UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

# Análisis y clasificación de los ataques y sus exploits: Framework Metasploit como caso de estudio

# Attacks and exploits analysis and classification: Case Study - Metasploit Framework


Realizado por
**Ana Tinoco Linares**

Tutorizado por
**Ana Nieto Jiménez**
**Francisco Javier López Muñoz**

Departamento
**Dpto. de Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA


MÁLAGA, JUNIO DE 2020

Fecha defensa: octubre de 2020

# Resumen

La seguridad informática es un área fundamental de la informática que afecta tanto al ámbito público como al privado. Además, dado que la informática forma parte de numerosos contextos, surge la ciberseguridad como una disciplina más general, en la que pueden estar involucrados muchos perfiles profesionales, no sólo informáticos. La gran demanda de profesionales en este sector requiere un esfuerzo de formación de nuevos expertos en la materia, capaces de analizar vulnerabilidades y riesgos y procurar la seguridad en diversos entornos. No sólo perfiles técnicos son necesarios, sino también urge que otros profesionales puedan entender mejor las herramientas que usan los ciberatacantes para, por ejemplo, ayudar a perfilar las características de los individuos detrás del ataque.

Una de las herramientas más empleadas hasta la fecha es el framework Metasploit, del que existe numerosa información, incluyendo ejemplos sobre cómo se pueden explotar vulnerabilidades en máquinas diseñadas para practicar (vulnerables por defecto), como Metasploitable2 y Metasploitable3. Sin embargo, no encontramos imágenes gráficas que nos muestren cómo son el procedimiento y seguimiento de las técnicas de explotación utilizadas, ayudándonos a entender de forma clara los diferentes niveles de abstracción que podrían emplearse durante el análisis y explotación de vulnerabilidades. Es decir, con las herramientas actuales somos capaces de obtener una lista de ciertas vulnerabilidades de un sistema y atacarlas, pudiendo observar los resultados de dichos ataques, pero careciendo de una visión gráfica del proceso. Mientras que el objetivo de dichas herramientas es, precisamente, abstraernos de la complejidad de las vulnerabilidades y de sus exploits, el objetivo principal de este trabajo fin de grado (TFG) es precisamente el análisis detallado de dicha automatización, para comprender así cómo los exploits se forman, se combinan con payloads y se dirigen a explotar las vulnerabilidades del sistema atacado. Además, este análisis ayudará a fragmentar y analizar desde una perspectiva modular los diferentes elementos durante el proceso de explotación, permitiendo así a su vez explicar los componentes identificados a estudiantes de ciberseguridad durante su formación académica. La descomposición en módulos que aquí se realiza sirve para explicar un proceso automatizado – por su complejidad – de forma entendible. Su uso en materia educacional puede adaptarse con ello a diferentes perfiles de estudiantes y profesionales, tanto del sector de la informática como a otros que requieran comprender la casuística de los ataques y sus fases.

**Palabras clave:**

Vulnerabilidad, Exploit, Payload

# Abstract

Computer security is an essential part of computing that affects both private and public scopes. Moreover, computer science is present in numerous contexts and this is why cybersecurity came as a general discipline where there are different profiles involved, not only computer related ones. The large demand of professionals in cibersecurity requires a big effort to educate new experts in this field, capables of analizing risks and vulnerabilities and securing different environments. Not only do we need technical profiles in this area, but it is also necessary that other professionals can better understand the tools used by attackers in order to, for example, help identify the profiles and individual characteristics from those behind the attacks.

One of the most used tools is Metasploit framework, for which there is a lot of documentation, including examples on how vulnerabilities can be exploited within machines made for that purpose, like Metasploitable2 and Metasploitable3. However, we do not find graphical Images that can show us the procedure and tracking of exploiting techniques being used in such processes in order to help us understand more precisely the different abstraction levels that can be found during analysis and exploiting of vulnerabilities. That is, with these tools we are capable of obtaining a list of certain vulnerabilities in a system and attacking them, letting us see the results of those attacks, but with no graphical vision of the process. While the objective of these tools is, In fact, to abstract and simplify vulnerabilities and their exploits complexity, the main objective of this final project is Indeed to obtain a detailed analysis of that automatization, learning graphically how exploits are made, used, combined with payloads and directed to a target system in order to exploit its vulnerabilities. Moreover, this analysis will help study the distinct elements present during exploitation from a modular perspective, so it is possible to explain the identified components to students during their academic period. This modularity helps explain an automated process in a more understandable way. It's use in education can be adapted to different student and professional profiles, so it can reach computer related profiles or simply for those who would like to understand attacking and its phases.

**Keywords:**

Vulnerability, Exploit, Payload

# Índice

# Tabla de Ilustraciones

Dada la naturaleza gráfica de este trabajo, se hallan numerosas ilustraciones a lo largo de la presente memoria. Mediante la siguiente tabla se recogen todas ellas y se facilita la búsqueda por ilustración.

# 1. Introducción

En esta sección se exponen la motivación, los objetivos, la estructura y la metodología del presente trabajo.

En primer lugar, en el apartado de motivación se desvelan las razones por las cuales se escogen la línea de la ciberseguridad y en concreto el tema de este trabajo, orientado al aprendizaje de vulnerabilidades y de ataques, y por qué se requiere y es importante la investigación en esta área. En segundo lugar, los objetivos puntualizan qué es lo que se pretende conseguir con el trabajo y las necesidades que con él se cubren. A continuación, la metodología indica el procedimiento seguido para clasificar y escenificar vulnerabilidades y ataques con objeto de elaborar un modelo o plantilla representativa final. Por último, se obtiene una visión general de la estructura de la memoria, describiendo el resto de las secciones que la componen.

## 1.1. Motivación

Actualmente, observamos una tendencia a la digitalización masiva que se expande continuamente. En el mundo de las tecnologías de la información encontramos una amplia gama de servicios para prácticamente cualquier ámbito (social, laboral, etc.). Sin embargo, existe una serie de amenazas que ponen en riesgo a personas, empresas y demás usuarios de estas tecnologías. Por ello, es importante ser conscientes de que hay gran cantidad de vulnerabilidades, como la utilizada por el popular WannaCry para infectar ordenadores y distribuirse, siendo necesario tomar medidas de seguridad y formar expertos que eviten estas situaciones en la medida de lo posible, por ejemplo, efectuando pruebas sobre el entorno a proteger de forma controlada. Para ello surge el hacking ético.

El hacking ético o *pentesting* comprende un conjunto de técnicas y metodologías empleadas para realizar pruebas de ataques sin causar daño. Para ello, se aplican una serie de conocimientos y técnicas de forma controlada y se reportan las vulnerabilidades encontradas. Una vulnerabilidad es una debilidad o defecto en la seguridad de un sistema que lo pone en riesgo.

De las herramientas empleadas para el entrenamiento – y el desempeño profesional – de los nuevos especialistas, el framework Metasploit[1] es uno de los más populares. Dicha utilidad está presente en numerosas distribuciones Linux preparadas para el pentesting (p.ej. Kali, Caine, etc.) y también es empleada como base por otras herramientas (p.ej. Armitage). Además, integra otras herramientas que pueden combinarse con bases de datos de vulnerabilidades (p.ej. OpenVAS). Es, sin duda, una potente herramienta gratuita que puede ayudar al análisis y explotación de vulnerabilidades con la finalidad de mejorar la seguridad en los sistemas. Sin embargo, pese a que simplifica dicha labor, la automatización

[1] Herramienta de explotación y validación de vulnerabilidades. https://docs.rapid7.com/metasploit/

de los ataques efectuados dificulta comprender en su totalidad las vulnerabilidades y sus correspondientes ataques en toda su magnitud.

La curiosidad por los ataques informáticos y las formas de combatirlos son lo que ha marcado el camino del TFG en la línea de la ciberseguridad. Es de especial importancia conocer a qué estamos expuestos y por qué es posible que un tercero consiga atacar nuestros entornos, a veces incluso de forma totalmente sencilla y rápida. No sólo eso, sino que, además, es necesario que otros tipos de experto, no necesariamente técnicos, comprendan esta problemática.

Gracias a este TFG se conseguirá añadir una importante fuente de conocimiento que ayude a descubrir y comprender todo el proceso en el que se ve envuelto un ataque. Las nuevas herramientas para realizar ataques resultan a veces tan triviales, que la dificultad de entender el ataque cae en un segundo plano, y esto es un riesgo muy grave a la hora de entrenar profesionales que deben ser capaces de arreglar los agujeros de seguridad en los que se sustentan los ataques. Es necesario comprender la dinámica y componentes de un ataque para poder frenarlo. Esto es más complicado que el mero hecho de apretar un botón, y en el proceso de aprendizaje las imágenes son fundamentales. Por ello este TFG se centra en esquematizar la información recogida para este trabajo, proponiendo un modelo que sea intuitivo para profesionales de muy diverso perfil.

