazas fueron catalogadas como severidad menor por *Snort,* algo que se observa en la Figura 70. Quizás esto haya ocurrido debido a que los datos expuestos no fueron considerados como críticos.

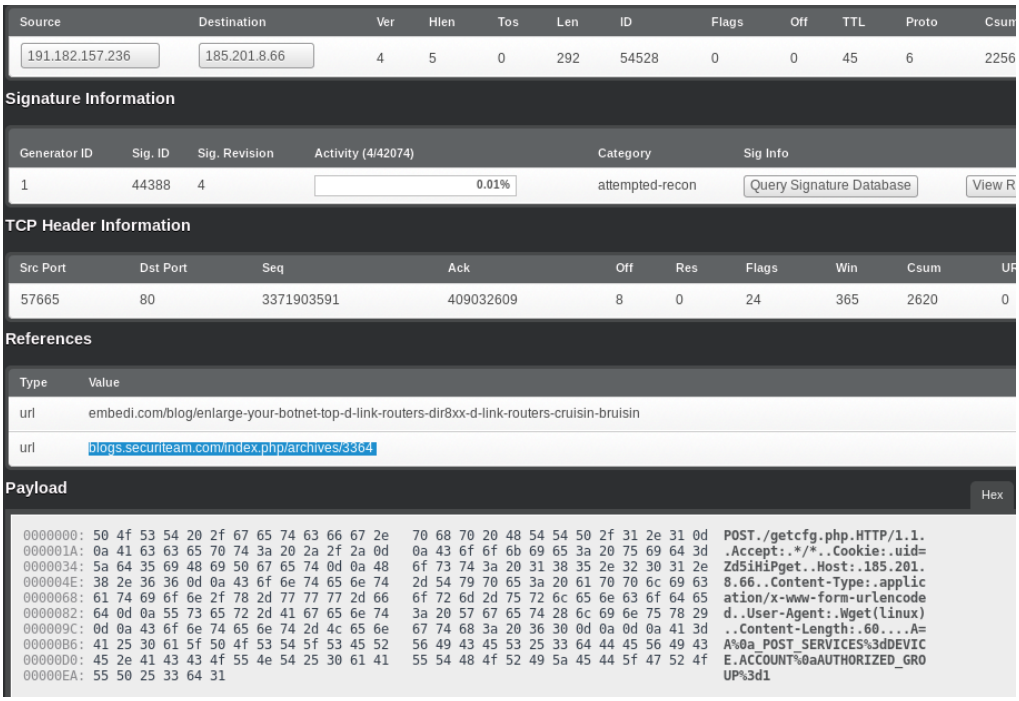
Figura 70 Snort Exposición de Datos – Path Transversal



Fuente: El Autor.

La *Exposición de Datos Sensibles* también se intenta aprovechar por medio de desarrollo de vectores de ataques más específicos. *Snort* identifica que usando el escáner *Nikto* se asume que la página web es de un router, intentando obtener usuario y contraseña. (ver blogs.securiteam.com/index.php/archives/3364). En este caso la severidad si ha sido considerada de nivel medio evidenciándose en la Figura 71.

Figura 71 Snort Exposición de Datos – Explotación de CGI de Router D-Link 850L



Fuente: El Autor.

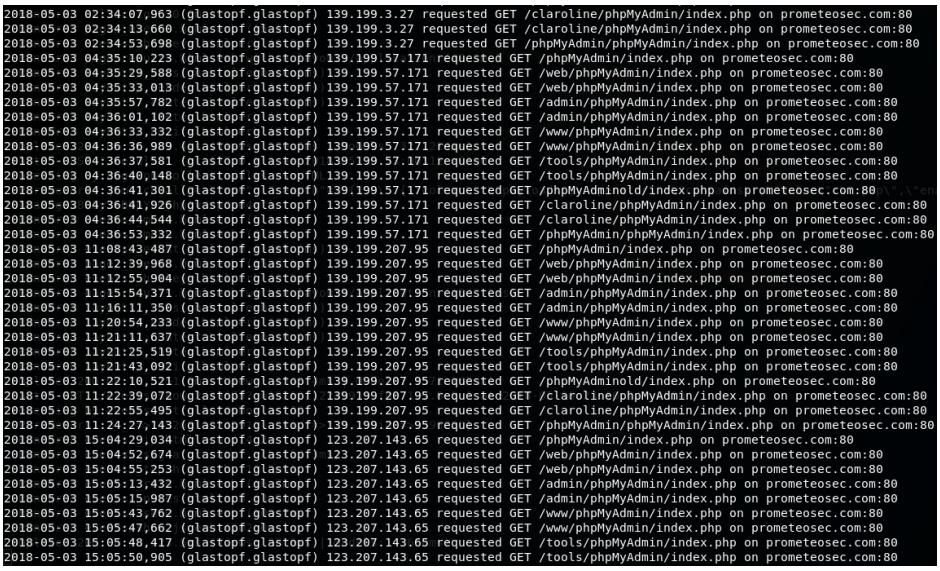
8.2.1.3 Control de Acceso Débiles

Glastopf tiene configurado por defecto el directorio de *phpMyAdmin* con la intención de verificar si existen peticiones dirigidas hacia este directorio. Hay que recordar que *phpMyAdmin* es un software bastante usado para las aplicaciones *Web* y permite la administración sencilla de las Bases de Datos *MySQL*[153]. Este directorio no ha pasado desapercibido, de hecho, ha sido uno de los que más se han

[153] PHPMYADMIN. Bringing MySQL to the Web About [online]. PhpMyAdmin oct., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.phpmyadmin.net/ >

consultado durante el periodo en que se ha desplegado el *HoneyPot,* siendo consultado más de *431* veces como se expone en la Figura 72.

Figura 72 Control de Acceso Débil – Accesos a phpMyAdmin - Glastopf

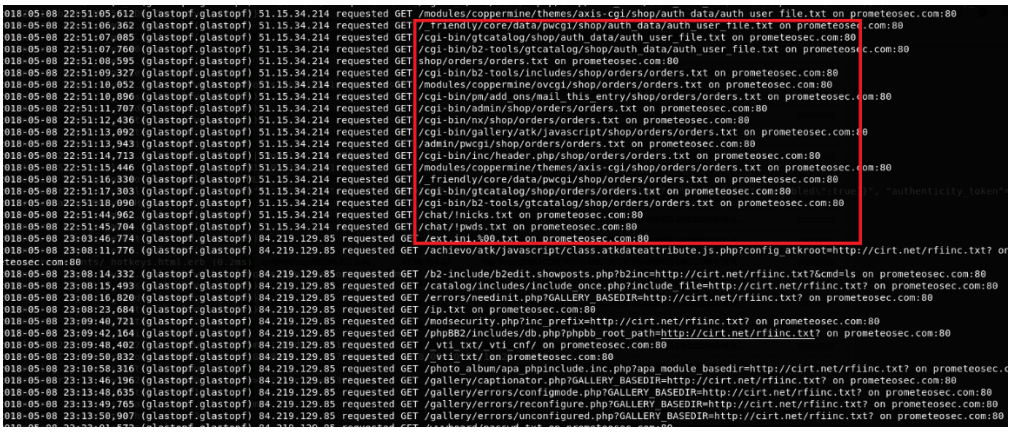


Fuente: El Autor.

Este tipo de directorios no debe ser expuesto de ninguna manera en una red pública y siempre debería ofrecer un tipo de autenticación fuerte. En este caso el *IDS Snort* no ha identificado este comportamiento como una amenaza.

En otras solicitudes se intenta obtener de manera aleatoria archivos de configuración, texto y otros recursos por medio del descubrimiento de directorios. Lo anterior se refleja en la Figura 73.

Figura 73 Control de Acceso Débil – Ingreso a Directorios de Configuración y Otros Glastopf



Fuente: El Autor.

Este tipo de ataques, dependiendo de qué información almacene; puede ir en conjunto con la categoría anterior.

8.2.1.4 Cross-Side Scripting

Se registra una gran cantidad de peticiones de *XSS* llegando casi a la suma de *2200* peticiones maliciosas. En su totalidad se presentan ataques de *XSS* de tipo reflejado y dirigido hacia el *DOM*[154], no se registran ataques de tipo almacenado ya que el mismo *HoneyPot* presenta un tipo de protección contra este ataque para no comprometer la práctica. Una muestra de ello es la Figura 74.

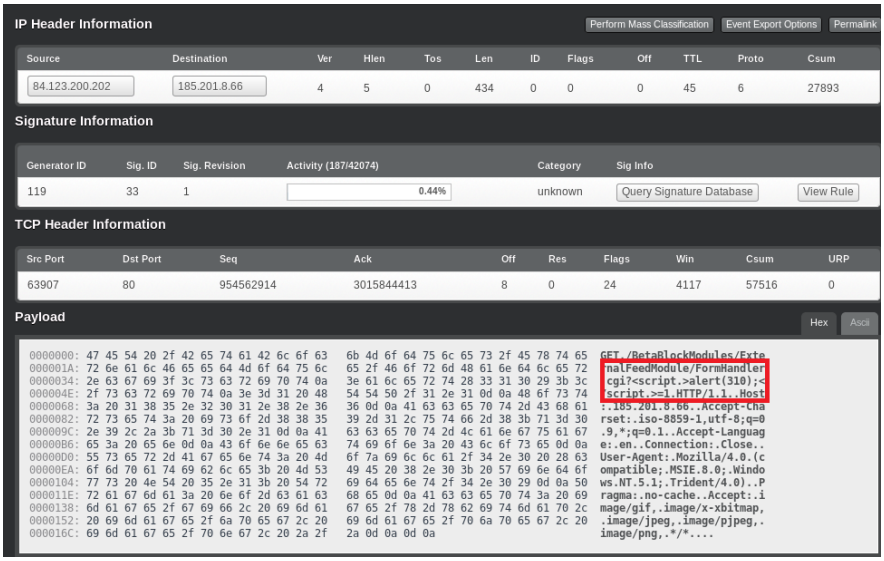
[154] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Cross-site Scripting (XSS) [online]. OWASP mar., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) >

Figura 74 Cross Side-Scripting reflejado – Glastopf

Fuente: El Autor.

Este vector de ataque fue clasificado como leve por *Snort,* la razón podría ser que no afecta directamente al servidor ya que sus consecuencias estarían del lado del cliente, a quien en realidad hace la petición de la *URL.* El evento se puede observar en la Figura 75.

Figura 75 Snort Cross-Side Scripting Reflejado



Fuente: El Autor.

8.2.1.5 Ataques de Fuerza Bruta

Un punto de enfoque para Ataques de Fuerza bruta dirigidos hacia *Glastopf* puede ser el formulario de autenticación ofrecido en la página de inicio. Escudriñando dichas peticiones en el log de *glastopf* sólo se identifica un número de *51,* algo que se puede ver en la Figura 76.

Figura 76 Ataques de Fuerza Bruta – Formulario de Autenticación Glastopf

```
77.111.247.6 requested POST /index on prometeosec.com:80
77.111.247.6 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
81.103.69.138 requested POST /index on prometeosec.com:80
66.229.228.240 requested POST /index on prometeosec.com:80
66.229.228.240 requested POST /index on prometeosec.com:80
66.229.228.240 requested POST /index on prometeosec.com:80
190.27.186.184 requested POST /index on prometeosec.com:80
181.49.1.10 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index on prometeosec.com:80
35.226.208.150 requested POST /index on prometeosec.com:80
35.188.181.252 requested POST /index on prometeosec.com:80
84.123.200.202 requested POST /index.php on prometeosec.com:80
```

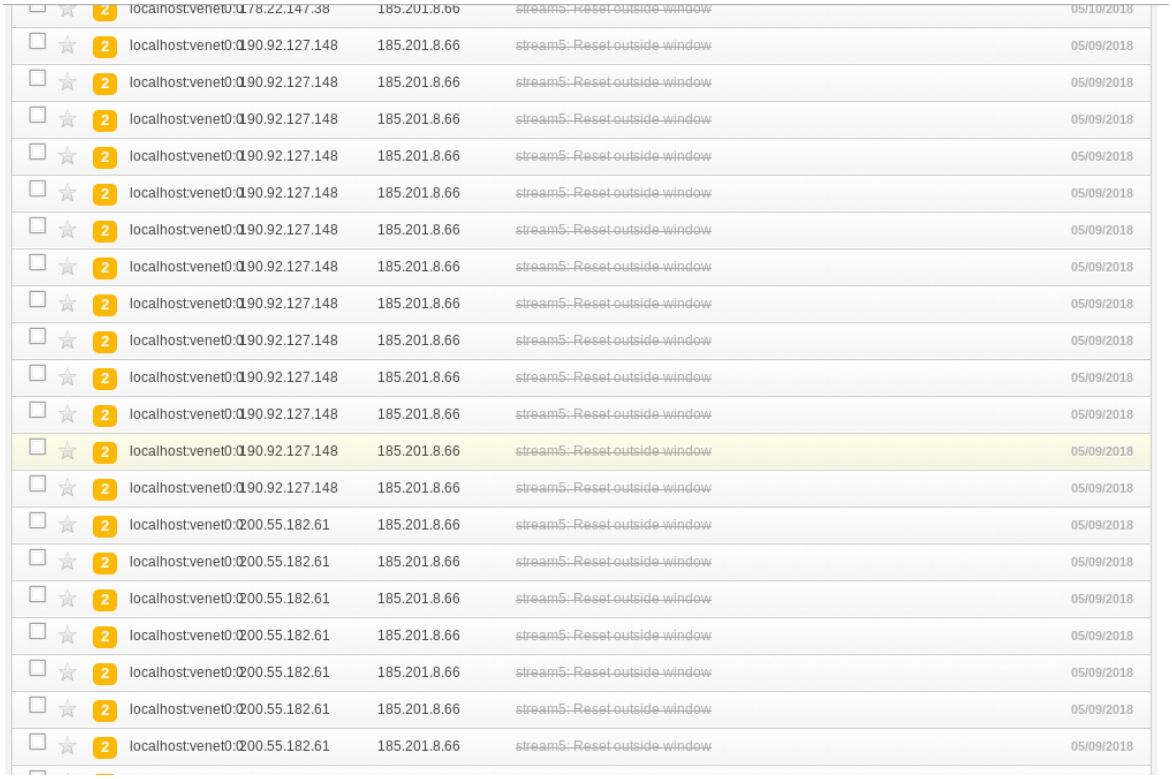
Fuente: El Autor.

8.2.1.6 Denegación de Servicio

A nivel de transporte, es decir en el protocolo *TCP* el cual es la base para el protocolo *HTTP* se identifica una serie de peticiones de tráfico incompleto o tramas corruptas. Debido a la cantidad de peticiones de este tipo, es bastante probable que se haya ejecutado un Ataque de tipo *TCP Flood.* El número de peticiones totales de este tipo es de *708,* pero muchas de ellas coinciden en un mismo día; en especial

el 9 de marzo ocurrieron muchos eventos de este tipo. La Figura 77 evidencia lo anterior.

Figura 77 Snort Denegación de Servicio – Posible Ataque TCP Flood



Fuente: El Autor.

8.2.2 Resumen y Estadísticas

La Tabla 4 muestra un número aproximado de las peticiones generadas por categoría. Se entiende por categoría el tipo de ataque desarrollado o en su defecto si la petición no tiene intención maliciosa. Se aclara que se trata de un número aproximado debido que no se descarta la presencia de amenazas desconocidas, además categorías como *Exposición de Datos Sensibles* puede ser relativa teniendo en cuenta que se trata de un *HoneyPot* el cual carece de lógica de negocio por lo que no se puede determinar exactamente cuál es la información sensible que se accede.

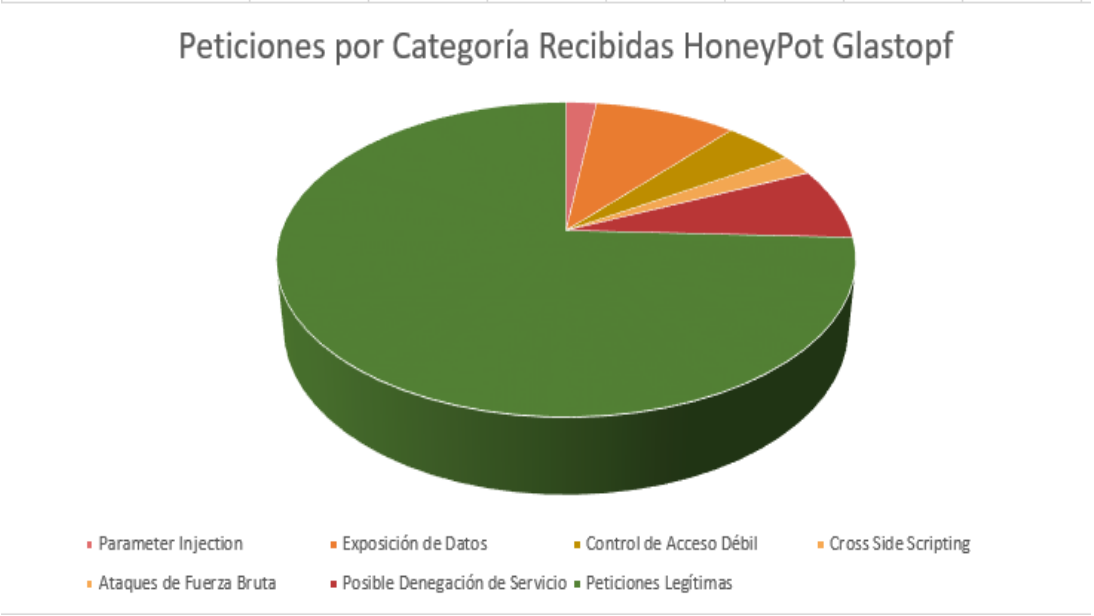
Tabla 4 Número de Peticiones Registradas por Categoría Glastopf

Categoría	Número de Registros	Porcentaje
Parameter Injection	2116	2,02536492
Exposición de Datos	9878	9,454893515
Control de Acceso Débil	4876	4,66714525
Cross Side Scripting	2252	2,155539603
Ataques de Fuerza Bruta	51	0,048815506
Posible Denegación de Servicio	7708	7,377841589
Peticiones Legítimas	77594	74,27039962
Total Peticiones HoneyPot	104475	

Fuente: El Autor.