La NVD (*National Vulnerability Database*)[2] contiene más de 50,000 registros de vulnerabilidades y publica de promedio 13 nuevos a diario. Si bien es prácticamente imposible documentar absolutamente todas las vulnerabilidades, los ataques existentes y sus variantes (a lo ya conocido se añaden los llamados "zero-day"), sí que es viable basarse en los más utilizados, así como en los puertos más comunes, para poder así establecer criterios que ayuden a su clasificación y esquematización.

El hecho de invertir tiempo en investigar sobre las vulnerabilidades, exploits, payloads, etc. más usuales, recopilar la información y hallar la forma de estructurarla gráficamente constituye una ganancia para aquellos iniciados en la materia, al igual que para mí, que espero obtener una sólida base en esta área.

## 1.2. Objetivos

A continuación, se desglosan los cuatro objetivos principales de este trabajo:

**O1. Estudiar las distintas alternativas para facilitar el aprendizaje del framework Metasploit**

Durante esta fase, trabajos anteriores en esta área serán recopilados y tenidos en consideración a la hora de hacer el estudio pertinente sobre el *framework* Metasploit y su usabilidad para el entrenamiento en ciberseguridad. Así mismo, se tratará de encontrar herramientas similares que ofrezcan las mismas funcionalidades que buscamos con Metasploit, aunque puedan abordarse desde otro enfoque. Esto ayudará a tener una visión

[2] https://nvd.nist.gov/

más amplia sobre lo que estamos utilizando.

**O2. Análisis detallado de ataques y exploits a distintos niveles**

Resulta necesario profundizar en el proceso que se sigue durante el análisis de vulnerabilidades separando entre los diferentes niveles para su mejor comprensión, así como la lógica detrás de los exploits que son listados para atacar dichas vulnerabilidades, de tal forma que se confiera un estudio claro y detallado que contribuya a la comprensión del trasfondo del *framework* para *pentesting* en cuestión.

Se debe recopilar tanta información como sea posible sobre los exploits a estudiar y los ataques ligados a ellos (descripciones, acciones, comandos, metodología, etc). Una vez se consiga conocimiento suficiente para cada caso, debemos hacer también un análisis de lo encontrado. Esto ayudará a comprender y acotar el ámbito del TFG.

**O3. Clasificación de los distintos componentes involucrados en los vectores de ataque**

Deberán establecerse un conjunto de criterios de clasificación para agrupar los componentes involucrados en los ataques, de forma que ayude a su entendimiento. Por otro lado, se deben organizar cada una de las partes de un ataque, así como cada uno de los pasos que se siguen a la hora de explotar vulnerabilidades en conjuntos que serán escogidos atendiendo a los criterios mencionados anteriormente.

La clasificación se ayudará de elementos visuales tales como colores o formas, para que el resultado quede bien estructurado e intuitivo, como parte de un mismo contexto unificado.

**O4. Facilitar la comprensión del framework Metasploit para el entrenamiento de nuevos profesionales de ciberseguridad**

Este último objetivo requiere enfocar el trabajo hacia el aprendizaje, de modo que resulte sencillo entender cómo, de dónde y por qué obtiene Metasploit los resultados que muestra al usuario, prácticamente en su totalidad. Se pretende convertir el estudio en un recurso para aquellos que deseen conocer y entender de manera extendida la materia.

## 1.3. Metodología de trabajo empleada en el TFG

En este apartado se describen las fases por las que ha transcurrido el desarrollo del presente trabajo:

Definición de objetivos, Estudio del Arte, Montaje de un entorno seguro de pruebas, Identificación de vulnerabilidades y Elaboración de una tabla para la relación de puertos, vulnerabilidades y exploits.

*Se recomienda leer el Anexo A.A. y A.B. para familiarizarse con conceptos básicos que se mencionan recurrentemente.

### 1.3.1. Definición de objetivos

Esta primera fase recoge los objetivos anteriormente desglosados en el apartado 1.2: Estudiar las distintas alternativas para facilitar el aprendizaje del framework Metasploit, análisis detallado de ataques y exploits a distintos niveles, clasificación de los distintos componentes involucrados en los vectores de ataque y facilitar la comprensión del framework Metasploit para el entrenamiento de nuevos profesionales de ciberseguridad.

Los objetivos han marcado la dirección de este trabajo. Antes de comenzar con el primer prototipo era necesario tener claro que se pretende lograr un mecanismo de clasificación y escenificación de vulnerabilidades y ataques que consiga mejorar la comprensión de estos, teniendo en cuenta que va enfocado a un público con nivel de iniciación, y que para ello se necesita hacer un estudio y unas pruebas acordes. En resumen, lograr un método simple y descriptivo para facilitar el aprendizaje de vulnerabilidades y ataques haciendo un estudio profundo.

Lo primero es el estudio del contexto sobre el que se está trabajando y la búsqueda de trabajos que hay dentro de dicho contexto, con sus conclusiones. De esta forma se recogen en el Estudio del Arte nociones y carencias de otros trabajos y se evitan duplicados. Con este primer objetivo se confirmaron las herramientas a utilizar (Kali, Metasploit, etc.) al comprobar su efectividad y adecuación para el trabajo. Asimismo, se ha reafirmado la falta de estudios sobre representación de vulnerabilidades y ataques.

Por otro lado, el análisis de vulnerabilidades y exploits en este caso ha estado marcado por una serie de puertos escogidos previamente para estudiar. Para cada uno de ellos se ha seleccionado una vulnerabilidad y ataque específicos que son estudiados previo a su escenificación. Con esta información se reconocieron los elementos comunes a tener en cuenta para elaborar el primer prototipo.

Finalmente, el último objetivo busca facilitar la comprensión de Metasploit y de sus procesos como herramienta de pentesting. Gracias al estudio de Metasploit y la información disponible en la página oficial de Rapid7[3] se puede identificar su modo de funcionamiento y las relaciones de los elementos que intervienen en un ataque (Metasploit como caso de estudio). Así se completa este último objetivo y se puede obtener el modelo de diagrama deseado.

### 1.3.2. Estudio del Arte

En esta fase, se han encontrado diversos trabajos interesantes que han funcionado como "base de pirámide" sobre la que construir nuestro modelo. La búsqueda se ha realizado principalmente en base a los documentos disponibles en *Google Scholar*[4].

Como veremos en el capítulo dedicado al Estudio del Arte, en lo referente a las herramientas utilizadas se pudo confirmar que son las apropiadas para este tipo de

[3] https://www.rapid7.com/
[4] http://scholar.google.es/

estudios. De igual forma, se confirmó la falta de trabajos de esta índole que contribuyen a la formación básica o elemental sobre vulnerabilidades y sus exploits.

### 1.3.3. Montaje de un entorno seguro de pruebas

Para montar un entorno seguro se ha optado por los siguientes elementos:

- Máquina virtual Kali.
- Máquina virtual Metasploitable2
- Hipervisor VMware Workstation Pro

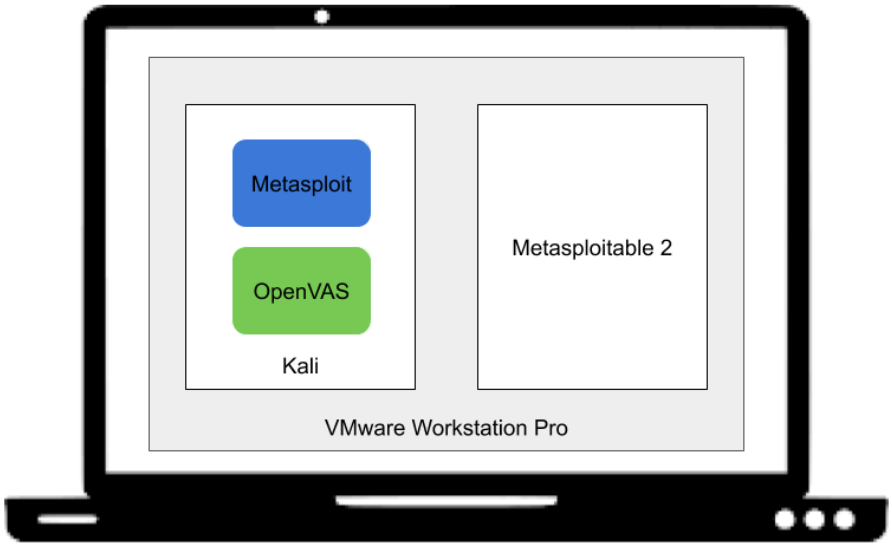



Ilustración 1. Entorno de pruebas. Componentes.

La máquina desde la que realizar las funciones de analista es Kali Linux, ya que es la que cuenta con las herramientas de pentesting preinstaladas. En concreto, para este trabajo se requieren Metasploit y OpenVAS[5]. Con OpenVAS se realiza el escaneo de vulnerabilidades y con Metasploit se efectúan los ataques a la máquina víctima para probar el aprovechamiento de las vulnerabilidades tal y como haría un atacante.

La máquina víctima es Metasploitable2[6], que viene preparada con múltiples vulnerabilidades con las que probar y aprender.

[5] Escáner de vulnerabilidades. https://www.openvas.org/
[6] https://docs.rapid7.com/metasploit/metasploitable-2

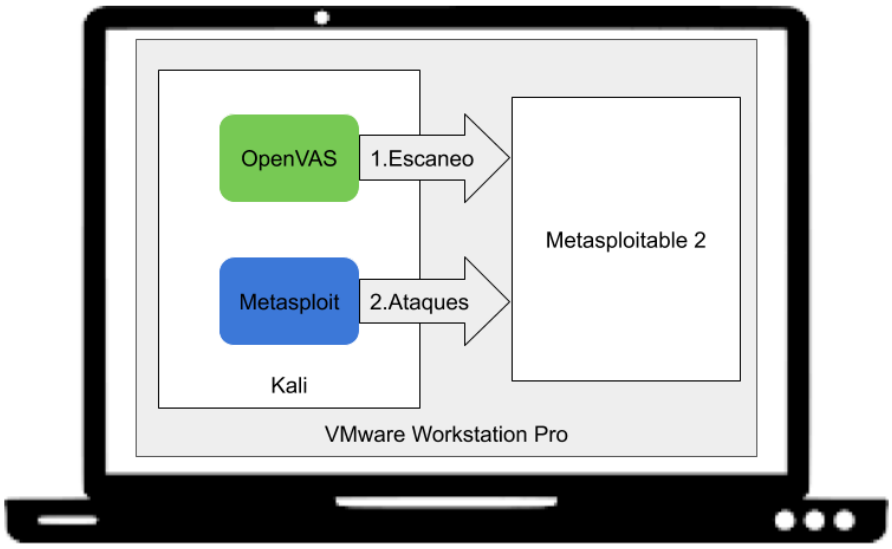



Ilustración 2. Entorno de pruebas. Acciones.

En el Apéndice A.C. se encuentra un manual de instalación para preparar este entorno de pruebas.

### 1.3.4. Identificación de vulnerabilidades

Como ya se ha indicado anteriormente, OpenVAS es el sistema de evaluación de vulnerabilidades (en inglés "Vulnerability Assessment System") utilizado para la identificación de vulnerabilidades. Es una herramienta que permite el escaneo de vulnerabilidades y la elaboración de informes de riesgo en distintos formatos, incluido el formato pdf.

Las evaluaciones de vulnerabilidades son revisiones sistemáticas de las debilidades de seguridad en un sistema de información. Se determina si el sistema es susceptible a las vulnerabilidades conocidas, se asignan niveles de riesgo a esas vulnerabilidades y se recomienda la corrección o mitigación [1].

En este caso, OpenVAS ofrece esas tres cosas características de una evaluación de vulnerabilidades: identifica las vulnerabilidades de la máquina víctima, indica el nivel de riesgo asociado según su CVSS (*Common Vulnerability Score System*) y agrega un apartado de solución con recomendaciones para mitigarla.

Toda esta recogida de información y detección de vulnerabilidades se lleva a cabo en la fase de escaneo de una prueba de penetración, que se puede desglosar en las siguientes acciones [2]:

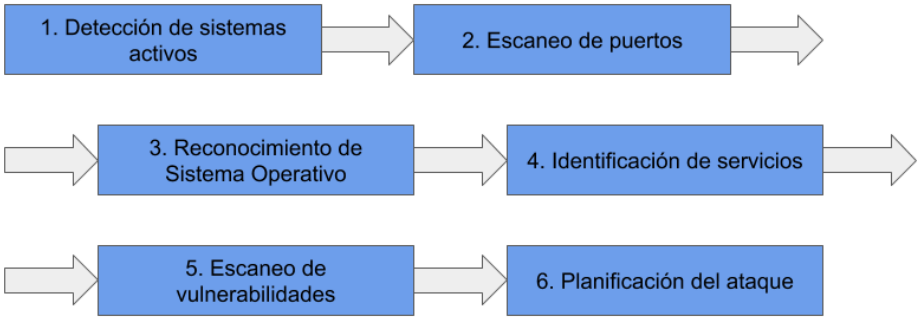



Ilustración 3. Diagrama de flujo. Fase de escaneo en un test de penetración.

La primera acción no es necesaria en este caso que se escoge previamente una máquina víctima concreta activada y controlada por el analista. Las acciones de escaneo son realizadas automáticamente por OpenVAS. Tras el escaneo, OpenVAS muestra los resultados teniendo en cuenta puertos, servicios, vulnerabilidades y riesgos. La planificación del ataque en este caso depende en gran parte de los exploits disponibles en Metasploit, por ser el caso de estudio.

Los pasos seguidos para el escaneo con la herramienta OpenVAS son los siguientes:

1. Activación del servicio
2. Especificación de la IP de la máquina víctima contra la que lanzar el escaneo
3. Lanzamiento del escaneo
4. Visualización de los resultados
5. Exportación del informe de resultados en formato pdf