La Figura 78 otorga de manera visual las porciones de afectación de los diferentes ataques, aunque se resalta que muchas de las peticiones fueron legítimas se puede observar que los diferentes ataques desarrollados ocupan un porcentaje que no puede ser menospreciado.

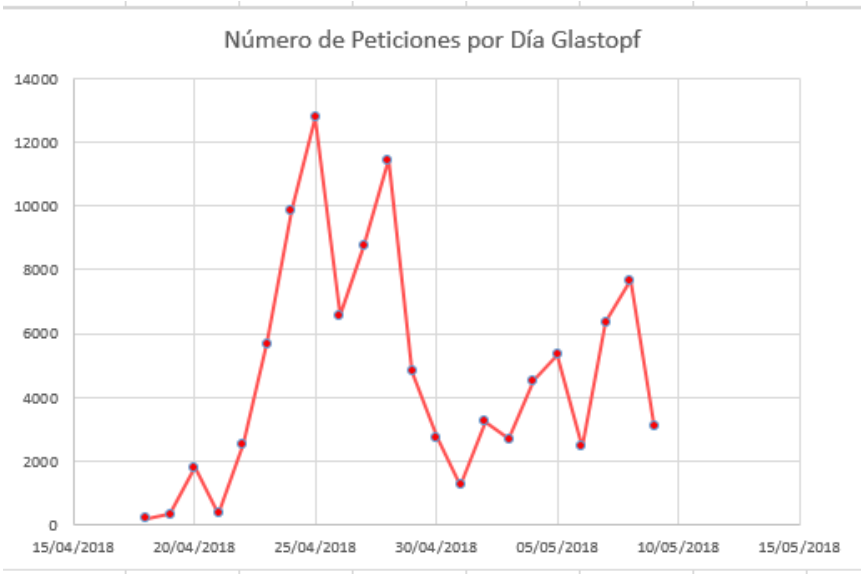
Figura 78 Porcentaje de Peticiones por Categoría Recibidas por Glastopf



Fuente: El Autor.

Por último, se expone una gráfica en la Figura 79 con el número de peticiones que fueron registrados durante el tiempo de exposición del *HoneyPot.*

Figura 79 Número de Peticiones por Día HoneyPot Glastopf



Fuente: El Autor.

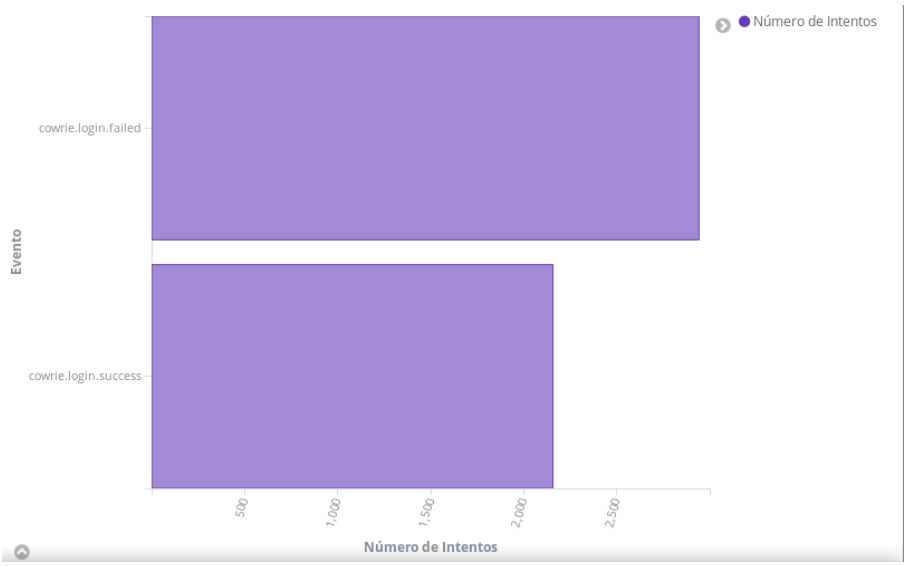
8.3 ATAQUES HACIA HONEYPOT COWRIE (PROTOCOLO SSH)

8.3.1 Ataques Registrados

8.3.1.1 Password Cracking y Ataques de Fuerza Bruta

Se identifica una alta incidencia de ataques de fuerza bruta cuyo objetivo es descubrir cuáles son los usuarios del sistema y sus respectivas credenciales de acceso. El *HoneyPot* muestra en sus archivos de *logs* un número elevado de accesos negados y accesos exitosos lo cual se evidencia en la Figura 80.

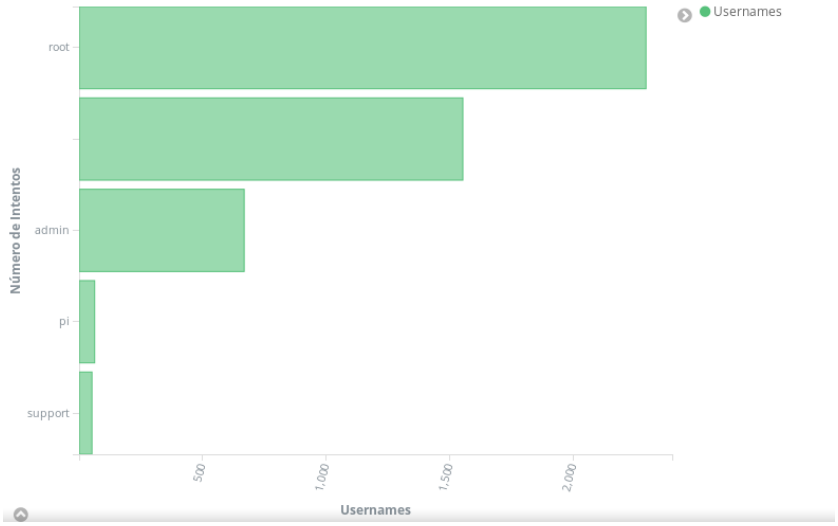
Figura 80 Número de Accesos Exitosos y Fallidos HoneyPot Cowrie



Fuente: El Autor.

De esta información se extraen cuáles son los usuarios y credenciales de acceso que se utilizaron con mayor frecuencia, esto se puede visualizar en las Figura 81 y Figura 82.

Figura 81 Número de Nombres de Usuario usados en Ataques de Fuerza Bruta HoneyPot Cowrie



Fuente: El Autor.

Figura 82 Contraseñas usadas en Ataques de Fuerza Bruta HoneyPot Cowrie

Fuente: El Autor.

8.3.1.2 Canales en Cubierto

También se registraron algunos comandos sospechosos que intentaban descargar *malware,* aplicar reconocimiento de la red o ingresar otro tipo de amenazas usando el canal *SSH.* A todas estas amenazas se les clasifica como Canales en Cubierto, ya que utilizan el mismo servicio *SSH* y sus capacidades criptográficas para no generar mucho ruido en una red organizacional. Cabe aclarar que la evidencia de la ejecución de este tipo de amenazas puede alojarse en el mismo *host* víctima. Como comandos sospechosos existe un gran número de intentos de reconocimiento.

```
uname -a;unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG
WATCH;history -n;export HISTFILE=/dev/null;export HISTSIZE=0;export
HISTFILESIZE=0;killall -9 top htop ps s f [atd] sync_users perl;cd
/var/tmp/;cd   /tmp/;chattr   -uais   *;rm   -rf   y.txt;wget   -q
http://203.146.208.208/drago/images/.ssh/y.txt;lwp-download
http://203.146.208.208/drago/images/.ssh/y.txt;fetch
http://203.146.208.208/drago/images/.ssh/y.txt;curl                -O
```

```
http://203.146.208.208/drago/images/.ssh/y.txt;perl          y.txt
162.243.233.156;perl y.txt 162.243.233.156;rm -rf y.txt y.txt*
```

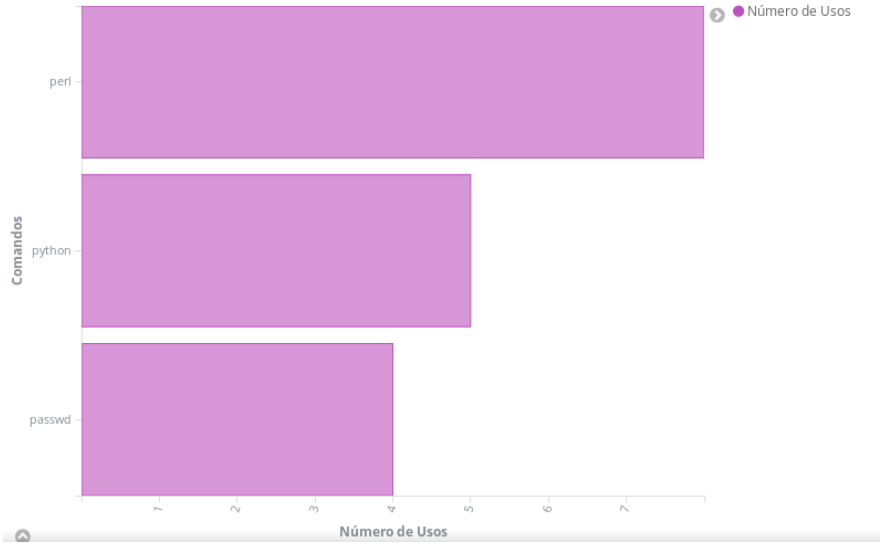
Este comando, sumamente peligroso, ejecuta un reconocimiento sobre la máquina víctima, posteriormente procede a eliminar cualquier herramienta de monitoreo de procesos he intenta descargar en repetidas veces un *script* escrito en *PERL* el cual contiene las instrucciones necesarias para agregar al *host* a una *botnet* dirigida por medio del protocolo *HTTP.* Por seguridad el *script* no es adjuntado en el documento.

```
echo                                                          -ne
'\\x6a\\x01\\xe8\\x3b\\xfe\\xff\\xff\\xc7\\x04\\x24\\x05\\x00\\x00
\\x00\\xe8\\xc9\\xfd\\xff\\xff\\x83\\xc4\\x10\\x8d\\x65\\xf4\\x5b\
\\x5e\\x5f\\x5d\\xc3\\x55\\x89\\xe5\\x5d\\xe9\\x72\\xfe\\xff\\xff\\
x90\\x55\\x57\\x56\\x53\\x8b\\x6c\\x24\\x2c\\x8b\\x7c\\x24\\x28\\x
8b\\x74\\x24\\x24\\x8b\\x54\\x24\\x20\\x8b\\x4c\\x24\\x1c'       >>
usb_ver; /gweerwe323f
```

En el segundo ejemplar se observa cómo se intenta escribir *shellcoding* en un *archivo* llamado *usb_ver,* aparentemente este comportamiento intenta explotar varias características de los Sistemas basados en *Unix* y es probable que se trate de una *botnet.*

Otros comandos como *cat /etc/passwd, cat /etc/lsb_release, cat /etc/shadow* fueron usados para un reconocimiento inicial de la máquina. En resumen, la Figura 83 muestra los tipos de comandos más utilizados cuyo nivel de peligrosidad es bastante elevado en caso de usarse para acciones maliciosas.

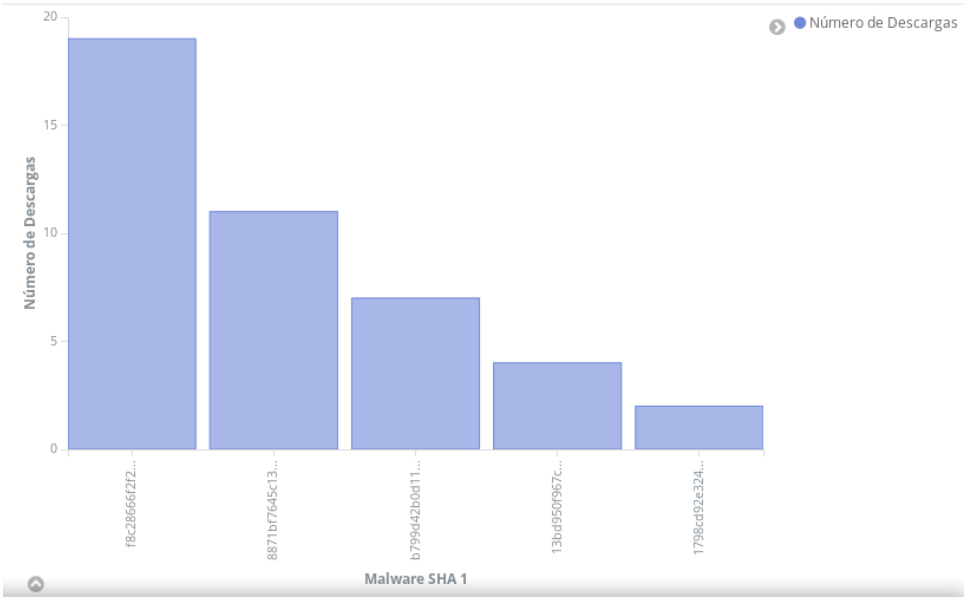
Figura 83 Comandos con nivel de alta peligrosidad HoneyPot Cowrie



Fuente: El Autor.

El problema con este tipo de comandos es, como se puede observar, el uso de diferentes intérpretes y compiladores de lenguajes de programación. Estos intérpretes o compiladores pueden ser usados para ejecutar programas descargados, este tipo de movimientos puede dificultar un análisis forense ya que se ejecutan varias *backdoors* que re dirigen así mismo a otro tipo de *backdoors.* El *HoneyPot* también ofrece datos estadísticos sobre la descarga de archivos sospechosos. Algunos archivos que se muestran en la Figura 84 también fueron identificados por *Maldet* como se puede observar en la sección *Resultados generales de Herramientas de Monitorización.*

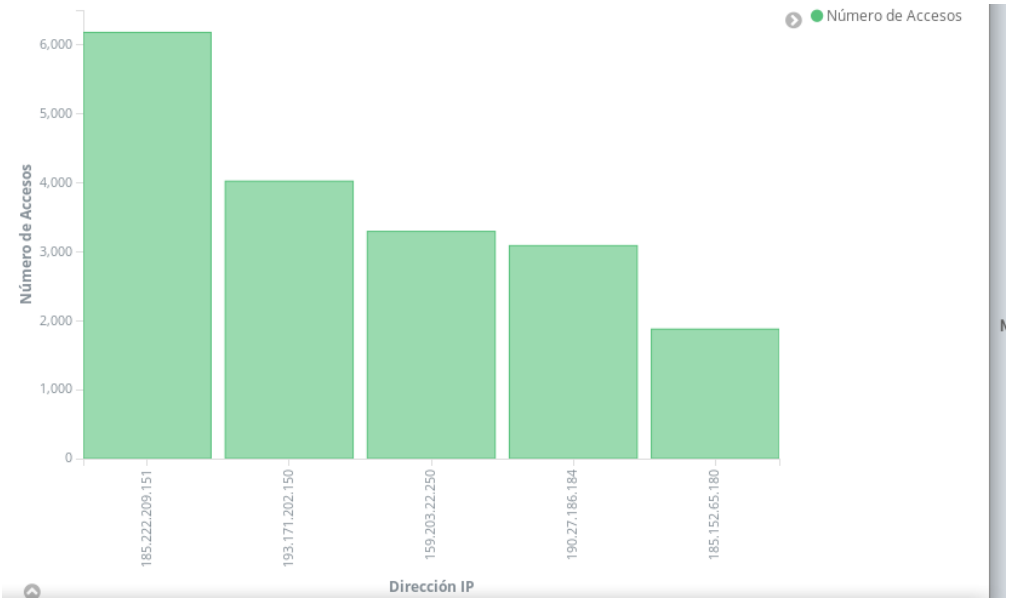
Figura 84 Número de Descargas de Malware y Archivos sospechosos HoneyPot Cowrie



Fuente: El Autor.

Estos provienen de diferentes accesos y conexiones desarrolladas por las direcciones IP expuestas en la Figura 85.

Figura 85 Direcciones IP orígenes de amenaza de Archivos sospechosos y Malware HoneyPot Cowrie



Fuente: El Autor.

8.3.2 Resumen y Estadísticas

La Tabla 5 muestra cuántos ataques se realizaron discriminados por categoría. En este caso se hace necesario resaltar que la mayoría de las peticiones que sucedieron fueron maliciosas.

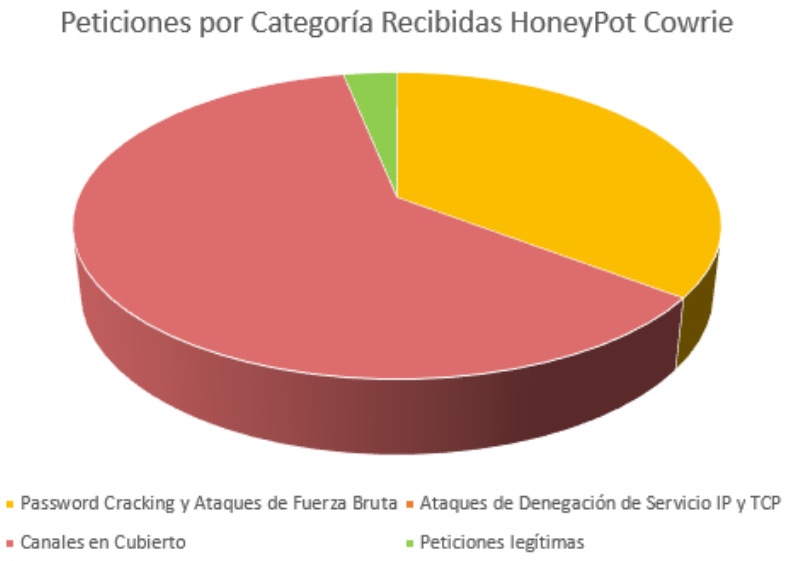
Tabla 5 Número de Peticiones registradas por categoría Cowrie

Categoría	Número de Peticiones	Porcentaje
Password Cracking y Ataques de Fuerza Bruta	14870	35,3366127
Ataques de Denegación de Servicio IP y TCP	0	0
Canales en Cubierto	25867	61,4695468
Peticiones legítimas	1344	3,19384045
Total Peticiones HoneyPots	42081	

Fuente: El Autor.

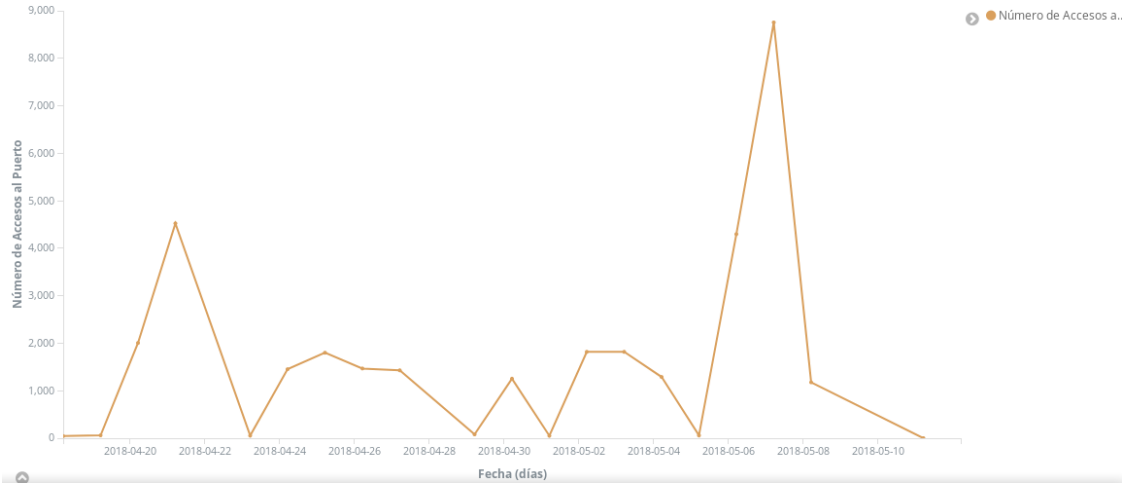
Al igual que en el análisis del *HoneyPot* anterior se otorga una exposición gráfica de la tabla anterior y el número de accesos al *HoneyPot* durante el tiempo en que estuvo expuesto, para ello se proporciona las Figura 86 y Figura 87.

Figura 86 Porcentaje Peticiones por Categorías Recibidas por HoneyPot Cowrie



Fuente: El Autor.

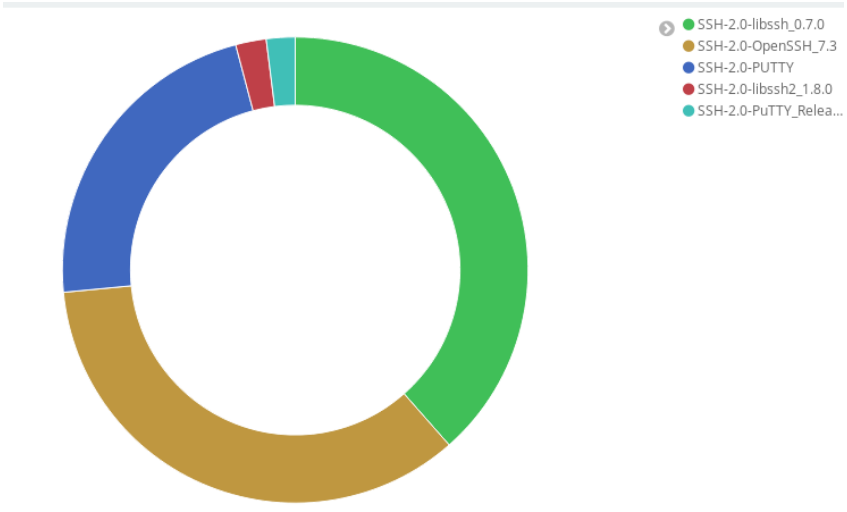
Figura 87 Número de Peticiones por Día HoneyPot Cowrie



Fuente: El Autor.

Como dato curioso, la herramienta *Cowrie* ofrece el porcentaje de peticiones realizados a cada versión del servicio simulada. Se puede observar en la Figura 88 cuáles han sido las versiones que han sido más atacadas.

Figura 88 Versiones más atacadas SSH simuladas por HoneyPot Cowrie



Fuente: El Autor.

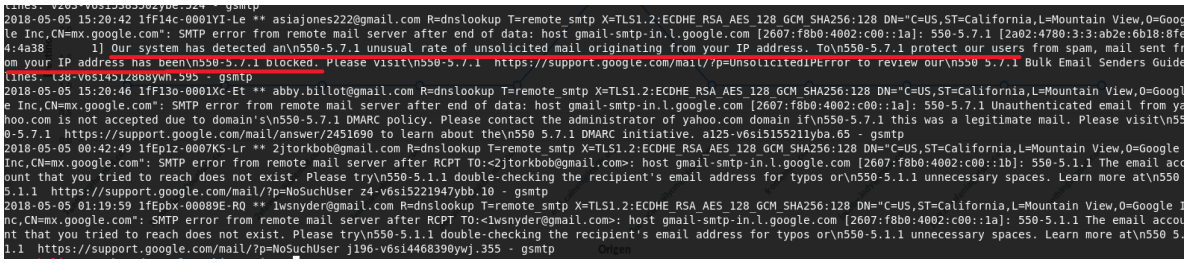
8.4 ATAQUES HACIA HONEYPOT SHIVA (PROTOCOLO SMTP)

8.4.1 Ataques Registrados

8.4.1.1 Spam

Aunque el *HoneyPot* registró eventos relacionados al *Spam,* en realidad no tuvo mucha actividad ya que pronto fue bloqueado por la mayoría de proveedores de correo electrónico, como muestra la Figura 89*.* Sin embargo, se puede resaltar una breve actividad sobre distribución de correos electrónicos; la mayoría de ellos con mensajes alusivos a contenido adulto o descarga de software malicioso.

Figura 89 Bloqueo de dirección IP para distribución de SMTP HoneyPot Shiva

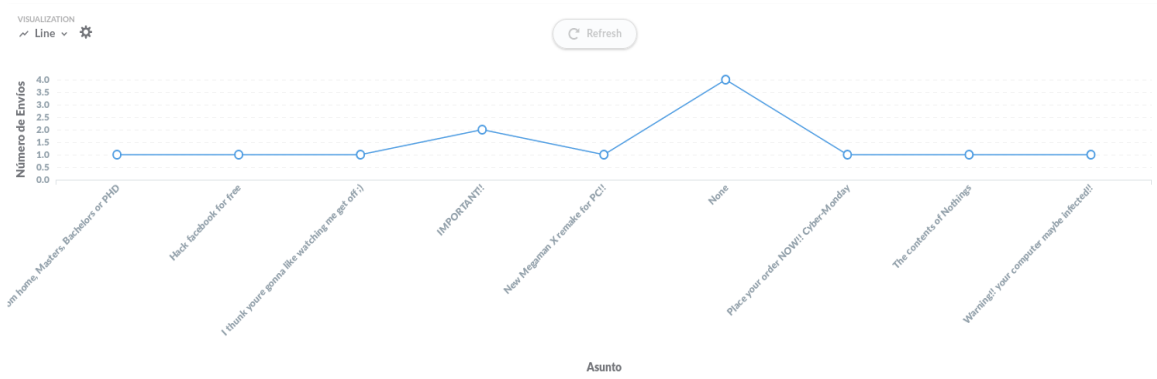


Fuente: El Autor.

Nota: El proveedor de *VPS* también bloqueó el puerto estándar *25* para la comunicación *SMTP,* algo que tuvo gran impacto en la recolección de datos. Sin embargo, se usaron otros puertos durante la recolección como el puerto *587* y el *2525.*

En Figura 90 se expone la agrupación de los mensajes que tienen el mismo asunto, pero cuyo contenido es diferente. Como se puede observar la mayoría de los asuntos prometen a la víctima el contacto con mujeres, la descarga de nuevos programas, videojuegos y nuevos productos.

Figura 90 Asuntos diferentes de campañas de Spam Shiva HoneyPot



Fuente: El Autor.

Algunos de estos mensajes tenían referencias hacia otros sitios, de los cuales se destacan los 4 expuestos a continuación debido a su contenido malicioso o relacionado con publicidad falsa. En este punto se resalta el uso de recortadores de *URL* que son usados para confundir a la víctima y hacer que la *URL* destino no sea sospechosa. Lo anterior se ve en la Figura 91.

Figura 91 Urls destacadas como contenido de Spam Shiva HoneyPot

Hyper Link
http://go.deliverymodo.com/afu.php?id=792658
https://bit.ly/2Ic4Msx
https://bit.ly/2JUqoXk
https://bit.ly/2jvpiX5

Fuente: El Autor.

El envío de correos fue amentando conforme pasaron los días de exposición del *HoneyPot* hasta el momento en que fue bloqueada. La Figura 92 muestra el número de correos *Spam* que fueron distribuidos.

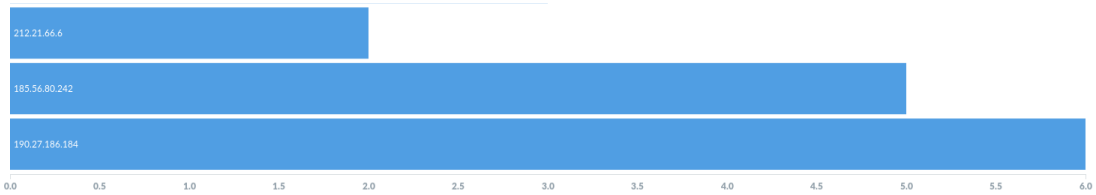
Figura 92 Númeor de spam enviado durante la exposición de HoneyPot Shiva

Fuente: El Autor.

Nota: Algunos correos no fueron enviados debido a que *Shiva HoneyPot* identifica su actividad maliciosa y bloquea el origen.

El origen del *Spam* fue poco variado dejando como resultado únicamente tres direcciones *IP* fuente del correo malicioso, evidenciados en la Figura 93.

Figura 93 Direcciones IP fuentes del correo Spam Shiva HoneyPot

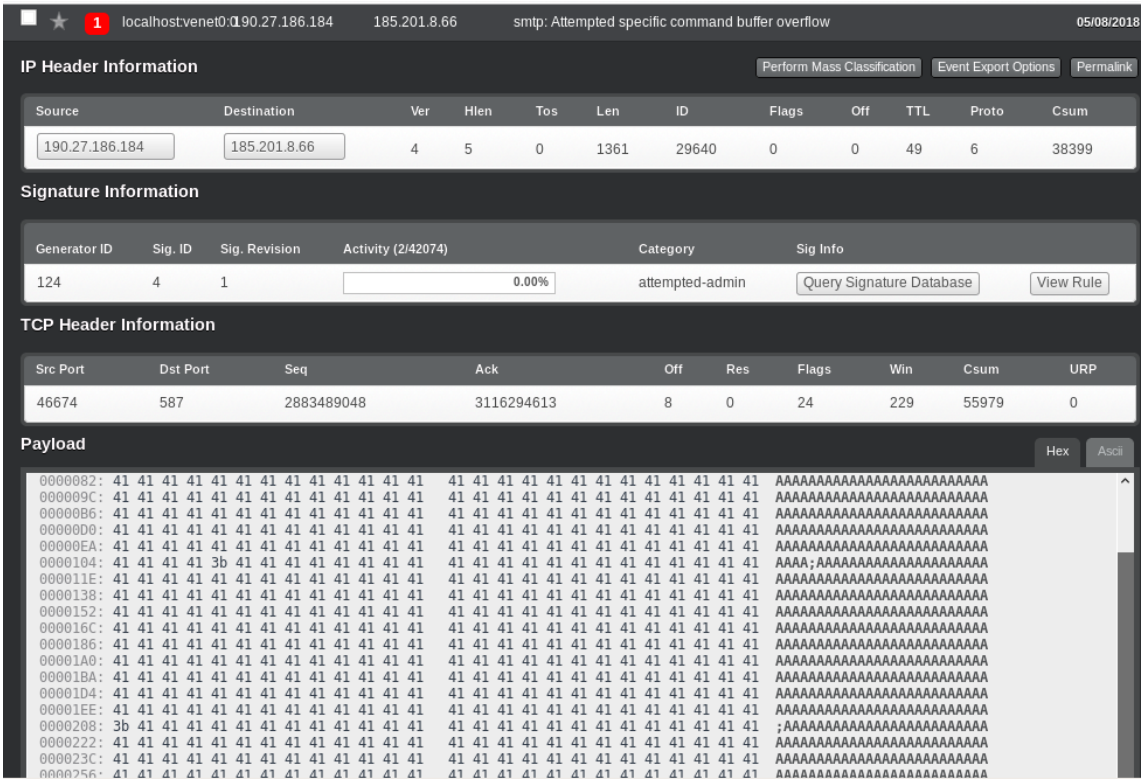


Fuente: El Autor.

8.4.1.2 Ejecución de Código Remota

Una amenaza que no se tenía prevista se manifestó durante la etapa de recolección, se trataba de un intento de *Ejecución de Código Remoto RCE* el cual tenía como objetivo quebrar el servicio del *HoneyPot.* La amenaza no fue identificada con el *HoneyPot,* por su parte fue detectada por *Snort,* el cual es evidenciado en la Figura 94.

Figura 94 Intento de Ejecución de Código Remoto RCE en Shiva HoneyPot



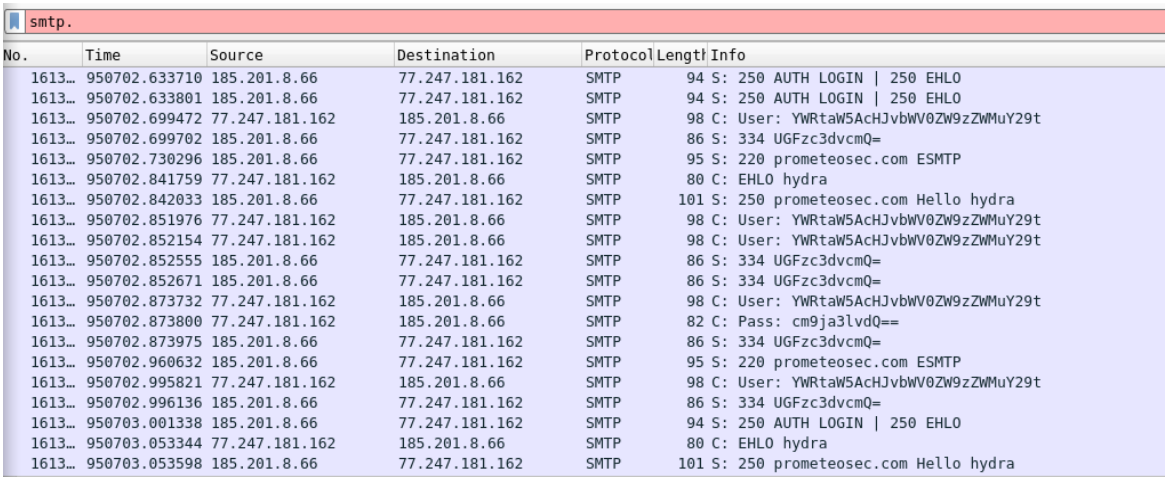
Fuente: El Autor.