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:17:96:1c
          inet addr:192.168.154.88  Bcast:192.168.154.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe17:961c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2954 (2.8 KB)
          Base address:0x2000 Memory:ef5c0000-ef5e0000
```

Ilustración 4. Metasploitable2 - IP de la máquina víctima.

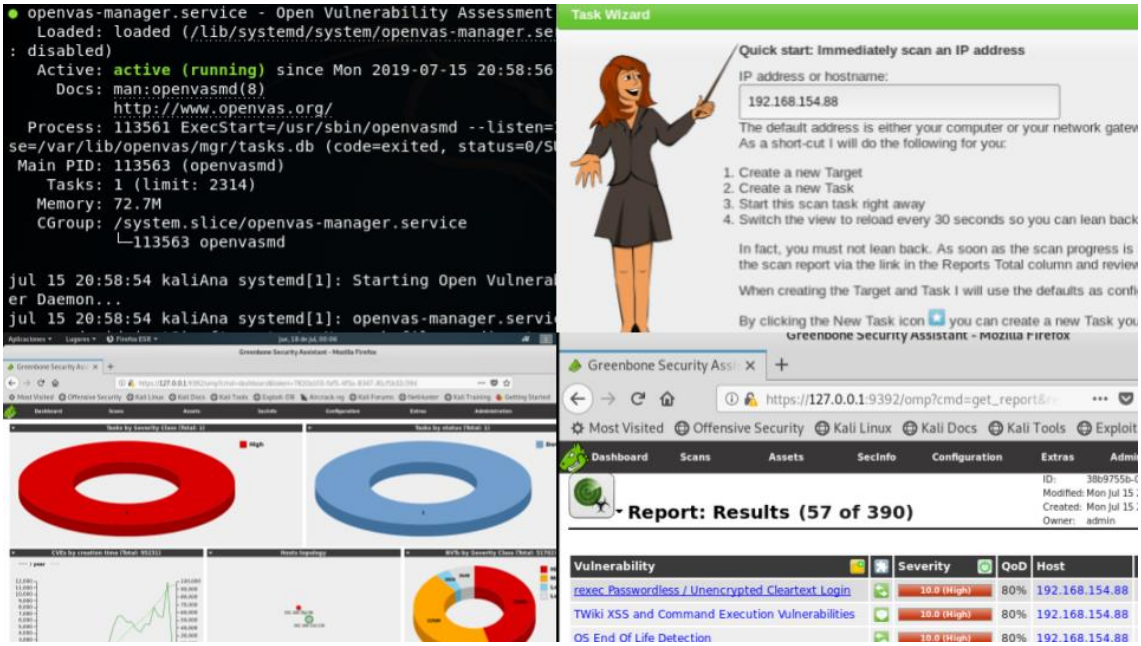


Ilustración 5. Escaneo con OpenVAS. Interfaz web.

Necesitamos escanear la máquina víctima con OpenVAS para encontrar las vulnerabilidades que tiene asociadas y estudiarlas. Una ventaja que tiene el resultado de este análisis y escaneo con OpenVAS es que el informe que se puede obtener de ella viene organizado por puertos, lo que facilita nuestra búsqueda (vamos a centrarnos en ciertos puertos de estudio).

### 1.3.5. Elaboración de una tabla para la relación de puertos, vulnerabilidades y exploits

Como se observa en la Ilustración 5, OpenVAS ha obtenido 390 resultados tras el escaneo a la máquina víctima Metasploitable2. Estos resultados se muestran con un nivel de severidad determinado según el riesgo de cada vulnerabilidad.

Antes de proceder al estudio de las vulnerabilidades encontradas con OpenVAS es necesario hacer una selección de las más importantes, explotables e interesantes para el aprendizaje. Para ello, para el estudio primero se escogió un listado de algunas de las más comunes empleadas en diversos cursos de seguridad informática y muy documentadas, elaborando como resultado una tabla de vulnerabilidades por puerto (Tabla 1, Tabla 2).

En dicha tabla se ha proporcionado un indicador (VSP + Puerto) a cada vulnerabilidad usada para este estudio por comodidad y legibilidad, dado que cada vulnerabilidad engloba un servicio y características específicas, y el puerto en el que está operativo el servicio. Así, por ejemplo, la vulnerabilidad sobre el servicio vsftpd que se encuentra en el puerto 21 recibe el identificador VSP21 (Vulnerabilidad del Servicio en el Puerto 21) en este trabajo.

Como es de esperar, pese a la documentación existente, toda ella se centra en la metodología de explotación, los payloads y exploits concretos que se usan para atacar las

vulnerabilidades [Anexo A.A. ], y serían de difícil comprensión según qué estudiantes deben asimilar esta materia. Por lo tanto, la documentación nos sirve para ejecutar los ataques, es un buen punto de partida, pero no para esquematizarlos, lo que se elabora en este trabajo.

Asimismo, como comentamos en el apartado anterior, la clasificación de las vulnerabilidades viene organizada por puertos al exportar el informe de OpenVAS. Por esta razón, es sencillo buscar en el informe las vulnerabilidades para cada puerto de estudio de los seleccionados.

Algunos de los puertos escogidos no aparecen en el informe. Esto es debido a que OpenVAS no detectó ninguna vulnerabilidad para los servicios que se encuentran en ellos. Por otro lado, tampoco todas las vulnerabilidades que sí aparecen en el informe tienen exploit disponible en Metasploit.

El hecho de que haya vulnerabilidades para las que no existe exploit en Metasploit nos ofrece la posibilidad de representar otros tipos de ataque. Es decir, se pueden aprovechar las vulnerabilidades sin exploit asociado para recoger algunos ataques posibles mediante distintas herramientas y metodologías. Esto añade flexibilidad al modelo de diagrama final.

De cada puerto de estudio se ha escogido una vulnerabilidad. Para aquellos puertos con más de una vulnerabilidad detectada se han tenido en cuenta algunos criterios de selección.

Uno de los criterios de selección de vulnerabilidades es el grado de severidad. Aquellas vulnerabilidades que suponen mayor riesgo tienen mayor prioridad. Suelen ser más interesantes las vulnerabilidades de riesgo alto que las que presentan un riesgo bajo, debido a que en ocasiones las de riesgo bajo tienen pocas probabilidades de ser atacadas con éxito.

Otro criterio de selección es la presencia de CVE (*Common Vulnerabilities and Exposures*) relacionado, que es un identificador de vulnerabilidad asignado por las CNAs[7], debido a que las vulnerabilidades con este tipo de identificador resultan más fáciles de buscar, y, por lo tanto, de estudiar.

La presencia de exploit disponible en Metasploit es uno de los criterios de selección más relevantes tenidos en cuenta, dado que el caso de estudio es esta herramienta y permite probar el ataque escenificado en los diagramas con el entorno de pruebas configurado.

Rapid7 y CVEdetails[8] son las principales fuentes de información sobre vulnerabilidades y exploits consultadas. CVE details es una de las mayores bases de datos de vulnerabilidades en la web. Rapid7 es la compañía propietaria de Metasploit.

Una vez estudiadas y descritas las vulnerabilidades escogidas y sus exploits y una vez recogida toda la información relevante, es posible la identificación de los elementos necesarios para elaborar un primer modelo de creación de diagramas de vulnerabilidades.

[7] https://cve.mitre.org/cve/cna.html
[8] https://www.cvedetails.com/

Tabla 1. Tabla de Vulnerabilidades de Estudio

| Identificador | Puerto | Nombre | Riesgo | CVSS | CVE | Detalles | Exploits |
|---|---|---|---|---|---|---|---|
| VSP21 | 21 | FTP - control (vsftpd) | Alto | 7.5 | CVE-2011-0762 | vsftpd Compromised Source Packages Backdoor Vulnerability | Exploit: exploit/unix/ftp/vsftpd_234_backdoor Payload: cmd/unix/interact |
| | | | Medio | 6.4 | CVE-1999-0497 | Anonymous FTP Login Reporting | Auxiliary: auxiliary/scanner/ftp/anonymous |
| | | | Medio | 5.1 | CVE-2009-5029 | vsftpd '__tzfile_read()' Function Heap Based Buffer Overflow Vulnerability | |
| VSP22 | 22 | SSH | Alto | 7.8 | CVE-2016-6515, CVE-2016-6210 | OpenSSH Denial of Service And User Enumeration Vulnerabilities (Linux) | Auxiliary: auxiliary/scanner/ssh/ssh_enumusers |
| | | | Alto | 7.5 | CVE-2014-1692 | OpenSSH 'schnorr.c' Remote Memory Corruption Vulnerability | |
| | | | Medio | 5.0 | CVE-2018-15473 | Open SSH User Enumeration Vulnerability-Aug18(Linux) | Auxiliary: auxiliary/scanner/ssh/ssh_enumusers |
| VSP23 | 23 | telnet | Medio | 4.8 | | Telnet Unencrypted Cleartext Login | |
| VSP25 | 25 | smtp | Medio | 5.0 | | Check if Mailserver answer to VRFY and EXPN requests | |
| | | | Medio | | CVE-2011-0411 | Multiple Vendors STARTTLS Implementation Plaintext Arbitrary Command Injection Vulnerability | |
| | | | Medio | | CVE-2015-4000 | SSL/TLS: 'DHE_EXPORT' Man in the Middle Security Bypass Vulnerability (LogJam) | MitM |
| VSP53 | 53 | DNS (bind) | Alto | 10.0 | | BIND End of Life Detection | |
| | | | Alto | 7.8 | CVE-2016-2776 | ISC BIND 'buffer.c' Assertion Failure Denial of Service Vulnerability (Linux) | auxiliary/dos/dns/namedown auxiliary/dos/dns/bind_tsig |

Tabla 2. Tabla de Vulnerabilidades de Estudio. Segunda parte.

| Identificador | Puerto | Nombre | Riesgo | CVSS | CVE | Detalles | Exploits |
|---|---|---|---|---|---|---|---|