De este tipo de amenazas se presentaron *4* eventos clasificados con severidad grave por el Sistema de Detección de Intrusos.

8.4.1.3 Fuerza Bruta

Existió un constante intento de autenticación al servicio, aunque éste no tenía configurados dichos parámetros en su configuración. Se considera un ataque de fuerza bruta debido al número elevado de intentos que existieron. Esta amenaza no es registrada en ninguno de los sensores del *HoneyPot* y tampoco ha sido capturada por el Sistema de Detección de Intrusos. Esta amenaza se puede observar gracias al *sniffer TcpDump,* demostrado en la Figura 95*.*

Figura 95 Peticiones de Fuerza Bruta dirigidas a Shiva HoneyPot



smtp.

No.	Time	Source	Destination	Protocol	Length	Info
1613…	950702.633710	185.201.8.66	77.247.181.162	SMTP	94	S: 250 AUTH LOGIN | 250 EHLO
1613…	950702.633801	185.201.8.66	77.247.181.162	SMTP	94	S: 250 AUTH LOGIN | 250 EHLO
1613…	950702.699472	77.247.181.162	185.201.8.66	SMTP	98	C: User: YWRtaW5AcHJvbWV0ZW9zZWMuY29t
1613…	950702.699702	185.201.8.66	77.247.181.162	SMTP	86	S: 334 UGFzc3dvcmQ=
1613…	950702.730296	185.201.8.66	77.247.181.162	SMTP	95	S: 220 prometeosec.com ESMTP
1613…	950702.841759	77.247.181.162	185.201.8.66	SMTP	80	C: EHLO hydra
1613…	950702.842033	185.201.8.66	77.247.181.162	SMTP	101	S: 250 prometeosec.com Hello hydra
1613…	950702.851976	77.247.181.162	185.201.8.66	SMTP	98	C: User: YWRtaW5AcHJvbWV0ZW9zZWMuY29t
1613…	950702.852154	77.247.181.162	185.201.8.66	SMTP	98	C: User: YWRtaW5AcHJvbWV0ZW9zZWMuY29t
1613…	950702.852555	185.201.8.66	77.247.181.162	SMTP	86	S: 334 UGFzc3dvcmQ=
1613…	950702.852671	185.201.8.66	77.247.181.162	SMTP	86	S: 334 UGFzc3dvcmQ=
1613…	950702.873732	77.247.181.162	185.201.8.66	SMTP	98	C: User: YWRtaW5AcHJvbWV0ZW9zZWMuY29t
1613…	950702.873800	77.247.181.162	185.201.8.66	SMTP	82	C: Pass: cm9ja3lvdQ==
1613…	950702.873975	185.201.8.66	77.247.181.162	SMTP	86	S: 334 UGFzc3dvcmQ=
1613…	950702.960632	185.201.8.66	77.247.181.162	SMTP	95	S: 220 prometeosec.com ESMTP
1613…	950702.995821	77.247.181.162	185.201.8.66	SMTP	98	C: User: YWRtaW5AcHJvbWV0ZW9zZWMuY29t
1613…	950702.996136	185.201.8.66	77.247.181.162	SMTP	86	S: 334 UGFzc3dvcmQ=
1613…	950703.001338	185.201.8.66	77.247.181.162	SMTP	94	S: 250 AUTH LOGIN | 250 EHLO
1613…	950703.053344	77.247.181.162	185.201.8.66	SMTP	80	C: EHLO hydra
1613…	950703.053598	185.201.8.66	77.247.181.162	SMTP	101	S: 250 prometeosec.com Hello hydra

Fuente: El Autor.

Los mensajes de tipo *EHLO hydra* dan un indicio que la herramienta usada es *Hydra,* reconocida en el mundo del *hacking* por sus capacidades de *Fuerza bruta.*

8.4.2 Resumen y Estadísticas

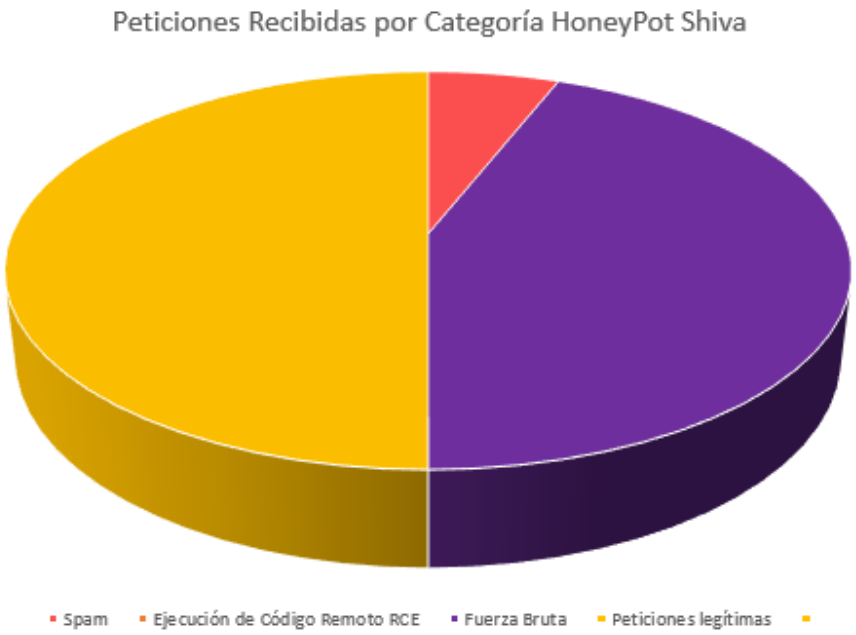
La Tabla 6 establece un valor aproximado de los diferentes ataques desarrollados sobre el *HoneyPot* Shiva. Se considera un valor aproximado debido que no se puede resolver con certeza el número de paquetes enviados por algunas limitantes del *HoneyPot.* Para una mayor compresión de estos datos se ortoga la Figura 96.

Tabla 6 Número de peticiones registradas por categoría Shiva

Categoría	Número de Peticiones	Porcentaje
Spam	1322	11,9292546
Ejecución de Código Remoto RCE	4	0,03609457
Fuerza Bruta	9756	88,0346508
Peticiones legítimas	0	0
Total Amenazas Detectadas	11082	

Fuente: El Autor.

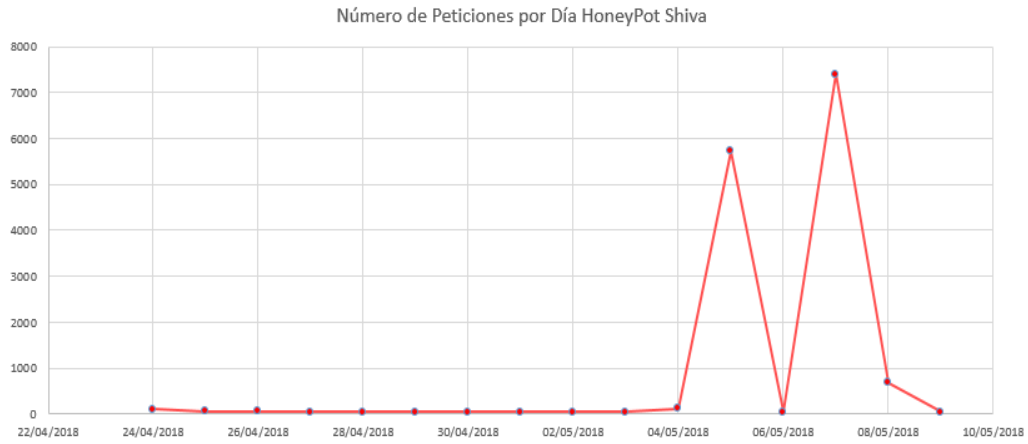
Figura 96 Porcentaje de Peticiones Recibidas por Categoría HoneyPot Shiva



Fuente: El Autor.

La Figura 97 muestra el número de peticiones realizadas por día hacia el *HoneyPot Shiva.*

Figura 97 Número de Peticiones por Día HoneyPot Shiva



Fuente: El Autor.

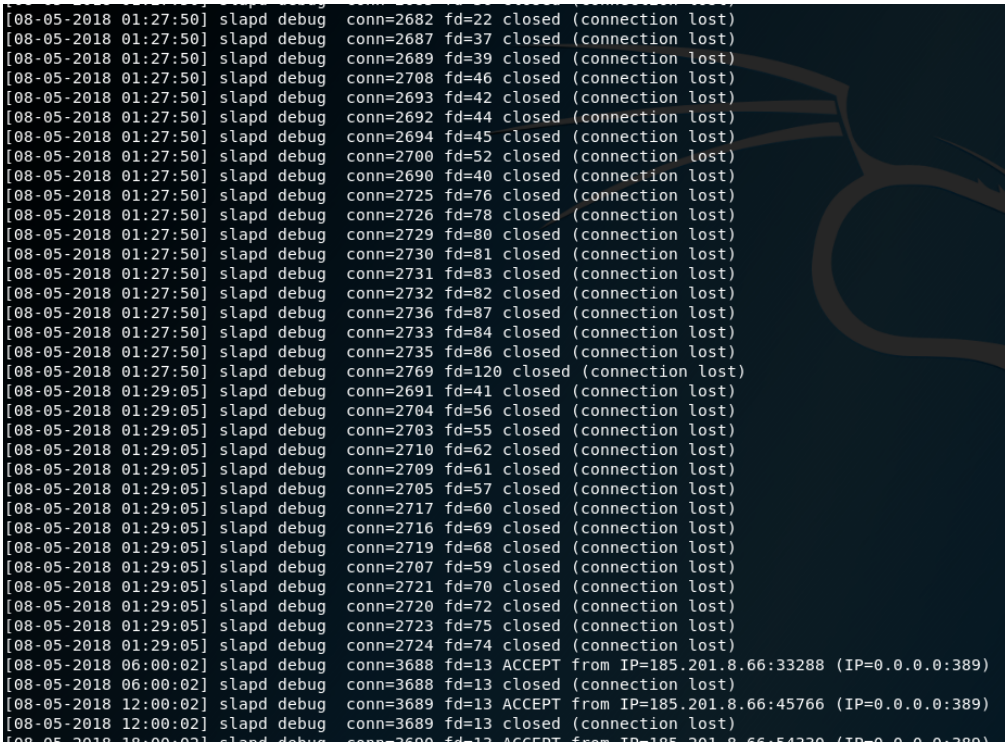
8.5 ATAQUES HACIA HONEYPOT OPENLDAP (PROTOCOLO LDAP)

8.5.1 Ataques Registrados

8.5.1.1 Denegación del Servicio por Uso Excesivo de los Recursos

Se registra un número elevado de peticiones que simplemente cierran o abren conexiones hacia el servicio *LDAP,* este comportamiento puede considerarse como un Ataque de Denegación de Servicio. Se registran aproximadamente *5425* peticiones desde que se expone el *HoneyPot.* Lo anterior se refleja en la Figura 98.

Figura 98 Número elevado de conexiones abiertas y cerradas posible DoS OpenLDAP

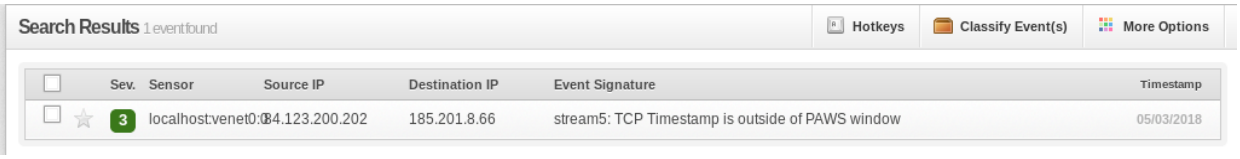


Fuente: El Autor.

Snort ha clasificado este tipo de evento con severidad leve, aunque identifica esta amenaza a nivel de la capa de transporte en el protocolo *TCP.* Esta amenaza puede

determinarse por el cambio de *TimeStamp* del protocolo. El evento se evidencia en la Figura 99.

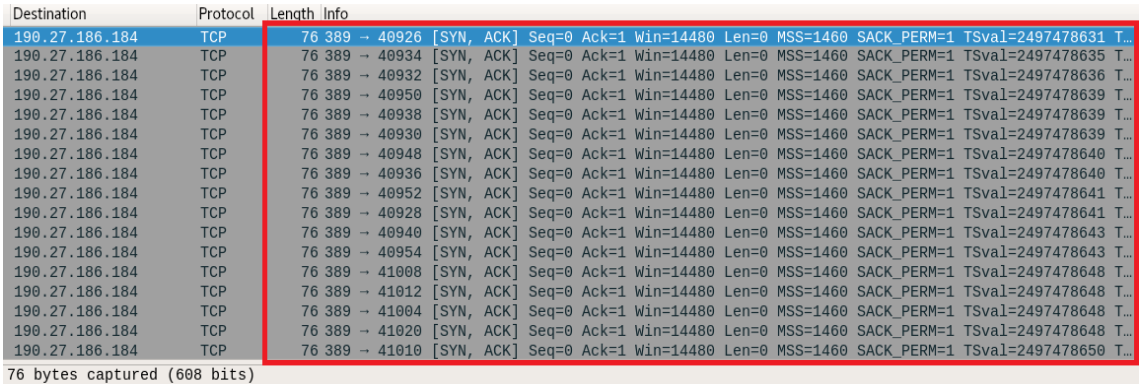
Figura 99 Snort posible Denegación de Servicio cambio Timestamp en LDAP



Fuente: El Autor.

Con la herramienta *TcpDump* se puede confirmar el comportamiento malicioso debido a una gran cantidad de peticiones del protocolo *TCP* de tipo *SYN+ACK.* Este es un comportamiento relacionado al vector de amenaza *TCP Flood,* como se ve en la Figura 100.

Figura 100 TCPDump confirmación de Denegación de Servicio LDAP TCP Flood



Fuente: El Autor.

Este es el único vector de ataque que se registra para este *HoneyPot,* hay que tener en cuenta que este se trata de un servicio mal configurado por lo cual es posible que no se identifiquen amenazas más especializadas. Llama la atención que *Snort* no identifica actividades maliciosas.

8.5.2 Resumen y Estadísticas

Se clasifican dos tipos de peticiones, aquellas dirigidas hacia *Denegación de Servicio* y aquellas que son legítimas. La Tabla 7 expone cuál es el número de peticiones de cada tipo.

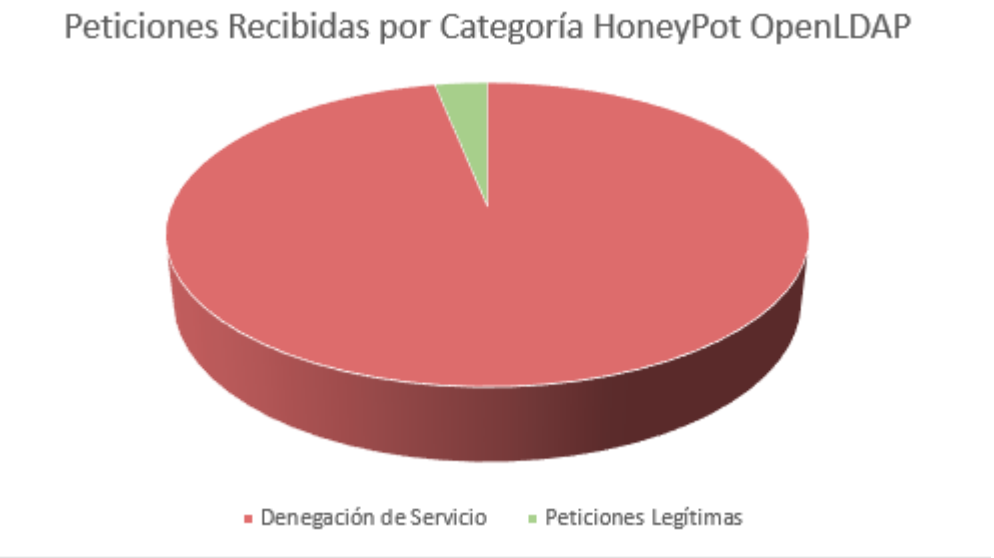
Tabla 7 Número de Peticiones Registradas por Categoría OpenLDAP

Categoría	Número de Peticiones	Porcentaje
Denegación de Servicio	5425	96,8404141
Peticiones Legítimas	177	3,15958586
Total Peticiones HoneyPots	5602	

Fuente: El Autor.

La Figura 101 expone en forma visual los datos enmarcados anteriormente.

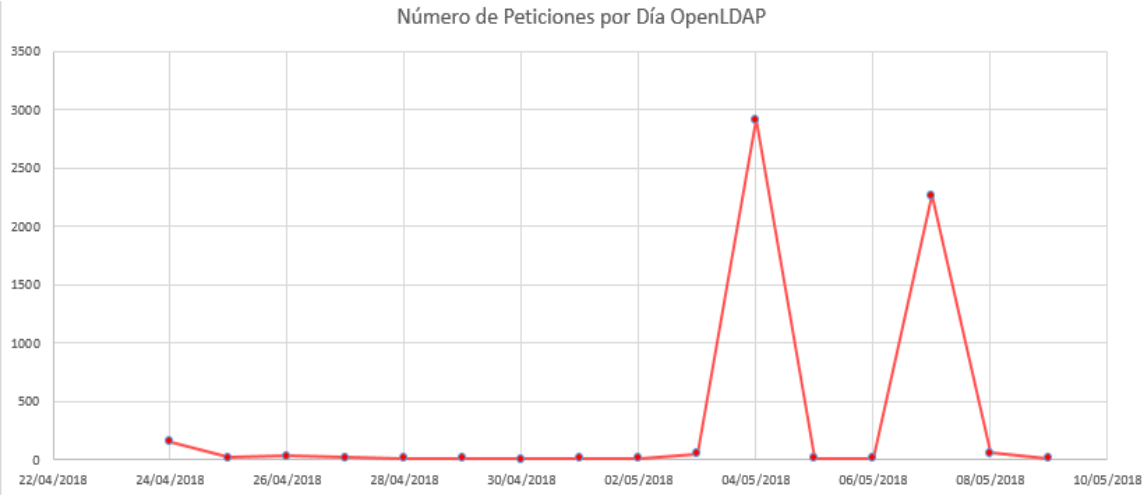
Figura 101 Porcentaje de Peticiones por Categoría HoneyPot OpenLDAP



Fuente: El Autor.

Finalmente, se registra el número de peticiones generadas por días por medio de la Figura 102.

Figura 102 Número de Peticiones por Día HoneyPot OpenLDAP



Fuente: El Autor.

Contrastando los datos de la última gráfica con los ataques registrados, se puede determinar que pudo existir un ataque de denegación de servicios en las fechas 5 y 7 de mayo.

9 ACCIONES DE HARDENING PARA LOS SERVICIOS DE LOS PROTOCOLOS HTTP, LDAP, SSH Y SMTP

Finalmente, después de aplicar el proceso de recolección de datos y analítica de la información, se hace necesario definir algunas acciones de defensa que se pueden adoptar en los diferentes servicios y la red misma para prevenir que su ocurrencia continúe o pueda tener algún tipo de afectación. En este capítulo se define por cada uno de los servicios qué tipo de configuraciones o instalación se debe tener en cuenta para blindar la información de la empresa conforme a las amenazas identificadas. Luego de ello se recogen algunas recomendaciones generales que pueden ser aplicadas de forma transversal para un ambiente en red en general.

9.1 SERVICIO WEB (PROTOCOLO HTTP)

9.1.1 Parameter Injection

a. La Inyección de Parámetros se aprovecha principalmente de las entradas de la Aplicación Web, por ende, la mejor defensa es desconfiar de todos los datos que puedan ser generados por el usuario. Cualquier parámetro debe validarse teniendo en cuenta la lógica de negocio y su contexto dentro del sistema.

b. En el caso de *SQL Injection,* una de las inyecciones de parámetros más conocidas y peligrosas, es necesario usar librerías estándar que permitan crear consultas con parámetros en los *Queries* lanzados hacia las Bases de Datos. Esto podría prevenir cualquier intento de ingresar caracteres que permitan escapar la consulta para así explotar la vulnerabilidad. Otra opción puede ser el uso de Procedimientos Almacenados, aunque es necesario que éstos no generen consultas dinámicas ya que de ser así no se estaría solucionando la vulnerabilidad[155].

   También se puede declarar una Lista Blanca de las entradas permitidas, es decir, si un campo está dispuesto para realizar un login sólo se permitirá el uso de nombres de usuarios en dicho campo. Esto también puede ayudar a prevenir la inserción de caracteres de escape.

---

[155] OWASP OPEN WEB APPLICATION SECURITY PROYECT. SQL Injection Prevention Cheat Sheet [online]. OWASP feb, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet >

c. En el caso de *Server Side Request Forgery SSRF* es necesario verificar si las Aplicaciones Web tienen configuraciones que requieran el uso de peticiones a otros servicios externos. Si es así, se recomienda verificar hasta qué grado pueden ser modificados estos parámetros de conexión por el usuario final. Si el usuario requiere tener un control completo sobre estos servicios, se debe establecer una Lista Blanca de los posibles valores que puede ingresar el mismo.

   También se recomienda deshabilitar los esquemas de direcciones que no sean usados. Ejemplos (*file:///, dict://, ftp://, gopher://*)[156].

d. Prestar especial atención a los archivos y datos procesados por la aplicación. Existen otro tipo de inyecciones relacionados al procesamiento de los mismos, los cuales pueden aprovecharse de ciertas vulnerabilidades en el proceso de deserialización. Algunas preguntas que se pueden realizar en este punto son: ¿Cuáles son los datos que procesará la aplicación? ¿Cuáles son los directorios de entrada que debería soportar la aplicación? ¿Cómo se manifiesta el flujo de los datos a través de los componentes internos de la aplicación? ¿Existe algún tipo de validación que deba realizarse, además del meramente lógico? ¿Cómo guarda los datos la aplicación? ¿Los datos requieren de algún tipo de proceso de cifrado y descifrado?

   Teniendo estos puntos clave en cuenta, se pueden identificar fácilmente posibles puntos de fallo al momento de procesar los datos de entrada a la Aplicación Web.

e. Mantener actualizadas las librerías, dependencias y componentes de la Aplicación Web. Esto incluye al mismo Servidor Web. Cualquier tipo de software puede ser susceptible a un fallo de seguridad, si se establecen planes de actualización es posible mitigar una posible explotación por el uso de software obsoleto.

f. En caso de no tener el código fuente de la aplicación, no poder aplicar actualizaciones fácilmente o contar con un ambiente bastante sensible al cambio; se recomienda la adición de mecanismos de seguridad que puedan ayudar a reducir este tipo de amenazas. Para Aplicaciones Web se puede usar un *Web Application Firewall WAF*[157]*.* Cabe aclarar que este mecanismo de seguridad no es infalible, y en lo posible se recomienda aplicar el patrón

---

[156] MUSCAT, Ian. What is Server Side Request Forgery (SSRF)? [online]. Acunetix ma., 2017. [citado 10 ma., 2018]. Disponible en Internet: < https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/ >

[157] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 198 p.

de *Seguridad por Defecto;* es decir, mitigar los riesgos de seguridad desde el mismo código de la Aplicación Web.

9.1.2 Exposición de Datos y Control de Acceso Débil

La Exposición de Datos y el Control de Acceso Débil son dos tipos de amenazas que comparten algunas características en su proceso de explotación. Por esta razón es posible definir acciones de defensa que cubran a ambas.

a. Definir filtros globales para la Aplicación Web y los recursos privilegiados. Estos filtros globales estarían encargados de verificar si el usuario final en realidad cuenta con los privilegios requeridos para acceder al recurso. Para ello la Aplicación Web debe proveer mecanismos de Autenticación y Autorización, para así administrar los permisos de los usuarios y sus credenciales de acceso.

   Dependiendo de los requerimientos de negocio, los procesos de Autenticación y Autorización podrían ser aislados para las Aplicaciones Web o podrían estar integrados a algún tipo de proceso central, como es la comunicación con servidores de Directorios Activos y otros protocolos. En caso de estar integrado con un componente externo es necesario asegurar que la comunicación hacia este elemento del sistema ofrece capacidades de Confidencialidad, Integridad y Disponibilidad.

b. Identificar cuáles son los recursos a los cuales cada usuario tiene acceso. En otras palabras, definir el esquema general de permisos sobre los objetos del sistema. Por ejemplo, en una Aplicación Web encargada de la contabilidad de una organización; los contadores tendrán acceso a ciertos recursos relacionados con sus labores, pero probablemente el gerente de la dependencia tendrá más permisos o accesos a otro tipo de objetos con mayor grado de Confidencialidad.

   Definir el grado de Confidencialidad de los Datos se hace vital para aclarar un buen esquema de permisos sobre el sistema.

c. Definir uno o varios estándares de Control de Acceso. Ayuda a los puntos anteriores tener en cuenta que existen varios estándares conocidos para al manejo del Control de Acceso. Algunos de ellos son: *Control de Acceso basados en Roles,* que se basan en el perfil de los individuos partiendo de roles definidos que pueden tener relación con la lógica de negocio; *Control de Acceso Discrecional,* el cual parte de dividir a cada uno de los usuarios en

grupos y así definir los permisos tanto del individuo como del grupo en su totalidad; *Control de Acceso Obligatorio,* el cual define que se deben establecer un conjunto de políticas de seguridad las cuales deben ser inviolables por todos los usuarios del sistema y *Control de Acceso basados en Permisos,* los cuales parten de que todo objeto puede tener un esquema de permisos básicos (lectura, escritura, ejecución, entre otros) donde a partir de ellos se puede definir los permisos para cada uno de los usuarios[158].

Los estándares de Control de Acceso pueden estar mezclados entre sí para garantizar una mejor protección. Sin embargo, es necesario discutir estas inclusiones de medidas de seguridad debido que podrían impactar fuertemente a la Usabilidad de las Aplicaciones.

d. De ninguna manera limitar la segregación y esquema de permisos a las Aplicaciones Web, extenderlo hacia todos los componentes que interactúan con el entorno Web. Esta es una buena práctica usada en cuanto a seguridad y permite ofrecer diferentes niveles de protección. Por ejemplo, suponga que una Aplicación Web es vulnerable a *Path Transversal* por lo cual se puede navegar libremente en el Sistema de Archivos que aloja el Servidor Web, ahora este servidor puede tener un Sistema Operativo basado en *Unix* donde se sabe que los ficheros */etc/shadow* y */etc/passwd* contienen información crítica y de suma importancia. Si en este escenario existe una correcta segregación de permisos sobre el proceso que ejecuta el Servicio Web, es decir, no se está ejecutando como *root* o cualquier usuario privilegiado; es poco probable que, aunque se explote la vulnerabilidad se pueda acceder a estos datos confidenciales.

e. Tener especial cuidado con los recursos que pueden ser accedidos públicamente o como usuario anónimo. Cualquier tipo de información que sea de carácter público debe ser tratada con especial cuidado, revisando que no se esté publicando algún dato confidencial que pueda comprometer la estructura o arquitectura del Servidor Web y las Aplicaciones. Entiéndase por recurso, cualquier tipo de dato o información que pueda otorgar el Servidor Web.

   Un ejemplo bastante conocido son los HTTP Headers. En ocasiones estos elementos pueden revelar información de las versiones y tecnologías de las

[158] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Access Control Cheat Sheet [online]. OWASP ene, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Access_Control_Cheat_Sheet >

Aplicaciones Web. Como en este caso los HTTP Headers son, en lo general públicos, se recomienda deshabilitarlos completamente[159].

f. Definir si el acceso a los recursos de las Aplicaciones Web se debe discriminar por direcciones IP o direcciones MAC. En algunos casos es probable que sólo se pueda acceder a la aplicación desde ciertos dispositivos, en este caso puede ser requerido bloquear algunos orígenes de la comunicación, un ejemplo, pueden ser algunos *routers* que bloquean el acceso a su panel de administración por medio de HTTP únicamente para las redes internas. De todas formas, se recomienda restringir el acceso por medio direcciones IP o direcciones MAC, no desde la Capa de Aplicación sino desde el Enlace de Datos haciendo uso de otros mecanismos de control como *Firewalls.*

9.1.3 Cross Side-Scripting XSS

El *Cross Side-Scripting* se puede clasificar como un tipo de inyección, aunque requiere de un tratamiento especial a comparación de la amenaza *Parameter Injection,* también pueden ser aplicables algunos de los mecanismos de defensa identificados en ese punto.

a. Bloquear o eliminar cualquier tipo de carácter que pueda estar relacionado con la inserción de código. Por ejemplo, eliminar de las peticiones cualquier carácter de tipo *"<"* o *">"* o si bien retornar un error al momento de procesar este tipo de peticiones[160].

b. Prestar bastante atención sobre cómo se renderiza la información del lado del cliente y de esta manera seleccionar una codificación adecuada para cada caso. Identificar básicamente si el usuario puede insertar información en páginas *HTML,* código Javascript, *CSS* y demás. Por ejemplo, si el nombre de usuario se está quemando directamente en el código *Javascript* se debe procurar usar una codificación acorde a este lenguaje que impida el uso de caracteres de escape[160].

---

[159] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Broken Access Control [online]. OWASP mar, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Broken_Access_Control >

[160] OWASP OPEN WEB APPLICATION SECURITY PROYECT. XSS (Cross Site Scripting) Prevention Cheat Sheet [online]. OWASP may., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet >

c. Definir una política de contenido acorde a los requerimientos del negocio. Las políticas de seguridad de contenido (conocidas como *Content Security Policy CSP*) permiten establecer cabeceras HTTP que indican al navegador que ciertos recursos sólo pueden ser ejecutados en dominios específicos. En caso de explotarse un *XSS* esto ayuda a que las peticiones no puedan ser transversales entre subdominios y/o dominios de la Aplicación Web[160].

d. Asegurar las Cookies y definir el *X-XSS-Protection* Header para el Servidor Web. El atributo *HttpOnly* de las Cookies previene la manipulación de estos elementos por medio de Javascript, es decir si se llegase a explotar este tipo de vulnerabilidades al menos no se filtrarían los datos de las Cookies. Adicionalmente el *X-XSS-Protection* Header indica al navegador que intente sanear cualquier tipo de posible ataque *XSS.* Estos mecanismos no son infalibles, pero pueden ayudar bastante a proteger las Aplicaciones Web.

e. En caso de no tener mantenibilidad de código se recomienda usar otro mecanismo de protección como un *Web Application Firewall WAF.* Nuevamente, cuando se tenga una aplicación sensible a actualizaciones o sobre la cual no se tenga el código fuente, es necesario agregar un mecanismo de seguridad adicional como lo es un *WAF*[161]*.*

9.1.4 Ataques de Fuerza Bruta

a. Establecer un número máximo de reintentos en aquellas operaciones que se consideren críticas para la Aplicación Web. Es necesario bloquear un recurso cuando se identifique un acceso repetitivo hacia un recurso de la aplicación, en especial aquellos que se relacionan con procesos críticos. Existe un ejemplo bastante común al momento de realizar la autenticación en las Aplicaciones Web, si el número de intentos de autenticación es fallido en cierto periodo de tiempo (por ejemplo 3 intentos en menos de 1 minuto); el usuario puede ser bloqueado o de otra manera se puede prohibir el acceso a la Aplicación Web.

b. Usar *captchas* para introducir un mecanismo de validación de las peticiones. Los *captchas* son reconocidos mecanismos de defensa contra los Ataques de Fuerza bruta ya que requieren la interacción de un usuario final. Esto

[161] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op. cit., 198 p.

previene el uso de *scripts* y rutinas automatizadas para realizar peticiones, ya que éstos no pueden adivinar fácilmente el valor de los *captchas*[162].

c. Bloquear el origen de las peticiones a la Aplicación Web. Se puede optar por banear las direcciones IP o MAC dependiendo del número de peticiones generados desde ese origen. A diferencia de otros mecanismos de defensa, esto puede realizarse en la lógica de la Aplicación Web, agregando una capa adicional a los IDPS y otros sistemas de defensa[162].

9.1.5 Otras Recomendaciones

a. Existen otro tipo de amenazas como *Cross-Side Request Forgery CSRF* el cual no se estudió durante el desarrollo del proyecto. Este ataque consiste en lanzar una petición desde una máquina donde exista una sesión autorizada, la petición puede ser lanzada desde contenido oculto de una *URL* u otro recurso como Correos Electrónicos. Por medio de esta amenaza se puede desarrollar acciones maliciosas como corrupción de datos o apertura de puertas traseras.

   Para prevenir este tipo de amenazas se puede inducir un dato aleatorio en todas las respuestas que otorga la Aplicación Web, luego cuando se realice la petición hacia la aplicación se verificara este valor aleatorio y si es válido, la petición puede ser atendida. Otro tipo de mecanismos de defensa se especifican en la página oficial de *OWASP*[163].

b. Adoptar buenas prácticas de desarrollo de software de Aplicaciones Web. Muchos de los patrones de desarrollo previenen la materialización de los riesgos descritos anteriormente, tener claro cuáles son los patrones de diseño y sus respectivos casos de uso pueden blindar completamente a la aplicación.

---

[162] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Blocking Brute Force Attacks [online]. OWASP mar, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks >

[163] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Cross-Site Request Forgery CSRF Prevention Cheat Sheet [online]. OWASP mar, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet >

9.2 SERVICIO SSH (PROTOCOLO SSH)

9.2.1 Password Cracking y Ataques de Fuerza Bruta

a. Configurar el acceso por medio de Llaves Públicas o Autenticación basada en *Host.* La autenticación basada en contraseñas no se recomienda de ninguna manera en ambientes críticos para el Servicio *SSH.* En vez de ello se recomienda aplicar una Autenticación por Llave Pública; la cual consiste en generar un par de llaves tanto pública o privada y luego enviar la llave pública al servidor para así comprobar la autenticidad del usuario. El otro método puede ser aquel basado en *Host,* donde por cada dirección *IP* se genera una cadena o registro criptográfico; una especie de *fingerpriting* que permita conectarse a la sesión *SSH* únicamente desde el *Host* específico[164].

   Ambos métodos pueden ser usados en conjunto y ambos ofrecen protección efectiva contra los intentos de *Password cracking* ya que involucran el uso de mecanismos criptográficos que no pueden ser fácilmente atacados por medio de amenazas y Fuerza Bruta.