| VSP80 | 80 | http (Apache) | Alto | 7.1 | CVE-2009-1891 | Apache 'mod_deflate' Denial Of Service Vulnerability - July0 | |
| | | | Alto | 7.5 | CVE-2009-3095 | Apache 'mod_proxy_ftp' Module Command Injection Vulnerability (Linux) | |
| | | | Medio | 5.0 | CVE-2017-9798 | Apache HTTP Server OPTIONS Memory Leak Vulnerability (Optionsbleed) | auxiliary/scanner/http/apache_optionsbleed |
| | 111 | sunrpc | | | | Log | |
| | 139 | SMB/CIFS | | | | Log | |
| VSP514 | 514 | RSH | Alta | 7.5 | CVE-1999-0651 | rsh Unencrypted Cleartext Login | auxiliary/scanner/rservices/rsh_login |
| VSP2121 | 2121 | ProFTPD | Alta | 10 | CVE-2010-4221 | ProFTPD Multiple Remote Vulnerabilities | exploit/freebsd/ftp/proftp_telnet_iac<br>exploit/linux/ftp/proftp_telnet_iac |
| VSP3306 | 3306 | MySQL | Alta | 7.5 | CVE-2009-4484 | MySQL 5.0.51a Unspecified Remote Code Execution Vulnerability | exploit/linux/mysql/mysql_yassl_getname |
| VSP5432 | 5432 | PostgreSQL | Alto | 9.0 | No encontrado | PostgreSQL weak password | auxiliary/scanner/postgres/postgres_login |
| | | | Medio | 5.5 | CVE-2010-1975 | PostgreSQL 'RESET ALL' Unathorized Access Vulnerability | No encontrado |
| VSP5900 | 5900 | VNC (RFB) | Alto | 9.0 | No encontrado | VNC Brute Force Login | use auxiliary/scanner/vnc/vnc_login |
| | | | Medio | 4.8 | No encontrado | VNC Server Unencrypted Data Transmission | Sniffer |
| VSP6200 | 6200 | FTP | Alto | 7.5 | CVE-2011-0762 | vsftpd Compromised Source Packages Backdoor Vulnerability | Exploit: exploit/unix/ftp/vsftpd_234_backdoor<br>Payload: cmd/unix/interact |
| | 6667 | IRC | - | - | - | Log | - |
| | 8009 | | - | - | - | Log | |

## 1.4. Estructura del cuerpo de la memoria

El cuerpo de la memoria se encuentra estructurado en los siguientes apartados: Estudio del Arte, Modelo propuesto, Modelo final aplicado a las vulnerabilidades, Discusión de resultados y por último Conclusiones y líneas futuras.

Tras finalizar esta sección de introducción, se encuentra el Estudio del Arte. Este contiene la recopilación de trabajos y herramientas relevantes. De este modo se sientan las bases que marcan el inicio del trabajo, puesto que una vez se tiene el contexto y se recopilan otros estudios, se puede tomar decisiones tales como las herramientas a utilizar y los temas a tratar, sin caer en estudios repetidos o de gran similitud. Esto va enlazado con el primero de los objetivos expuestos en el apartado de Objetivos (**O1**), realizando un estudio de las diferentes alternativas a Metasploit y otras herramientas utilizadas en este trabajo.

Por su parte, la sección de Modelo propuesto abarca los objetivos restantes (**O2, O3 y O4**). Se lleva a cabo el análisis de las vulnerabilidades, los ataques y sus exploits, así como la identificación de los elementos involucrados en ellos, la clasificación de dichos elementos y por último la elaboración de un diseño representativo reutilizable para todos ellos.

En relación con lo anterior encontramos la sección Modelo final aplicado a las vulnerabilidades. En ella se prueba el modelo elaborado en la sección anterior con todos los puertos de estudio y se explica el uso del modelo para cada uno de ellos. Es decir, se ofrece una visión del uso del modelo propuesto en casos de estudio reales.

Se ha añadido un apartado Discusión de resultados, en el que se argumentan las ventajas y resultados que se obtienen tras haber probado el modelo.

Finalmente, la sección de Conclusiones y líneas futuras. En ella están recogidos los puntos clave del trabajo, tales como los objetivos que se cumplen con el modelo presentado. Además de las conclusiones, en esta sección se encuentran algunas posibles líneas futuras que abarcar a partir de este estudio, de tal forma que quien desee continuar con la línea de este trabajo puede encontrar propuestas.

# 2. Estudio del Arte

A pesar de que este trabajo entra en el área de la ciberseguridad, esta es muy extensa. En este caso interesan principalmente los estudios sobre *pentesting*, análisis de vulnerabilidades, la configuración de un entorno de pruebas y el uso de Metasploit para hacer pentesting y el modelado de vulnerabilidades y ataques de manera visual.

## 2.1. Análisis de vulnerabilidades vs Pentesting

Una prueba o test de penetración (comúnmente llamado *pentesting*) es una metodología empleada en seguridad informática con el objetivo de encontrar vulnerabilidades en un sistema y erradicarlas. Estas pruebas se realizan en entornos controlados, lo que quiere decir que el resultado no compromete el sistema a agentes externos, sino que se compromete de forma interna y controlada para poder estudiar cómo paliar las debilidades y los puntos de acceso no deseados que hayan sido identificados.

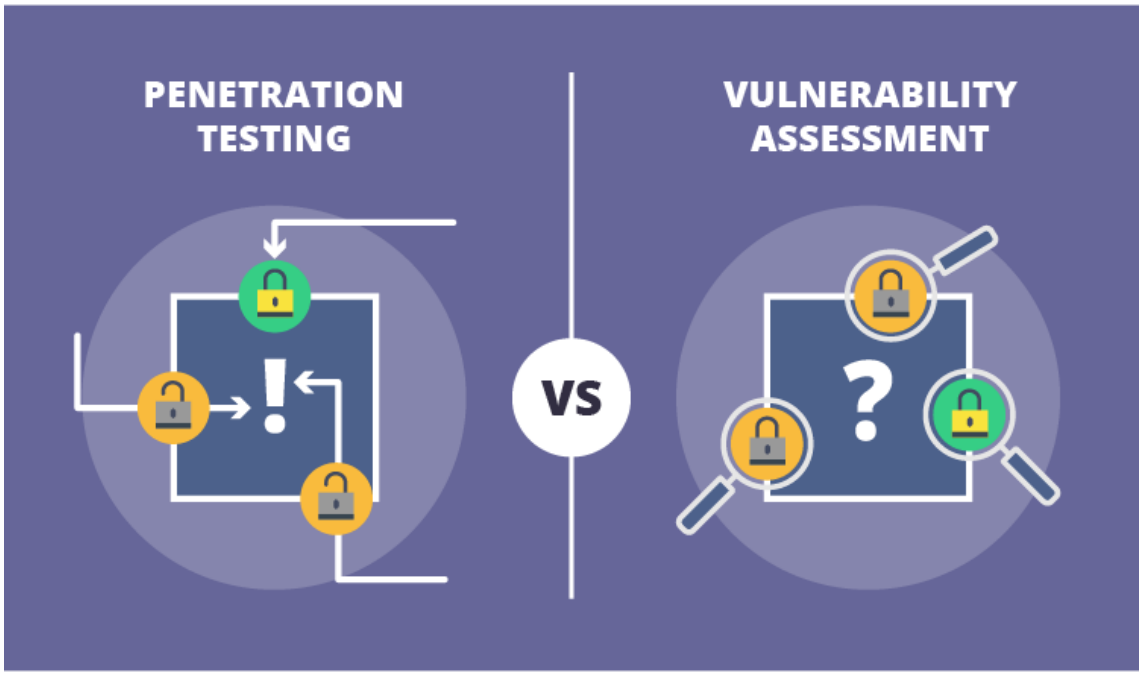



Ilustración 6. Pentesting vs Análisis de Vulnerabilidades

En la Ilustración 6 [3] se compara un test de penetración y una evaluación de vulnerabilidades. El test de penetración incurre en el sistema y pone a prueba sus vulnerabilidades, mientras que la evaluación se limita a identificarlas, analizarlas y reportarlas.

Generalmente, las pruebas de penetración constan de las siguientes fases: recogida de información y estudio de la víctima u objetivo, escaneo de vulnerabilidades, ataque o acceso y persistencia.

Para realizar pruebas de penetración, hoy en día existen numerosas herramientas. Metasploit es una de las más conocidas y utilizadas actualmente. Esta herramienta es muy versátil y completa. En este caso en concreto, la herramienta ha supuesto una gran fuente de información sobre vulnerabilidades y exploits y ha permitido probar ataques en el entorno de pruebas que ha sido configurado previamente usando las máquinas virtuales Kali Linux y Metasploitable2 (Apéndice A.C.).

## 2.2. Entornos de prueba

Para configurar un entorno de estudio seguro y controlado es frecuente usar máquinas virtuales. Concretamente las máquinas Kali Linux y Metasploitable2, las escogidas para este trabajo, son muy comunes para pruebas de seguridad.

La distribución de Kali Linux para laboratorios de seguridad fue la escogida para realizar las pruebas durante este trabajo, al tratarse de software libre con herramientas de seguridad instaladas por defecto [4]. Existen alternativas de distribuciones para pruebas de penetración, como por ejemplo BackBox o Parrot OS [5]. BackBox está basada en Ubuntu, posee las herramientas más habituales de hacking y ofrece una experiencia rápida y efectiva. Parrot OS, aunque más reciente, es parecida y una buena alternativa a Kali y también está basada en Debian. Sin embargo, el hecho de contar con más de 600 herramientas para principiantes, su liberación continua (actualizaciones), su fama (cuanta más gente usa una herramienta, más información y errores habrá reportados) y su comunidad, hace a Kali Linux la finalmente escogida para este trabajo.

En cuanto a Metasploitable2, es una máquina virtual cubierta de fallos de seguridad con la que pueden trabajar tanto expertos como iniciados en la materia [6]. Existen otras alternativas como la distribución Damn Vulnerable Linux (DVL) o Kioptrix, con varios niveles y retos [7].