   Es necesario tener en cuenta que habilitar estos métodos de autenticación pueden disminuir la usabilidad del servicio *SSH,* en especial en el caso que los usuarios del Sistema Operativo no tengan conocimientos sobre manejo de criptografía asimétrica.

   Nota: En el caso de Autenticación de Llave Pública ser bastante cuidadoso con el tratado de la clave privada, ya que ésta no se puede compartir a ningún otro *Host* y debe ser única por usuario.

b. Desactivar completamente la Autenticación por Contraseñas. Se recomienda desligar completamente el manejo de credenciales de usuarios locales de la autenticación de *SSH.* Esto evita posibles riesgos como que un usuario defina una contraseña débil para su inicio de sesión.

c. Deshabilitar el acceso de usuarios Administradores. Desactivar el acceso de los administradores (como lo son el usuario *Root* en Linux y *Administrator* en Windows), ayuda a prevenir los accesos no autorizados por medio de Ataques de Fuerza Bruta. Es preferible ejecutar comandos como *sudo,* en Linux y *runas,* en Windows; cuando ya se encuentre autenticado en el sistema. En pocas palabras, para ejecutar un comando como usuario

[164] YLONEN Tatu. Ssh Key [online]. Ssh.com mar., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.ssh.com/ssh/key/ >

administrador del sistema se recomienda acceder al Servicio *SSH* con credenciales de un usuario que pueda escalar fácilmente a Administrador[164]

d. Bloquear el usuario del Sistema Operativo cuando se haya superado un tiempo máximo de intentos fallidos. El Servicio *SSH* está fuertemente integrado con los procesos de autenticación y autorización del Sistema Operativo, por esta razón se pueden configurar componentes del propio sistema para bloquear temporalmente los usuarios que hayan tenido muchos intentos fallidos. Un ejemplo de ello se puede ver con Linux con los módulos de PAM, en especial de *pam_tally2.so* la configuración se explica en Tecmint[165].

e. Establecer un número de intentos de login máximo en un período de tiempo definido. Este mecanismo de defensa ayuda a prevenir Ataques de Fuerza Bruta en caso que se tenga habilitado la Autenticación por medio de Contraseñas y no se pueda configurar otro mecanismo de Autenticación. Normalmente *SSH* tiene establecido un número máximo de intentos fallidos a 3.

9.2.2 Canales en Cubierto

a. Aplicar auditorías a los comandos y acciones ejecutadas por los usuarios. Esta auditoría puede ser ejecutada por medio de módulos externos al Servicio *SSH,* aunque ayudarían a monitorear si ha existido una posible filtración de datos. Según el Sistema Operativo se pueden aplicar diferentes técnicas de auditoría, en el caso de Linux se puede configurar los módulos *PAM* por medio de *pam_tty_audit* para guardar cada uno de los comandos ejecutados en consola[166].

b. Aplicar una correcta segregación de permisos limitando el acceso a ciertos ficheros, comandos específicos, protocolos, entre otros. Esta división y esquema de permisos puede adoptar el Control de Acceso expuesto en la sección anterior del Servicio *SSH,* en especial aquellos relacionados a *Control de Acceso Obligatorio MAC, Control de Acceso Discrecional DAC* y *Control de Acceso basado en Roles RAC.* Existen componentes del propio

[165] SHERSTHA Narad. Use Pam_Tally2 to Lock and Unlock SSH Failed Login Attempts [online]. Tecmint abr., 2013. [citado 10 may., 2018]. Disponible en Internet: < https://www.tecmint.com/use-pam_tally2-to-lock-and-unlock-ssh-failed-login-attempts/ >

[166] REDHAT. 7.9. Configuring PAM for Auditing [online]. Redhat Customer Portal oct., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sec-configuring_pam_for_auditing >

Sistema Operativo que permiten configurar estos accesos de forma nativa sin la necesidad de instalar software de terceros[167].

La razón de configurar el Control de Acceso es que limitan al usuario del Servicio *SSH* a abrir puertas traseras, así como acceder a recursos o ejecutar operaciones maliciosas sobre el servidor.

c. Restringir el acceso de los usuarios finales aplicando el concepto de jaula (*Jail*). Es posible configurar el servicio *SSH* para restringir el acceso completo de los usuarios hacia ciertos recursos específicos, esta técnica se conoce como *Chrooted Jail* o simplemente *Jail* (jaula). A diferencia de la opción anterior tiene la ventaja de controlar completamente cuando un usuario puede acceder a ciertos archivos, sin necesidad de llevar a cabo una configuración de Control de Acceso tediosa. Es posible que también sea más segura que la opción anterior ya que el usuario técnicamente no podría salir del directorio done ha sido enjaulado.

   En Linux crear una jaula sobre un directorio puede hacerse sin instalar ningún tipo de software adicional tal como demuestra el portal *Tecmint[168].*

9.3 SERVICIO DE CORREO ELECTRÓNICO (PROTOCOLO SMTP)

9.3.1 Spam

a. Crear filtros personalizados de *Spam* del lado del Servidor. Una forma efectiva, usada por grandes proveedores; es generar filtros personalizados para controlar el *Spam.* Aunque es una técnica poco documentada, si se conoce a detalle el protocolo *SMTP;* su contenido y cabeceras, se pueden configurar reglas en los *IDPS* y diferentes mecanismos de protección para denegar cierto tipo de correos electrónicos provenientes de direcciones IP, con cierto dominio, con asuntos específicos o incluso con cierto tipo de adjuntos ya identificados como malignos.

   La ventaja de usar filtros del lado del servidor, es que pueden ser una primera medida de protección y no requieren de instrucción a los usuarios finales o mucho menos configuración en sus máquinas.

---

[167] OWASP OPEN WEB APPLICATION SECURITY PROYECT. Op cit., 231 p.

[168] KILI, Aaron. Restrict SSH User Access to Certain Directory Using Chrooted Jail [online]. Tecmint mar., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.tecmint.com/restrict-ssh-user-to-directory-using-chrooted-jail/ >

b. Usar software en el cliente o usuario final para identificación de correos de *Spam.* Muchos de los clientes de correo electrónico proveen funciones *Built-in* para identificar correos con posible contenido de *Spam.* Tal es el caso de *Outlook*[169]*,* por ejemplo, el cual puede establecer un filtro de protección contra el correo no deseado. Aunque esto no atañe directamente al protocolo *SMTP* y se encuentra, por otro lado, ligado a los protocolos de acceso de buzón; puede ser una táctica estratégica en la lucha de *Spam,* siendo una última línea de protección en caso que los filtros del lado del Servidor no hayan sido identificados.

c. Firmar digitalmente los Correos Electrónicos, en especial cuando se trate de ambientes organizacionales o contengan información importante. Las firmas digitales, las cuales no deben confundirse con las firmas personalizadas por los mensajes, ayudan a verificar la autenticidad y origen de los Correos Electrónicos; esto puede proteger a una compañía frente a una campaña de *Spam* ya que por medio de la aplicación de criptografía da un aval sobre quién fue el verdadero autor del Correo Electrónico.

   Muchos proveedores de Correos Electrónicos y sus respectivos clientes tienen el soporte para estas funcionalidades, las cuales en ocasiones pueden venir acompañadas del uso de cifrado asimétrico de los correos. Por ejemplo, el *Outlook* existe una forma básica de proteger los correos mediante la firma digital; incluso en su página oficial se explica cuál es la importancia de la firma digital y su diferencia con la firma tradicional o estándar[170].

d. Instruir a los usuarios sobre técnicas de identificación de *Spam.* No es mala idea prevenir a las víctimas del uso de *Spam,* para ello es necesario instruir a los usuarios de la organización sobre la identificación de correos que puedan contener código malicioso. Normas básicas como verificar el idioma del correo, su estructura, semántica y emisor; pueden aplicarse fácilmente por medio de campañas de capacitación. Enseñar a los usuarios finales que todo correo de origen desconocido debe ser desechado inmediatamente y procurar bloquear a su emisor.

---

[169] MICROSOFT. Change the level of protection in the Junk Email Filter [online]. Microsoft, jun., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://support.office.com/en-us/article/change-the-level-of-protection-in-the-junk-email-filter-e89c12d8-9d61-4320-8c57-d982c8d52f6b >

[170] MICROSOFT. Proteger los mensajes con una firma digital [online]. Microsoft, jun., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://support.office.com/es-es/article/proteger-los-mensajes-con-una-firma-digital-549ca2f1-a68f-4366-85fa-b3f4b5856fc6 >

9.3.2 Fuerza Bruta

a. Evitar el uso de autenticación por medio de usuario/contraseña. Algunos proveedores como *Gmail* tienen configurados sus servidores *SMTP* para autenticarse usando el método de *OAuth 2.0,* esto ayuda a prevenir que se compartan credenciales como usuario y contraseñas, las cuales pueden ser más fáciles de adivinar que un *Token* generado por *OAuth 2.0*[171]*.*

   Los tokens *OAuth 2.0* pueden ser codificados en Base 64 y enviados directamente al Servidor *SMTP.* Un Ataque de Fuerza Bruta que desee adivinar este tipo de tokens puede ser infructuoso debido al tamaño de caracteres que suelen tener. Este mecanismo de autenticación puede integrarse con el uso de *SASL*[171]*.*

   Otro tipo de mecanismos de autenticación comprenden la integración únicamente del framework *SASL.* Cuando se integra este tipo de framework se debe evitar el uso de contraseñas en texto plano debido a temas de confidencialidad de la información y para evitar las entradas posibles de los ataques de fuerza bruta. *Postfix* por ejemplo, tiene una integración con *SASL* que puede ayudar la manera de autenticación al servidor[172].

   Nota: No todos los proveedores de Correo Electrónico o Servidores *SMTP* cuentan con la compatibilidad necesaria para estos métodos de autenticación.

b. Verificar si el proveedor del Servicio *SMTP* cuenta con las configuraciones que limitan el número de intentos fallidos de login, si es así colocarlas de acuerdo a los requerimientos del negocio. Algunos proveedores del *SMTP* contienen en su configuración inicial la posibilidad de establecer un parámetro de números de intentos de autenticación fallida en cierto periodo de tiempo, si este número es sobrepasado es posible que el servidor no conteste durante otro tiempo establecido o que aumente el número de latencia.

---

[171] GMAIL. OAuth 2.0 Mechanism [online]. Google oct., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://developers.google.com/gmail/imap/xoauth2-protocol >

[172] POSTFIX. Postfix SASL Howto [online]. Postfix sep., 2017. [citado 10 may., 2018]. Disponible en Internet: < http://www.postfix.org/SASL_README.html >

Un ejemplo de ello en Linux, es el manejo de *Postfix* por medio de los parámetros smtpd_client_connection_count_limit y smtpd_client_connection_rate_limit[173].

9.3.3 Otras Recomendaciones

a. Agregar soporte de *TLS* al protocolo *SMTP.* Originalmente el protocolo de *SMTP* no tiene soporte de cifrado de su confidencialidad, sin embargo, desde el año 2002 se agregaron las extensiones necesarias para la adición de cifrado a nivel de transporte por medio de *TLS.* Para ello es necesario configurar el Servidor *SMTP* para obtener soporte del comando *STARTTLS,* afortunadamente muchos de los proveedores actualmente cuentan con esta funcionalidad[174].

   Agregar cifrado a nivel de transporte mejora la Confidencialidad e Integridad de algunos de los procesos realizados en una comunicación *SMTP,* esto es de gran importante en especial en los procesos de autenticación.

b. Cifrar los Correos Electrónicos por medio de Criptografía Asimétrica. Cifrar la comunicación a nivel de red por medio de *TLS* puede ser importante, sin embargo, es posible aún descifrar la capa de transporte aprovechándose de algunas debilidades del cifrado asimétrico. Para asegurar completamente que los correos se envían sobre un canal confidencial, se puede encapsular el propio correo electrónico haciendo uso de la criptografía de llave pública[175].

   Muchos de los proveedores de *SMTP* soportan estas capacidades, de hecho, es algo transparente para muchos de ellos y simplemente se puede lograr con la implementación de algunos *plugins.* Incluso existen proveedores como *ProtonMail*[175]*,* que ofrecen estas capacidades por defecto.

   El cifrado del correo electrónico podría no incluir las cabeceras del protocolo de *SMTP,* ya que son necesarias para encontrar el destinatario del correo electrónico.

c. Si no se cuenta con mucha experiencia sobre la Administración de Correos Electrónicos y configuración de Servidores *SMTP,* es preferible contratar un

---

[173] POSTFIX. Postfix Tunning Performance [online]. Postfix sep., 2017. [citado 10 may., 2018]. Disponible en Internet: < http://www.postfix.org/TUNING_README.html >

[174] INTERNET MAIL CONSORTIUM. SMTP Service Extension for Secure SMTP over Transport Layer Security [online]. Internet Engineering Task Force IETF feb, 2002. [citado 22 mar., 2018]. Disponible en Internet: < https://www.ietf.org/rfc/rfc3207.txt >

[175] PROTONMAIL. What is encrypted? [online]. ProtonMail mar., 2018. [citado may., 2018]. Disponible en Internet: < https://protonmail.com/support/knowledge-base/what-is-encrypted/ >

servicio externo. Muchos proveedores como *Gmail, Outlook, Proton Mail;* entre otros, ya cuentan con mecanismos de seguridad que ayudan a proteger la información de sus usuarios. Además de esto, ofrecen el servicio de correo a las diferentes empresas lo cual es bastante conveniente para aquellas que no cuentan con el personal y recursos suficientes para configurar sus propios servicios de *SMTP.*

9.4 SERVICIO DE DIRECTORIOS (PROTOCOLO LDAP)

9.4.1 Denegación de Servicio

a. Deshabilitar el uso de usuarios anónimos. Procurar establecer métodos de autenticación fuerte para los usuarios, teniendo en cuenta las características y capacidades del Directorio Activo o el Servicio *LDAP.* De esta manera se evita el acceso por medio de cuentas anónimas o que no requieran ingresar sus credenciales de usuario. Esta medida de defensa no sólo protege al servicio de peticiones maliciosas que puedan abusar del uso de sus recursos, también ayudan a proteger el propio servidor de estar involucrado en Ataques de Denegación de Servicio Distribuida por medio de Amplificación[176].

b. Deshabilitar el protocolo *LDAP* sobre *UDP*. Aunque esto no es una medida de protección que se haya encontrado directamente sobre el *HoneyPot,* se puede tomar esta defensa para prevenir que el servidor *LDAP* se encuentre involucrado en *botnets* o redes de computadoras dispuestas para Denegación de Servicio Distribuida por medio de Ataques de Amplificación[176].

9.4.2 Otras Recomendaciones

a. Agregar soporte de *TLS* a *LDAP.* Mucha de la información que se comparte a través de una comunicación *LDAP* es sensible, en especial si se considera que el servicio de Directorio Activo puede prestar las funcionalidades de una Base de Datos organizacional. Por esta razón se recomienda incluir el soporte de *TLS* para *LDAP,* el cual se puede encontrar en las versiones 2 y

[176] CONSTANTIN, Lucian. Attackers are now abusing exposed LDAP servers to amplify DDoS attacks [online]. IDG News Service, oct., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://www.pcworld.com/article/3135771/security/attackers-are-now-abusing-exposed-ldap-servers-to-amplify-ddos-attacks.html >

3 del protocolo. Esta es una forma útil de proteger la negociación sobre los mecanismos de autenticación, independientemente de cual sea[177].

b. Definir un método de autenticación que se ajuste a los requerimientos de la organización. Por definición *LDAP* soporta generalmente dos tipos de autenticación: Simple e integración de *SASL*[177]*.* Se debe procurar usar un método de acceso basado en el protocolo *SASL* ya que puede ser integrado con otros protocolos de comunicación más seguros como lo son *Kerberos* por medio de *SASL GSSAPI.* En caso de escoger un método de autenticación simple, es necesario añadir soporte sobre *LDAP* con *TLS* para asegurar al menos un nivel de Confidencialidad e Integridad a nivel de transporte.

c. Definir un Control de Acceso sobre las entidades del Directorio Activo. Identificar cuáles son los diferentes usuarios y cuál debería ser el acceso en cuanto a lectura y modificación de la información disponible en los diferentes objetos del Directorio Activo. Para ello es necesario preguntarse: ¿Cuáles son los objetos que acceden los diferentes usuarios? ¿Cuáles objetos y elementos contienen información sensible?[177]

   Es necesario adaptar el Directorio Activo a las políticas de seguridad de la organización.

d. Política de contraseñas. Normalmente los proveedores del Servicio *LDAP* guardan las credenciales de usuario de forma segura usando métodos de cifrado fuerte. Sin embargo, es necesario definir en la organización ciertas normas como cambio de contraseñas de manera periódicos, estructura de las contraseñas, entre otras características[177].

9.5 ACCIONES DE HARDENING ADICIONALES

a. Procurar aislar los procesos críticos de los servicios aplicando la técnica de *Sandboxing* o aislando los procesos a contextos controlados*.* Para implementarla se puede hacer uso de *software* disponible de forma comercial y gratuita como hipervisores y contenedores de procesos. Asegurar que los procesos críticos se ejecutan en un ambiente aislado al Sistema Operativo ayuda a proteger la información de los demás procesos, también es una técnica bastante efectiva contra la explotación; impidiendo que en caso de ejecutar algún tipo de ataque de Ejecución de Código Remota éste código sea realmente desplegado como un usuario con privilegios.

[177] FINDLAY, Andrew. Best Practices in LDAP Security [online]. Apache, sept., 2011. [citado 10 may., 2018]. Disponible en Internet: < https://people.apache.org/~elecharny/ldapcon/Andrew%2520Findlay-paper.pdf >

En la actualidad muchos procesos críticos, como es el caso de los Navegadores Web suelen implementar estas técnicas para prevenir que un atacante pueda hacerse con el control total de una máquina o mitigar daños que puedan comprometer la memoria y contexto de otros procesos[178].

b. Realizar planes de actualizaciones de manera periódica. Agendar actualizaciones semanal y mensualmente es un mecanismo efectivo para mantener a raya las amenazas emergentes. Actualmente la mayoría de los proyectos que cuentan con soporte, actúan de forma rápida y adecuada frente a los riesgos de seguridad. Aunque en algunas ocasiones las soluciones a las vulnerabilidades pueden no ser pertinentes y adecuadas al ambiente organizacional; se recomienda mantener las últimas versiones estables del software que presta el servicio de red.

   Esta acción de defensa puede ser efectiva incluso para los Sistemas Operativos y cualquier tipo de software en general.

c. Configurar *Port Knocking* para los servicios críticos. El *Port Knocking* es una técnica conocida aplicada en las reglas de *Firewall,* consiste en únicamente abrir un puerto cuando se han tocado otra serie de puertos en un orden preciso. Fue aplicado durante la ejecución del proyecto para asegurar que el verdadero servicio de *SSH* no fuese descubierto ni atacado por diversos tipos de amenazas.

   Esta medida de defensa es especialmente efectiva cuando se trata de hacer frente a Ataques de Fuerza Bruta y Denegación de Servicios basados en vulnerabilidades del protocolo de transporte como *TCP SYN* o *UDP Flooding.*

d. Instalar y configurar Sistemas de Monitoreo como *IDPS* y Antivirus. Los Sistemas de Detección y Prevención de Intrusos, como se ha podido observar en el proyecto, pueden ayudar a identificar diferentes tipos de amenazas de manera automatizada por medio del análisis de tráfico de red, esto puede suceder en tiempo real o incluso se puede configurar para analizar tramas que hayan sido capturadas anteriormente por *sniffers.* Monitorear la red en su totalidad puede ser de gran ayuda ya que se identificarían otros tipos de amenazas más generalizados, además de los servicios que se han presentado en el proyecto.

   Así mismo, estos los *IDPS* basados en *Host* y los Antivirus; son herramientas potentes contra la detección de puertas traseras, consecuencias que pueden suceder en una etapa de *Post-Exploitation* de alguna vulnerabilidad de un servicio de red.

[178] MOZILLA. Op. cit., 145 p.

e. Habilitar niveles de auditoría de los Servicios de Red. Configurar los diferentes proveedores y *software* que ofrecen los Servicios de Red para así registrar su actividad mediante archivos de *logs.* Estos archivos posteriormente pueden ser procesados para encontrar comportamientos atípicos e identificar posibles problemas. La revisión de los archivos de auditoría generados, debería ser aplicada al menos una vez al día.

f. Aplicar niveles de seguridad en las Capas de Enlace de Datos. Diseñar una red para ser segura, desde el plano y la distribución de subredes, definir la configuración de *Vlans,* separar las redes y recursos públicos o privados, son conceptos básicos que pueden ayudar a mitigar la expansión de las amenazas o incluso prevenir su ocurrencia.

g. Defensa en Profundidad (*Defence in Deep*). Este es un concepto aplicado en ámbitos militares, pero puede ser trasladado al campo de la Seguridad Informática. Establecer varios controles tanto a nivel de red, como servidores, *software,* configuración y despliegue; puede ayudar a prevenir la expansión de una amenaza. La Defensa en Profundidad consta en definir varias capas de seguridad que se pueden implementar para acceder a un recurso. Por ejemplo, un Servidor Web podría ofrecer un primer nivel de seguridad, el cual sea conectarse a una *Vlan* específica para enlazar su conexión al servidor, posteriormente se podría ejecutar un *Port Knocking* para desbloquear el servicio, luego se debería aplicar una autenticación para acceder a una Aplicación Web específica y finalmente se debe contar con la *URI* la cual puede ser aleatoria para acceder a la aplicación.

   En un principio esta estrategia podría parecer algo exagerada, pero suele ser el mecanismo más efectivo contra amenazas de todo tipo. Incluso este concepto puede aplicarse a nivel de Seguridad Lógica y Seguridad Física.

10 CONCLUSIONES

- El proveedor de VPS Hostinger implementa una virtualización por medio de OpenVZ con imágenes del kernel personalizadas para cada Sistema Operativo, durante el montaje de los HoneyPots esto fue una limitante debido que no fue posible instalar cierto tipo de *software* y otras dependencias requeridas por HoneyPots de versiones recientes.

- Snort, Maldet y TcpDump son herramientas de licencias Open Source y/o libres que permitieron la corrobaración de los datos arrojados por los HoneyPots examinando el tráfico de red malicioso y otras actividades sospechosas dentro del VPS.

- Los servicios más atacados han sido HTTP y SSH, el primero de ellos muestra una gran variedad de vectores de ataque que intentó aprovechar vulnerabilidades de Aplicaciones Web mientras en el segundo se observó una predilección por los Ataques de Fuerza Bruta. Ambos servicios son bastante conocidos y usados en las redes de computadoras por ello se cree que han presentado una alta incidencia de ataques.

- Las acciones de hardening recomendadas para fortalecer la seguridad de los servicios han sido redactadas de forma general sin hacer referencia a ambientes organizacionales específicos, debido a esto su implementación puede variar según el contexto y red que se disponga. A su vez, es posible que en algunos casos estas acciones no puedan implementarse en su totalidad.

- Se ha verificado los ataques de los servicios para los protocolos HTTP, LDAP, SSH y SMTP por medio de la configuración y uso de HoneyPots y en base a ello se han redactado una serie de acciones de hardening que pueden ayudar a mejorar la Seguridad de la Información en una organización. Este trabajo demuestra un proceso de análisis completo cimentado en los resultados de los HoneyPots.

11 RECOMENDACIONES

- Para configurar un ambiente de HoneyPots a nivel organizacional, se recomienda examinar el nivel de riesgo al cual podría estar expuesta la organización en caso que el HoneyPot llegase a fallar. El autor recomienda no implementar herramientas obsoletas, ya que este riesgo puede aumentar significativamente.

- El uso de herramientas de monitorización como los Sistemas de Detección y Prevención de Intrusos IDPS son necesarios en las organizaciones. Es preferible leer un poco acerca de la configuración de este tipo de software, aunque sea libre y no cuente con un soporte pago, que dejar a una red organizativa únicamente protegida por un Firewall. Es necesario recordar en este punto que los Firewall sólo bloquean y desbloquean puntos de entrada o salida, o pueden actuar como *routers* en ciertos casos; pero nunca analizan la comunicación a fondo.

- Si el lector desea desarrollar una herramienta HoneyPot para un servicio de red, el autor recomienda documentarse muy bien sobre el protocolo para poder abarcar cualquier posible punto de entrada. Entre más capacidades del protocolo se tengan en cuenta, el HoneyPot estará más completo y menos expuesto a ciertos tipos de amenaza. Se recomienda de igual manera se consulta sobre técnicas de *sandboxing* y aislamiento de procesos.

- Muchas de las acciones de hardening fueron extraídas de fuentes especializadas en el tema. Se recomienda que siempre que se deseen aplicar políticas de seguridad nuevas, se consulte a nivel organizacional sobre el posible impacto que puede generar esta nueva política, así mismo asesorarse sobre fuentes oficiales para no incurrir en la apertura de una brecha de seguridad.

- Algunos de los HoneyPots se configuraron haciendo uso de *software* abandonado, es decir que no cuenta con soporte. Esto puede traer implicaciones negativas en la Seguridad de la Información si se quiere desplegar los HoneyPots en redes organizacionales no aisladas. Por ello es altamente recomendado comprobar los términos de soporte y mantenibilidad de un *software* de HoneyPot antes de implementarlo en ambientes productivos.

12 REFERENCIAS Y BIBLIOGRAFÍA

ALVARES, Carlos. Aspectos penales relativos al uso de “Honeypots”. Legal Legis. Colombia: dic 2003. 20 p.

ANLEY, Chris; HEASMAN, John; RICHARTE, Gerardo et al. The Shellcoder’s Handbook, Discovering and Exploiting Security Holes. En: Chapter 1 Before you Begin. 2 ed. Estados Unidos: Wiley Publishing Inc, 2007. p 3-10. ISBN 978-0-470-08023-8.

AWHITEHATTER. Mailoney SMTP HoneyPot [online]. Github jul, 2016. [citado 20 mar.,2018]. Disponible en Internet: < https://github.com/awhitehatter/mailoney >

BARRET, Daniel y SILVERMAN Richard. Threats that SSH Doesn’t Prevent [online]. SSH: The Secure Shell The Definitive Guide. O’Reilly ene, 2001. [citado 21 mar., 2018]. Disponible en Internet: < https://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch03_11.htm >

BELL, Simon. Building a HoneyPot to Research Cyber-Attack Techniques. Reino Unido: Faculty of Computer Systems and Software Engineering, University College of Engineering and technology mar, 2014. 69 p.

BOUCHERON, Brian. How to Install and Configure OpenLDAP and phpLDAPadmin on Ubuntu 16.04 [online]. Digital Ocean jun., 2017. [citado 27 mar., 2018]. Disponible en Internet: < https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-a-basic-ldap-server-on-an-ubuntu-12-04-vps >

BRANDON, John. Why your Business might be a Perfect target for Hackers [online]. Inc.com nov, 2017. [citado 21 mar., 2018]. Disponible en Internet: < https://www.inc.com/magazine/201312/john-brandon/hackers-target-small-business.html >

CHAKRABORTY Nilotpal. Intrusion Detection System and Intrusion Prevention System: A Comparative Study [online]. India. Mayo, 2013. [citado 13 mar., 2018]. Capítulo 5: Intrusion Detection System y Capítulo 6: Intrusion Prevention System. Disponible en Internet: <

https://pdfs.semanticscholar.org/c91d/d6f155cdbd0c72017a8413c74f1953b517c3.pdf >

CHESWICK Bill. An Evening with Berfer In Which a Cracker is Lured, Endured and Studied. Nueva Jersey: AT&T Bell Laboratories, 1991. 11 p.

COLOMBIA, CONGRESO DE LA REPUBLICA. Ley 1266. Bogotá. (Diciembre 31 de 2008). Diario Oficial 47.219 de diciembre 31 de 2008. p. 1-15.

COMMON VULNERABILITIES AND EXPOSURES CVE. Vulnerabilities Details CVE-2017-10271 [online]. Common Vulnerabilities and Exposures CVE feb., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.cvedetails.com/cve/CVE-2017-10271/ >

CONGRESO DE LA REPÚBLICA DE COLOMBIA. Ley 1273 de 2009 Nivel Nacional [online]. Alcaldía de Bogotá ene, 2009. [citado 21 mar., 2018]. Disponible en Internet: < http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=34492 >

CONSTANTIN, Lucian. Attackers are now abusing exposed LDAP servers to amplify DDoS attacks [online]. IDG News Service, oct., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://www.pcworld.com/article/3135771/security/attackers-are-now-abusing-exposed-ldap-servers-to-amplify-ddos-attacks.html >

CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3) [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2251 >

CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2252 >

CRITICAL ANGLE INC, NETSCAPE COMMUNICATIONS CORP, ISODE LIMITED et al. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2253 >

CRITICAL ANGLE INC. A Summary of the X.500(96) User Schema for use with LDAPv3 [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2256 >

CROCKER, David. Standard for the Format of ARPA Internet Text Messages [online]. Internet Engineering Task Force IETF ago, 1982. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc822 >

CYBSEC. Seguridad Informática HoneyPots [online]. CYBSEC jul, 2008. [citado 13 mar., 2018]. 3. Ubicación. Disponible en Internet: < http://www.cybsec.com/upload/ESPE_Honeypots.pdf >

DESASTER. Kippo SSH HoneyPot [online]. Github mar, 2010. Github. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/desaster/kippo >

DEUTSCHE TELEKOM AG. T-Pot: A Multi-HoneyPot Platform [online]. Github mar, 2015. [citado 20 mar., 2018]. Disponible en Internet: < http://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html >

EICAR. European Expert Group For IT-Security [online]. Eicar mar., 2018. [citado 1 abr., 2018]. Disponible en Internet: < http://www.eicar.org/ >

ESET. ¿Qué es un 0-day? Explicando términos de seguridad [online]. We live Security feb, 2015. [citado 22 mar., 2018]. Disponible en Internet: < https://www.welivesecurity.com/la-es/2015/02/25/que-es-un-0-day/ >

ESPAÑA, GOBIERNO DE LA REPÚBLICA. MAGERIT – Versión 3.0 Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información, Libro 1 – Método [online]. 1 ed. España. Octubre, 2012. [citado 13 mar., 2018]. Método de Análisis de Riesgos, Conceptos paso a paso. Disponible en Internet: < https://www.ccn-cert.cni.es/documentos-publicos/1789-magerit-libro-i-metodo/file.html >

ESTRELLA GUIJIJE, Gustavo. Diseño del Prototipo de una HoneyPot Virtual que permitirá mejorar el Esquema de Seguridad en Redes de la Carrera Ingeniería en Sistemas Computacionales y Networking de la Universidad de Guayaquil. Ecuador: Universidad de Guayaquil sept, 2011. 370 p.

FABRIC8. Process Container [online]. Fabric8 oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://fabric8.io/gitbook/processContainer.html >

FINDLAY, Andrew. Best Practices in LDAP Security [online]. Apache, sept., 2011. [citado 10 may., 2018]. Disponible en Internet: < https://people.apache.org/~elecharny/ldapcon/Andrew%2520Findlay-paper.pdf >

FRUHLINGER Josh. What is Stuxnet, who created it and how does it work [online]. Agosto, 2017. [citado 13 mar., 2018]. Disponible en Internet: < https://www.csoonline.com/article/3218104/malware/what-is-stuxnet-who-created-it-and-how-does-it-work.html >

GLASLOS. Glastopf Web Application HoneyPot [online]. Github feb, 2014. Github. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/glastopf >

GMAIL. OAuth 2.0 Mechanism [online]. Google oct., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://developers.google.com/gmail/imap/xoauth2-protocol >

GOOGLE INC, MOZILLA, BITGO et al. HyperText Transfer Protocol Version 2 [online]. Internet Engineering Task Force IETF may, 2015. [citado 20 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc7540 >

GOOGLE y PING IDENTITY. OAuth 2.0 for Native Apps [online]. Internet Engineering Task Force IETF oct, 2017. [citado 22., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc8252 >

HIGTY. Understanding the Insides of the SMTP Mail Protocol: Part 1 [online]. CodeProject dic, 2012. [citado 22 mar., 2018]. Disponible en Internet: < https://www.codeproject.com/Articles/399207/Understanding-the-Insides-of-the-SMTP-Mail-Protocol >

HOSTINGER COLOMBIA. Términos y Condiciones de Uso del Servicio [online]. Hostinger mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://www.hostinger.co/terminos-uso >

INNOSOFT INTERNATIONAL INC, DOVER BEACH CONSULTING INC, NETWORK MANAGEMENT ASSOCIATES INC et al. SMTP Service Extensions [online]. Internet Engineering Task Force IETF nov, 1995. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc1869 >

International Organitation for Standardization ISO. ISO/IEC 7498-1:1994 Information Technology – Open System Interconnection – Basic Reference Model: The Basic Model [online]. Switzerland. Junio, 1996. [citado 13 mar., 2018]. Introduction to Open Systems Interconnection (OSI), Definitions. Disponible en Internet: < https://www.iso.org/standard/20269.html >

INTERNET MAIL CONSORTIUM. SMTP Service Extension for Secure SMTP over Transport Layer Security [online]. Internet Engineering Task Force IETF feb, 2002. [citado 22 mar., 2018]. Disponible en Internet: < https://www.ietf.org/rfc/rfc3207.txt >

JAYANDCATCHFIRE y SINDRESOURHUS. Awesome list [online]. Github mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/sindresorhus/awesome >

JURADO PALLARÉS, Diego. Trabajo de Fin de Grado, Análisis y Estudio de Honeypots Complejos: Honeynets. Madrid: Universidad Autónoma de Madrid. Escuela Politécnica Superior. Departamento de Ingeniería Informática. 2016. 93 p.

KARPESKY. What is a Tunneling Protocol? [online]. Marzo, 2018. [citado 13 mar., 2018]. Disponible en Internet: < https://www.kaspersky.com/resource-center/definitions/tunneling-protocol >

KILI, Aaron. Restrict SSH User Access to Certain Directory Using Chrooted Jail [online]. Tecmint mar., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.tecmint.com/restrict-ssh-user-to-directory-using-chrooted-jail/ >

KLENSIN. Simple Mail Transfer Protocol [online]. Internet Engineering Task Force IETF oct, 2008. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc5321 >

LAKHANI, Amit. Deception Techniques Using HoneyPots. Londres: University of London. 2007 p 46-49.