Asimismo, el framework Metasploit es el framework de pentesting más usado del mundo y ha tomado el papel principal en múltiples estudios sobre pruebas de penetración [8,9].

Por otro lado, a pesar de haber sido objeto de estudio y un tema que ha suscitado gran interés en los últimos años, el nivel de abstracción al usar estas herramientas sigue siendo considerable. Estas herramientas sin duda ayudan a configurar un entorno de pruebas deseado y fácilmente llevar a cabo las fases de una prueba de penetración, desde la recogida de información hasta la elaboración de un informe final. Es tan sencillo que no se necesita conocer la mayor parte de factores involucrados. Basta con utilizar herramientas de análisis de vulnerabilidades que determinen cuáles son los servicios vulnerables en la máquina objetivo y a continuación buscar exploits que afecten a dichas vulnerabilidades. No es intuitivo determinar los procesos que siguen por detrás dichas herramientas sin un diagrama que los trace.

Como herramienta gráfica relacionada con Metasploit cabe destacar Armitage.

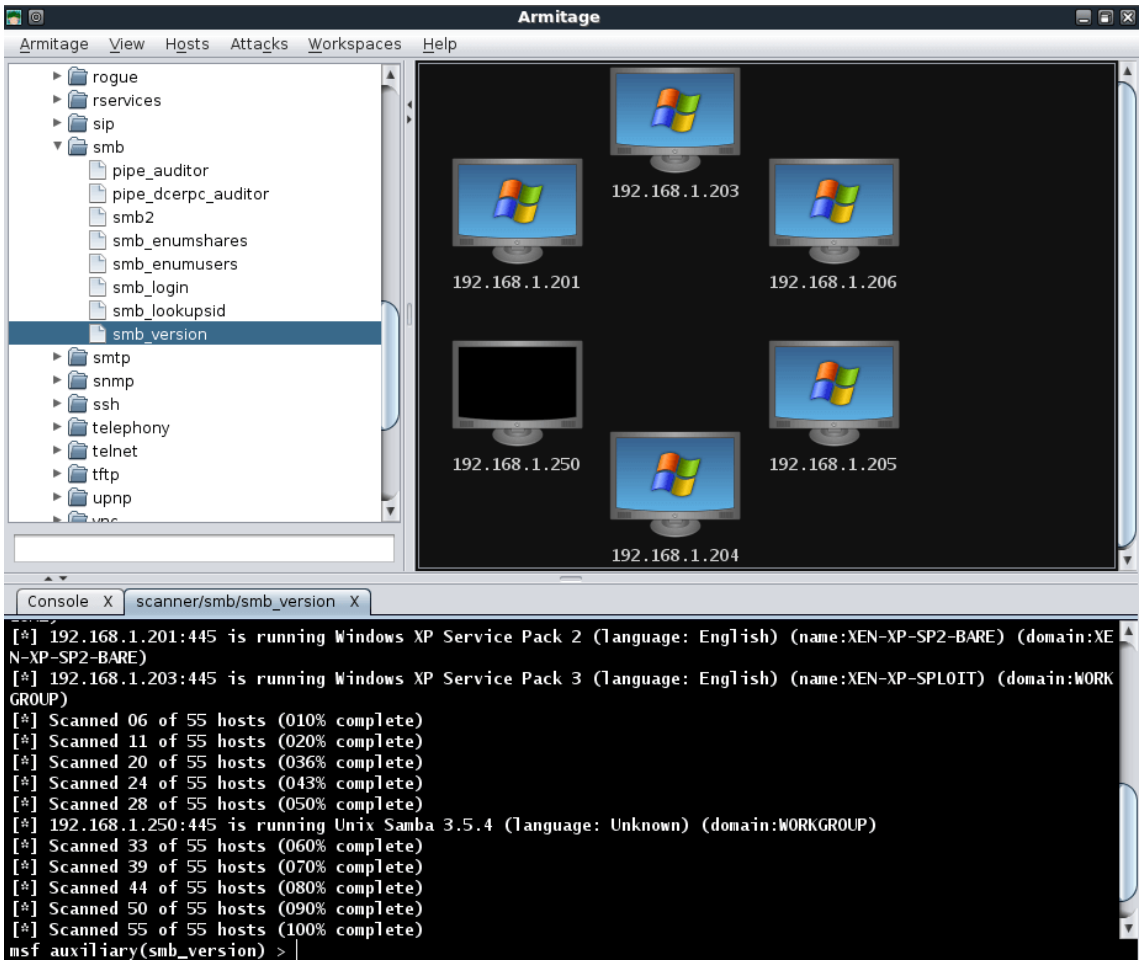


Ilustración 7. Armitage

Como se aprecia en la Ilustración 7[9], Armitage Es una interfaz gráfica de usuario con la que utilizar las funcionalidades de Metasploit. Con ella no es necesario iniciar los ataques por línea de comandos. Sin embargo, tampoco muestra gráficamente el proceso de ataque ni las vulnerabilidades afectadas.

## 2.3. Modelado gráfico de vulnerabilidades

En esta línea de representación hay menos estudios y de momento ninguno se ajusta a un modelado que permita entender los procesos desde un nivel de conocimiento básico en esta área. Por ejemplo, el modelado de grafos de ataque por medio de la inteligencia artificial [10], que puede ocasionar extensos grafos difíciles de interpretar compuestos por decenas de nodos.

En lo que a modelados gráficos se refiere, podemos hablar sobre el método CORAS [11]. Este se centra en la medición de riesgos mediante el modelado de escenarios de amenazas. Su herramienta permite analizar los riesgos, pero no entender de dónde vienen las

[9] Recuperado de https://www.offensive-security.com/metasploit-unleashed/armitage-scanning/

amenazas ni la relación explícita con los métodos de explotación. Tampoco es posible analizar la propagación de vulnerabilidades a los componentes del sistema.

Un modelado que se ajusta más a este trabajo es el dispuesto en una ontología estudiada [12] que ofrece una propuesta de modelado de relaciones y activos que intervienen en el entendimiento de vulnerabilidades y exploits, pero no concluye una escenificación de estos con formas y otros elementos visuales. Se plantea más como base teórico-relacional para la elaboración de una propuesta de diagrama final con plena representación de los elementos.

En cuanto a la herramienta de modelado, existen diversas alternativas a la utilizada para diagramas relacionales, pero fueron descartadas por diversos motivos.

Además de la ya mencionada herramienta de CORAS, draw.io[10] es una herramienta de modelado de amenazas. Esta herramienta sin embargo no cuenta con gran cantidad de formas geométricas y figuras para representar distintos elementos. Ofrece una escenificación más sobria con etiquetados y en su mayoría formas rectangulares. Por estos motivos es Gliffy la herramienta finalmente escogida.

Gliffy[11] es una aplicación disponible en Chrome Web Store que cuenta con una amplia variedad de formas y elementos relacionales que permite crear diagramas fácilmente. Cuenta con formas básicas y representativas y es posible descargar los diagramas en formato jpg y png, aunque no incluye la opción de agregar notas.

Como se ha mencionado anteriormente, en lo referente a las herramientas utilizadas se puede confirmar que son las apropiadas para este tipo de estudios. De igual forma, se confirma la falta de trabajos de esta índole representativa que contribuyen a la formación básica o elemental sobre vulnerabilidades y sus exploits.

[10] diagrams.net, 2019

[11] https://www.gliffy.com/

# 3. Modelo propuesto

En esta sección se desvela el resultado de un modelo de clasificación, representación y aprendizaje como fruto de la investigación sobre vulnerabilidades y ataques seleccionados para varios puertos de estudio. Esta es una de las secciones principales de este TFG, dado que el resto de análisis se basan en el modelo y la metodología para su enriquecimiento. La sección se estructura como sigue.

En primer lugar se propone el modelo para la representación de Vulnerabilidades y Exploits, que se ha denominado REVEX. El diseño propuesto es capaz de integrar elementos comunes a las vulnerabilidades y ataques de estudio. Permite la identificación de las vulnerabilidades y la comprensión de los ataques, de los cuales se distinguen aquellos con exploit disponible en Metasploit. Aparecen detallados los elementos que conforman el modelo y la guía de representación propuesta para llevar a cabo el modelado de vulnerabilidades con sus ataques.

En segundo lugar, la metodología para la creación del modelo. En ella se describen las etapas necesarias para crear el modelo propuesto. De esta forma queda explicado el proceso que se ha seguido para crear el modelo y queda abierto para aquellos que quieran usarlo, ya sea para crear un nuevo modelo o modificar el aquí propuesto.

Por último, la ejecución de la metodología. En este apartado se observa cómo se ha aplicado la metodología de creación del modelo, para lo que se siguen las etapas anteriormente mencionadas.

## 3.1. REVEX - REpresentación de Vulnerabilidades y EXploits

El modelo REVEX definido en este TFG consta de una plantilla de elementos y unas reglas que permiten utilizar dicha plantilla de forma adecuada.

La plantilla de elementos ha sido creada con la ya mencionada herramienta de desarrollo de diagramas “Gliffy”. En esta plantilla de elementos se muestran las diferentes formas que dan representación a los elementos esenciales de vulnerabilidades y ataques de estudio. Cada figura está ligada a un elemento diferente, procurando mantener toda la simplicidad y distintividad posible.

Por otro lado, es importante definir una serie de reglas que indican cómo organizar los elementos que conforman la plantilla y cómo colocarlos de forma que se relacionen entre sí y se dispongan manteniendo la concordancia, la simetría y el sentido en cada uno de los diagramas realizados a partir de ella.

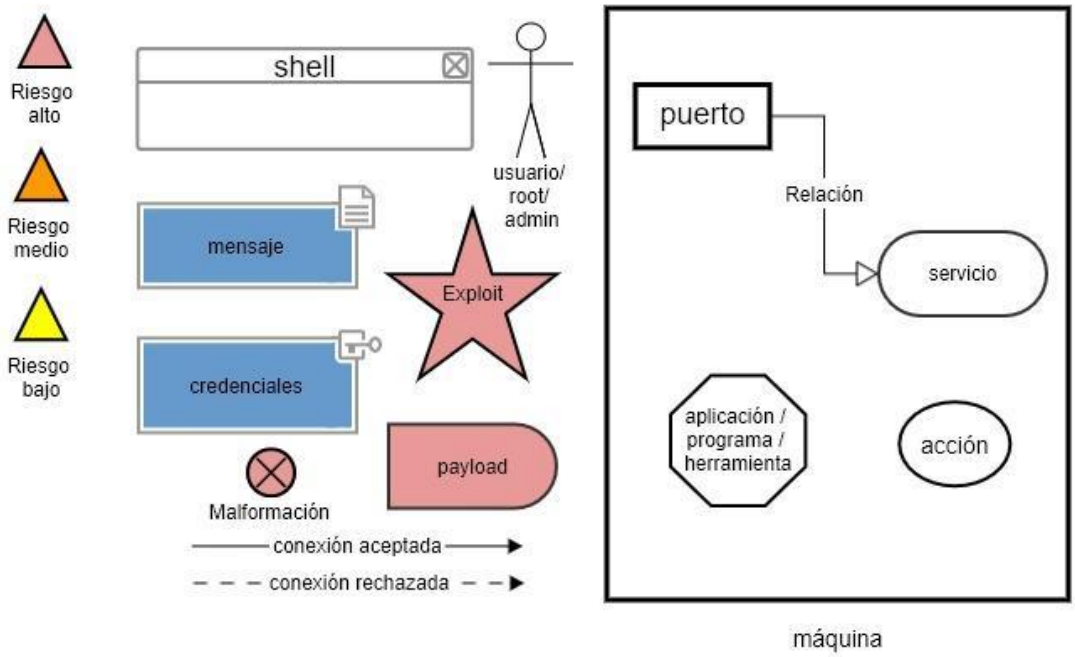



Ilustración 8. Modelo final propuesto

Lo primero a destacar en la Ilustración 8 es la disposición de los elementos en tres agrupaciones principales (de izquierda a derecha): Tipo de riesgo, componentes y máquina.

Las figuras triangulares a la izquierda advierten el tipo de riesgo que supone la vulnerabilidad expuesta. En rojo esta figura aparecerá para vulnerabilidades de riesgo alto, en naranja para las de riesgo medio y finalmente amarillo para las de riesgo bajo.

En general, los componentes y figuras que representan a los elementos pueden mostrarse en cualquier zona del diagrama, pero tal y como podemos ver en la Ilustración 8. Modelo final propuesto, hay algunos elementos que van a aparecer siempre dentro de las máquinas, como por ejemplo los puertos y servicios y la relación entre ellos.

Los elementos representados en el modelo final son los siguientes: tipo de riesgo que entraña la vulnerabilidad representada, shell o intérprete de comandos, tipo de usuario (root, admin, etc.), mensaje (puede referirse a una petición, texto en claro, etc.), credenciales (nombres de usuario y/o contraseñas), exploit disponible en Metasploit, payload o carga útil maliciosa, malformación, conexión aceptada, conexión rechazada, puerto lógico, servicio que se encuentra albergado en el puerto, relación entre elementos, aplicación/programa/herramienta, acción y máquina.

En cuanto a las reglas, deben servir a modo de guía para elaborar los diagramas en base al modelo planteado, por lo que se han aplicado y son comunes a todos los esquemas finales de vulnerabilidades y ataques realizados en este trabajo, dispuestos en el apartado Modelo final aplicado a las vulnerabilidades. A continuación, se exponen dichas reglas.

En primer lugar, el color de la máquina debe reflejar si se trata de la máquina atacante escogida para las pruebas de penetración o no. Para ello, la máquina atacante es representada en color negro, mientras que el resto de las máquinas (inofensivas) son representadas en color azul.

Por otro lado, cabe destacar que las interacciones y relaciones de elementos se dispondrán en orden de escritura (de izquierda a derecha y de arriba hacia abajo) siempre que sea posible. Esto permite seguir el orden del flujo de interacciones.

Dependiendo de la configuración de la máquina víctima, en algunas ocasiones se encuentran unos servicios o versiones de servicios diferentes a los que se encuentran en otras máquinas en el mismo puerto. Además, no todas las versiones de un servicio tienen por qué estar afectadas por una vulnerabilidad. Por este motivo, se debe especificar para cada puerto la versión o versiones del servicio que están ligadas a la vulnerabilidad y al ataque escenificados.

La representación de las acciones y de las aplicaciones viene delimitada dentro de una máquina. En el caso de las acciones, se encuentran dentro de la máquina en la que se realizan. En el caso de las aplicaciones, se encuentran dentro de la máquina que las posee.

Ciertos ataques tienen como resultado la obtención de una shell de comandos con privilegios root. Es interesante que esta información quede reflejada en el diagrama. De esta manera se tiene constancia del tipo de usuario al que se tiene acceso mediante la shell. Hay que tener en cuenta que, en ocasiones, el tipo de usuario al que se tendrá acceso dependerá del usuario que la víctima haya configurado para el servicio vulnerable. Por ello no siempre será fijo o conocido y, en tal caso, no será incluido.

Los elementos comprometedores o comprometidos son representados en color rojo, de modo que la única diferencia entre un elemento y ese mismo elemento estando comprometido o siendo comprometedor será el color (blanco o rojo).

Siguiendo la línea de los colores, los riesgos de las vulnerabilidades se representan conforme a los colores y las formas indicadas en el modelo final (colores rojo, naranja y amarillo y forma triangular) y serán colocados en la esquina superior izquierda del diagrama creado.

Por último, la representación de exploits disponibles en Metasploit viene dispuesta sobre el elemento o elementos final/es que explota/n la vulnerabilidad. Esto quiere decir que un exploit no aparecerá en un elemento comprometido, sino en un elemento comprometedor.

## 3.2. Metodología para la creación del modelo

A lo largo de la ejecución del trabajo se ha podido comprobar que el modelo no puede ser estático, sino que necesita evolucionar, por ejemplo, para incluir nuevos componentes que sean necesarios para perfilar futuros ataques. Es por ello que se hizo necesario definir o seguir una metodología para poder hacer el modelo evolutivo.

La metodología empleada para la construcción del modelo es la de prototipado. Esta metodología se basa en la elaboración de prototipos que secuencialmente se prueban y mejoran hasta llegar a una versión final [13].

La metodología, aplicada a este caso en concreto, resulta como sigue. El primer prototipo nace de la identificación de los agentes involucrados en el estudio de vulnerabilidades y exploits y la relación entre ellos, así como de las formas que los van a representar. Este prototipo es probado para representar con él los primeros casos de estudio. Con la prueba del modelo, para cada vulnerabilidad estudiada se identifican elementos que deben ser incluidos, modificados o eliminados. Tras los cambios necesarios en el modelo, se realiza una nueva versión o prototipo de este y se repite el proceso. De esta forma, el modelo es creado de manera retroactiva.

Con las distintas iteraciones se busca crear un modelo representativo lo suficientemente genérico, es decir, capaz de representar una amplia variedad de vulnerabilidades y ataques distintos.

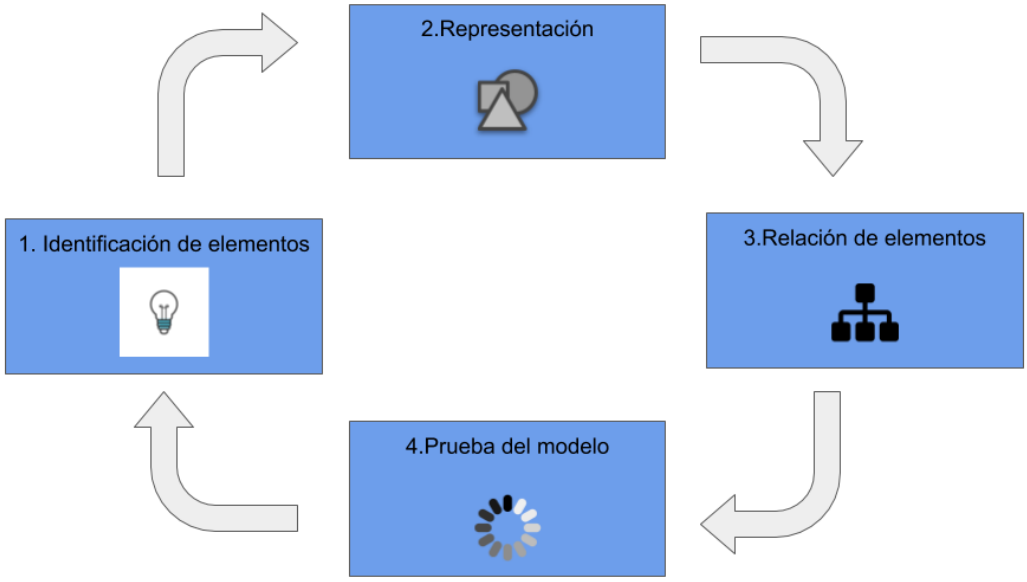



Ilustración 9. Diagrama de flujo. Diseño por prototipado.

La metodología aquí descrita se ha añadido con el objetivo de poder utilizarla en otros contextos y también para otras vulnerabilidades: Como se observa en la ilustración, las distintas fases del diseño permiten que en el futuro se pudiera modificar el modelo propuesto al identificar nuevas necesidades tras probarlo en nuevas vulnerabilidades sujetas a estudio.

Como observamos en Ilustración 9, los pasos seguidos para la creación de cada prototipo hasta llegar al modelo final son los siguientes: Identificación de elementos, Representación de elementos, Relación de elementos y Prueba del modelo. Las siguientes secciones describen de forma más detallada las fases de la metodología, comprendiendo que para cada vulnerabilidad analizada deben hacerse de uno a varios ciclos pasando por las cuatro fases descritas en la Ilustración 9.

### 3.2.1. Identificación de elementos

Antes de identificar los elementos, es importante tener en cuenta que lo primero e indispensable a representar son las máquinas involucradas. En este caso, como mínimo se necesitan dos máquinas: una atacante y otra víctima, por lo que en todos los modelos vamos a encontrar la representación de estas dos máquinas.

Dicho esto, lo segundo a destacar son los servicios. Las vulnerabilidades están directamente relacionadas con los servicios, ya que son estos los que las causan. Para cada puerto escogido hay un servicio activo asociado. Si no hubiera servicio en el puerto o no estuviera activo, no se podría probar el ataque ni encontrar vulnerabilidad en la fase de escaneo. Por ejemplo, en el primer puerto de estudio, puerto 21, mediante OpenVAS se ha detectado una vulnerabilidad causada por el servicio activo vsftpd.