LINFO. Kernel Definition [online]. Linux Information Project mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < http://www.linfo.org/kernel.html >

LINFO. Kernel Space Definition [online]. Linux Information Project mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < http://www.linfo.org/kernel_space.html >

MAURER, Ashley. Virtual Private Servers – An Introduction (And 3 Reasons You Might Need One). Abril, 2017. [citado 13 mar., 2018]. Disponible en Internet: < https://www.a2hosting.com/blog/virtual-private-servers/ >

MICHELOOSTERHOF. Cowrie HoneyPot [online]. Github ene., 2018. [citado 26 mar., 2018]. Disponible en Internet < https://github.com/micheloosterhof/cowrie >

Microsoft Azure. What is cloud computing? A beginner's guide [online]. Marzo, 2018. [citado 13 mar., 2018]. Uses of cloud computing. Disponible en Internet: < https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/ >

MICROSOFT. Change the level of protection in the Junk Email Filter [online]. Microsoft, jun., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://support.office.com/en-us/article/change-the-level-of-protection-in-the-junk-email-filter-e89c12d8-9d61-4320-8c57-d982c8d52f6b >

MICROSOFT. Common Types of Network Attacks [online]. Microsoft oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://technet.microsoft.com/en-us/library/cc959354.aspx >

MICROSOFT. How POP3 Service Works [online]. Microsoft mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc737236(v=ws.10) >

MICROSOFT. Proteger los mensajes con una firma digital [online]. Microsoft, jun., 2016. [citado 10 may., 2018]. Disponible en Internet: < https://support.office.com/es-es/article/proteger-los-mensajes-con-una-firma-digital-549ca2f1-a68f-4366-85fa-b3f4b5856fc6 >

MICROSOFT. Security Threats [online]. Microsoft oct, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://msdn.microsoft.com/en-us/library/cc723507.aspx >

MICROSOFT. SMTP Security [online]. Microsoft mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://msdn.microsoft.com/en-us/library/cc505927.aspx >

MICROSOFT. User mode and Kernel mode [online]. Microsoft abr, 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode >

MKRUL. Spam HoneyPot Tool [online]. Github jun, 2015. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/miguelraulb/spamhat >

MONGODB. User Management Methods, db.createUser() [online]. MongoDB mar., 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://docs.mongodb.com/manual/reference/method/db.createUser/ >

MOZILLA. Sandbox Architecture of Firefox [online]. Mozilla mar, 2018. Security/Sandbox/Process model. [citado 22 mar., 2018]. Disponible en Internet: < https://wiki.mozilla.org/Security/Sandbox/Process_model >

MUÑOZ, Alfonso; AGUIRRE, Jorge. Cifrado de las comunicaciones digitales: De la cifra clásica al algoritmo RSA. En: Capítulo 1 Introducción. España: 0xword, 2011 p 13 – 15. ISBN 978-84-616-3124-7.

MUSCAT, Ian. What is Server Side Request Forgery (SSRF)? [online]. Acunetix ma., 2017. [citado 10 ma., 2018]. Disponible en Internet: < https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/ >

MUSHORG. Better Function Replacer base don APD [online]. Github oct., 2015. [citado 22 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/BFR >

MUSHORG. Glastopf Installation – Debian Squeeze [online]. Github mar., 2016. [citado 22 mar., 2018]. Disponible en Internet: < https://github.com/mushorg/glastopf/blob/master/docs/source/installation/installation_debian.rst >

NEEMANY, Eyal. Honeypot Buster: A Unique Red-Team Tool [online]. Javeli Networks jul, 2017. [citado 20 mar., 2018]. Disponible en Internet: < https://jblog.javelin-networks.com/blog/the-honeypot-buster/ >

NETSCAPE COMMUNICATIONS CORP. The LDAP URL Format [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2255 >

NETSCAPE COMMUNICATIONS CORP. The String Representation of LDAP Search Filters [online]. Internet Engineering Task Force IETF dic, 1997. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2254 >

NIXCRAFT. How to Format Date for Display or Use in a Shell Script [online]. Nixcraft, mar., 2016 [citado 1 abr., 2018]. Disponible en Internet: < https://www.cyberciti.biz/faq/linux-unix-formatting-dates-for-display >

NIXCRAFT. How to install PHP 7 on Debian Linux 8.7/7.x Jessie/Wheezy [online]. Cyberciti feb., 2017. [citado 22 mar., 2018]. Disponible en Internet: < https://www.cyberciti.biz/faq/installing-php-7-on-debian-linux-8-jessie-wheezy-using-apt-get/ >

ORACLE. Chapter 21 Information_Schema Tables [online]. MySQL Development ene., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://dev.mysql.com/doc/refman/5.6/en/information-schema.html >

ORACLE. X.500 Standard [online]. Oracle mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://docs.oracle.com/javase/jndi/tutorial/ldap/models/x500.html >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Access Control Cheat Sheet [online]. OWASP ene, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Access_Control_Cheat_Sheet >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Blocking Brute Force Attacks [online]. OWASP mar, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Broken Access Control [online]. OWASP mar, 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Broken_Access_Control >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Cross-Site Request Forgery CSRF Prevention Cheat Sheet [online]. OWASP mar, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Cross-site Scripting (XSS) [online]. OWASP mar., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. OWASP Top 10 [online]. OWASP ene, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Path Transversal [online]. OWASP jun., 2015. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Path_Traversal >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. SQL Injection Prevention Cheat Sheet [online]. OWASP feb, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. Web Application Firewall WAF [online]. OWASP ene, 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/Web_Application_Firewall >

OWASP OPEN WEB APPLICATION SECURITY PROYECT. XSS (Cross Site Scripting) Prevention Cheat Sheet [online]. OWASP may., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet >

PANDA SECURITY. Enciclopedia de Virus Stuxnet. [online]. Junio, 2010. [citado 13 mar., 2018]. Disponible en Internet: < https://www.pandasecurity.com/peru/homeusers/security-info/222123/information/Stuxnet >

PARALAX. Awesome HoneyPots [online]. Github mar, 2018. [citado 20 mar., 2018]. Disponible en Internet: < https://github.com/sindresorhus/awesome >

PETER, Eric y SCHILLER Tood. A Practical Guide to HoneyPots [online]. Washington D.C. Mayo, 2009. [citado 13 mar., 2018]. Disponible en internet: < https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/index.html >

PHPMYADMIN. Bringing MySQL to the Web About [online]. PhpMyAdmin oct., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.phpmyadmin.net/ >

POSEY, Brein. 10 common Network Security design flaws [online]. Tech Republic oct, 2009. [citado 21 mar., 2018]. Disponible en Internet: < https://www.techrepublic.com/blog/10-things/10-common-network-security-design-flaws/ >

POSTEL, Jonathan. Simple Mail Transfer Protocol [online]. Internet Engineering Task Force IETF ago, 1982. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc821 >

POSTFIX. Postfix SASL Howto [online]. Postfix sep., 2017. [citado 10 may., 2018]. Disponible en Internet: < http://www.postfix.org/SASL_README.html >

POSTFIX. Postfix Tunning Performance [online]. Postfix sep., 2017. [citado 10 may., 2018]. Disponible en Internet: < http://www.postfix.org/TUNING_README.html >

PRITCHARD, Chris. Shiva the Spam HoneyPots: Tips and Tricks for getting it up and running [online]. Pentest Partners jun., 2015. [citado 26 mar., 2018]. Disponible en Internet: < https://www.pentestpartners.com/security-blog/shiva-the-spam-honeypot-tips-and-tricks-for-getting-it-up-and-running/ >

PROTONMAIL. What is encrypted? [online]. ProtonMail mar., 2018. [citado may., 2018]. Disponible en Internet: < https://protonmail.com/support/knowledge-base/what-is-encrypted/ >

QUALCOMM INCORPORATED. Internet Message Format [online]. Internet Engineering Task Force IETF oct, 2008. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc5322 >

RAIU, Costin; GUERRERO S, Juan A y BAUMGARTNER, Kurt. Kaspersky Security Bulletin: Threat Predictions for 2018 [online]. Secure List nov, 2017. [citado 21 mar., 2018]. Disponible en Internet: < https://securelist.com/ksb-threat-predictions-for-2018/83169/ >

RAJ. Install Linux Malware Detecto on Debian / Ubuntu / LinuxMint – A Malware Scanner for Linux Operating System [online]. IT'z Geek, nov., 2017 [citado 1 abr., 2018]. Disponible en Internet: < https://www.itzgeek.com/how-tos/linux/debian/install-linux-malware-detect-on-debian-ubuntu-linuxmint-a-malware-scanner-for-linux-operating-system.html >

RAMDEV. Ten most frequently used Linux networking services, in enterpirse unix networks [online]. Unix admin schoo may, 2015. [citado 21 mar., 2018]. Disponible en Internet: < http://unixadminschool.com/blog/2012/05/ten-most-frequently-used-inux-networing-services-in-enterprise-networks/ >

REDHAT. 7.9. Configuring PAM for Auditing [online]. Redhat Customer Portal oct., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sec-configuring_pam_for_auditing >

REDHAT. El concepto de la virtualización [online]. Red Hat oct, 2016. [citado 22 mar., 2018]. Disponible en Internet: < https://www.redhat.com/es/topics/virtualization >

RESNICK. Internet Message Format [online]. Internet Engineering Task Force IETF abr, 2001. [citado 22 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2822 >

R-FX NETWORKS. Linux Malware Detect [online]. R-fx Networks, mar., 2017. [citado 1 abr., 2018]. Disponible en Internet: < https://www.rfxn.com/projects/linux-malware-detect/ >

RIVERO, Marcelo. ¿Qué son los Malwares? [online]. Info Spyware oct, 2017. [citado 22., 2018]. Disponible en Internet: < https://www.infospyware.com/articulos/que-son-los-malwares/ >

ROSENBLUM, Mendel. Web Servers [online]. Stanford mar, 2016. [citado 20 mar., 2018]. Disponible en Internet: < https://web.stanford.edu/class/cs142/cgi-bin/slides/WebServers.pdf >

SHARIFI, Ahmad; NOOROLLAHI, Akram y FAROKHMANESH Farnoosh. Intrusion Detection and Prevention Systems (IDPS) and Security Issues [online]. International Journal of Computer Science and Network Security nov, 14. [citado 22 mar., 2018]. Vol 14, no. 11. Disponible en Internet: < https://pdfs.semanticscholar.org/04da/5558cc056746a2edba2a7173ebb7c4d67807.pdf >

SHERSTHA Narad. Use Pam_Tally2 to Lock and Unlock SSH Failed Login Attempts [online]. Tecmint abr., 2013. [citado 10 may., 2018]. Disponible en Internet: < https://www.tecmint.com/use-pam_tally2-to-lock-and-unlock-ssh-failed-login-attempts/ >

SHIVA-SPAMPOT. Shiva HoneyPot [online]. Github jun., 2016. [citado 26 mar., 2018]. Disponible en Internet: < https://github.com/shiva-spampot/shiva >

SITEGROUND. Email Protocols – POP3, SMTP and IMAP Tutorial [online]. SiteGround mar, 2018. [citado 21 mar., 2018]. Disponible en Internet: < https://www.siteground.com/tutorials/email/protocols-pop3-smtp-imap/ >

SNORT. Snort Documents [online]. Snort downloads, ene., 2018. [citado 1 abr., 2018]. Disponible en Internet: < https://www.snort.org/downloads >

SNORT. Snort Documents [online]. Snort official page, ene., 2018. [citado 1 abr., 2018]. Disponible en Internet: < https://www.snort.org/documents >

SOKOL, Pavol; PEKARČK Patrik y BAJTOŠ Tomáš. Data Collection and Data Analysis in HoneyPots and Honeynets. Košice: Facultad de Ciencias de la Computación Universidad de Pavol Jozef Safárik, 2015. 16 p.

SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Transport layer Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4253 >

SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Connection Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4254 >

SSH COMMUNICATIONS SECURITY CORP y CISCO SYSTEMS INC. The Secure Shell (SSH) Authentication Protocol [online]. Internet Engineering Task Force IETF ene, 2006. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc4252 >

STALLINGS, William. Protocol Basics: Secure Shell Protocol [online]. The Internet Protocol Journal Cisco Systems dic, 2009. [citado 21 mar., 2018]. Vol. 12, no. 4. Disponible en Internet: < https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html >

STOLL, Cliff. The Cuckoo's Egg. Cambridge: Universidad de Harvard, 1998. 254 p.

SUN MICROSYSTEMS INC, EDB MAXWARE, OBLIX INC y Otros. Authentication Methods for LDAP [online]. Internet Engineering Task Force IETF may, 2000. [citado 21 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2829 >

TAKHION. Use the Cowrie SSH Honeypot to Catch Attackers on your Network [online]. Null byte ene., 2018. [citado 26 mar., 2018]. Disponible en Internet: < https://null-byte.wonderhowto.com/how-to/use-cowrie-ssh-honeypot-catch-attackers-your-network-0181600/ >

TANLI, Ilker; KOLCALAR, Turgut. Deception Techniques, Methods, Honeypots, Honeynets and Usage. MontClair. Inglaterra: jun, 2017. 10 p.

TCPDUMP. TcpDump Man Page [online]. TcpDump. Feb., 2017 [citado 1 abr., 2018]. Disponible en Internet: < https://www.tcpdump.org/tcpdump_man.html >

TECHOPEDIA. Computer Network Definition [online]. Techopedia mar, 2018 [citado 22 mar., 2018]. Disponible en Internet: < https://www.techopedia.com/definition/25597/computer-network >

TECHOPEDIA. Firmware Definition [online]. Techopedia mar, 2018 [citado 22 mar., 2018]. Disponible en Internet: < https://www.techopedia.com/definition/2137/firmware >

TELEGRAM. Bots: And Introduction for developers [online]. Telegram mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://core.telegram.org/bots >

TIVOLI SOFTWARE. Understanding LDAP Design and Implementation. IBM: United States. 2004. 774 p.

TREND MICRO. New Apache Stryts Vulnerability Could Be Worse tan Poodle [online]. Trend Micro sept., 2017. [citado 10 may., 2018]. Disponible en Internet: < https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-xxploits/new-apache-struts-vulnerability-could-be-worse-than-poodle >

TUTORIELS MEDDEB. Enable the production of Openldap Log File [online]. Tutoriels Meddeb sept., 2017. [citado 27 mar., 2018]. Disponible en Internet: < http://tutoriels.meddeb.net/openldap-tutorial-log/ >

U.S.A GOVERNMENT. 18 U.S. Code Chapter 119 – Wire and Electronic Communications Interception and Interception of Oral Communications [online]. Legal Information Institute, 2015. [citado 21 mar., 2018]. Disponible en Internet: < https://www.law.cornell.edu/uscode/text/18/part-I/chapter-119 >

UNIÓN EUROPEA. Reglamento General de Protección de Datos [online]. ESET feb., 2018. [citado 21 mar., 2018]. Disponible en Internet: < https://gdpr.eset.es/ >

VIRTUOZZO CONTAINERS. OpenVZ Get Started and Instalation [online]. Virtuozoo Containers mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://openvz.org/Main_Page >

VULTR. How to Configure Snort on Debian [online]. Vultr Docs, ene 2016. [citado 1 abr., 2018]. Disponible en Internet: < https://www.vultr.com/docs/how-to-configure-snort-on-debian >

VUSAL Aliyev. Using honeypots to study skill level of attackers base don the exploited vulnerabilities in the network. Göteborg: Department of Computer Science and Engineering Division of Computer Security, Chalmers University of Technology. 2010. 69 p.

WANG, Ping; WU, Lei; CUNNINGHAM, Ryan et al. Honeypot detection in advanced botnet attacks. International Journal of Computer Science and Network Security feb, 2010. Vol. 4, issue 1. P 30 – 51.

WIKIPEDIA. Operating System [online]. Wikipedia mar, 2018. [citado 22 mar., 2018]. Disponible en Internet: < https://en.wikipedia.org/wiki/Operating_system >

XEROS, W3C/MIT, MICROSOFT et al. HyperText Transfer Protocol HTTP 1.1 [online]. Internet Engineering Task Force IETF jun, 1999. [citado 20 mar., 2018]. Disponible en Internet: < https://tools.ietf.org/html/rfc2616 >

YLONEN Tatu. Ssh Key [online]. Ssh.com mar., 2018. [citado 10 may., 2018]. Disponible en Internet: < https://www.ssh.com/ssh/key/ >

YUNUS, Mammon. Online LDAP Test Server [online]. Forum System The Leader in API Security Management feb, 2014. [citado 22 mar., 2018]. Disponible en Internet: < https://www.forumsys.com/tutorials/integration-how-to/ldap/online-ldap-test-server/ >

ZOU Cliff, CUNNINGHAM Ryan. Honey-Aware Advanced Botnet Construction and Maintenance. Orlando: School of Electrical Engineering and Computer Sciente, University of Central Florida. 2009. 22 p.