En tercer lugar, cabe destacar el tipo de interacción que lleva a cabo el atacante con el puerto: escucha de paquetes, envío de mensajes, envío de peticiones, intercepción de mensajes, etc. Relacionado con esto último están las posibles herramientas usadas por el atacante, también representadas, que le permiten interactuar de alguna de estas formas con la máquina víctima.

Por otro lado, el nivel de abstracción puede verse reducido por el número de acciones representadas en los diagramas, que pueden añadir información tanto del lado del atacante como de la víctima.

En cuanto a los riesgos que entrañan las vulnerabilidades, se añade un elemento identificativo del tipo de riesgo según su severidad, que se obtiene del análisis de vulnerabilidades de OpenVAS.

Finalmente, lo que serían los elementos más característicos del uso de Metasploit: presencia de exploit y payload y consecuencia del ataque. Esta consecuencia suele ser en la mayoría de los casos la ganancia de acceso remoto a la máquina víctima por medio de una shell de comandos. El tipo de privilegios de acceso vendrá determinado por el tipo de usuario que obtengamos con la shell, que en muchos casos es root.

Hemos visto la variedad de elementos que se pueden identificar, que, como resulta evidente, no se encuentran presentes en todas las vulnerabilidades y ataques. Es por esto que es necesario seguir esta fase de identificación para cada uno de los casos de estudio.

### 3.2.2. Representación de elementos

*"*Las representaciones visuales favorecen el aprendizaje*"* - Antonio M. (2018), *Formación basada en evidencias: representaciones visuales.*

Es esencial primero escoger los elementos a representar acordes al previo estudio de las vulnerabilidades, y segundo escoger adecuadamente los pilares de representación de estos

elementos ya señalados para conseguir identificar las vulnerabilidades y escenarios de ataque de manera visual y educativa.

La representación de los elementos se ha basado en tres pilares: colores, formas y disposición.

Los colores se han seleccionado tomando como referentes tres herramientas muy comunes y por lo tanto conocidas en el área de la seguridad: OpenVAS, Nessus y WireShark. Las dos primeras son escáneres de vulnerabilidades y asignan los colores dependiendo del riesgo que implican las vulnerabilidades. WireShark es un analizador de tráfico de red y asigna los colores para identificar el tipo de información de cada entrada de resultado.

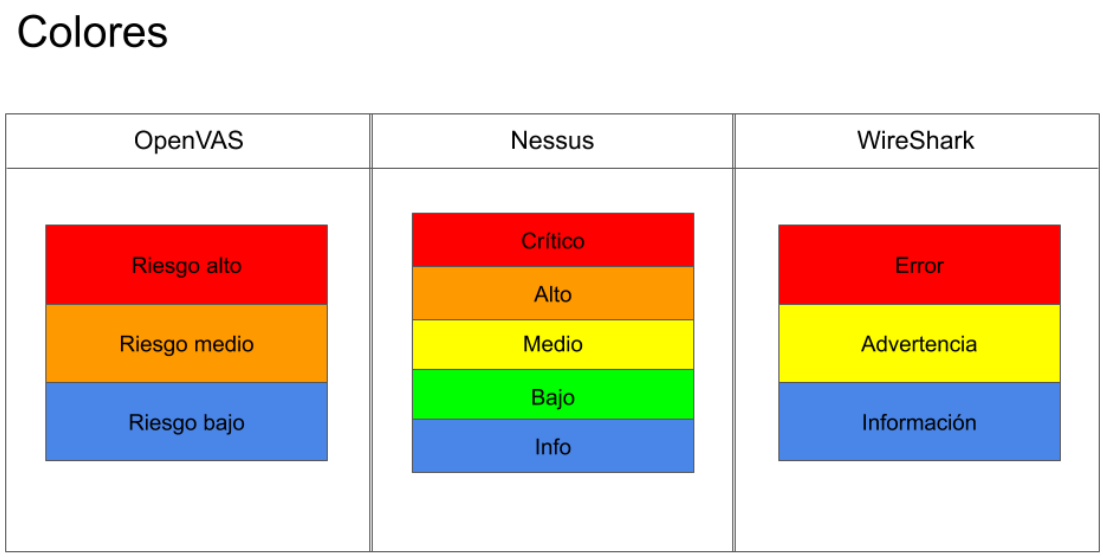



Ilustración 10. Asociación de colores en herramientas de seguridad.

La influencia de los colores que utilizan estas herramientas [14,15] facilita la concordancia entre estas y el modelo. Los analistas o estudiantes que utilicen el modelo y las mencionadas herramientas estarán familiarizados con los colores y su significado. Se concluye que los colores más cálidos son comúnmente utilizados para expresar mayor riesgo o fallo, mientras que los más fríos tienden a expresar información y fiabilidad.

Tras el análisis de las herramientas tomadas como referentes y de su uso de los colores, el modelo sigue su misma línea de asociación de colores.

Como consecuencia, se asocia los colores más cálidos a las vulnerabilidades con mayor riesgo (Ilustración 11).


Ilustración 11. Modelo propuesto. Tipos de riesgo.

A pesar de haber analizado las vulnerabilidades de la máquina víctima Metasploitable2 mediante la herramienta OpenVAS, la asociación de colores escogida para las vulnerabilidades en este caso no es exactamente la misma que la de la herramienta. Esto se debe a que OpenVAS asocia el riesgo bajo al color azul, mientras que este mismo color en este modelo ha sido escogido para representar máquinas no atacantes y elementos que transportan información, como los mensajes. La asociación de colores, por tanto, es más parecida finalmente a la herramienta Nessus.

**FORMAS**

El principal reto que presenta la elección de las formas es la combinación de sencillez y personalización, es decir, la capacidad del modelo para adaptarse al diseño que ofrecen otras herramientas y la capacidad para identificar los elementos inequívocamente con sus formas.

Tabla 3. Tabla asociativa de formas y elementos.

| Forma | Significado asociado | Elemento representado |
|---|---|---|
| Círculo / Óvalo | Movimiento | Acciones |
| Cruz | Rechazo, error | Malformación |
| Rectángulo | Solidez | Puerto |
| Rectángulo redondeado | | Servicios |
| Triángulo | Peligro | Riesgo |
| Octágono | Sinergia, trabajo | Aplicaciones / Programas / Herramientas |
| Estrella | Esteroide luminoso.<br>En algunos países se asocia al totalitarismo y terror estatal. | Exploit |
| Bala | Proyectil | Payload |
| Flecha | Direccionamiento | Relaciones |
| Monigote | Persona | Tipo de usuario |

Todas las formas escogidas son propensas a aparecer en herramientas de diseño y esquematización y son fáciles de crear en dos dimensiones. Como se observa, cada forma tiene una relación con el elemento al que representa [16,17,18].

En cuanto a disposición, se ha configurado una estructura única que ha sido utilizada desde el primero de los prototipos hasta el diseño final:

Direccionamiento según el orden de escritura latino, es decir, de arriba a abajo y de izquierda a derecha, máquinas inofensivas distinguibles mediante un bordeado de color azul, en contraposición al por defecto bordeado de color negro, que se mantiene para identificar la máquina atacante, y por último interconexión puerto-servicio, ya que puertos y servicios han sido representados el uno relacionado con el otro desde el primero de los prototipos de modelo.

### 3.2.3. Relación de elementos

Respecto a la relación de elementos y a propósito del direccionamiento antes mencionado, la forma utilizada por excelencia es la flecha. Las flechas permiten indicar dirección y conexión y son un componente fundamental en los diagramas.

En este caso, se han utilizado tres tipos de flecha según su funcionalidad:

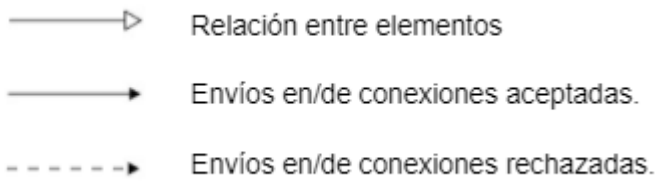



Ilustración 12. Flechas

Como se observa en la Ilustración 12, las relaciones de conexiones son representadas con flechas de punta negra. Para envíos de conexiones aceptadas se utiliza una flecha de línea continua. Para aquellas conexiones que son rechazadas se utiliza la flecha de línea discontinua.

La relación entre elementos es representada con una flecha de punta blanca. Esta es habitual para relacionar elementos dentro de una misma máquina, como por ejemplo la relación entre un puerto y el servicio al que se accede por ese puerto, mientras que las flechas de conexiones suelen aparecer para relacionar elementos entre máquinas (p.ej. conexión desde una herramienta de la máquina atacante al puerto de la máquina víctima).

### 3.2.4. Prueba del modelo

En esta fase se prueba cada prototipo de modelo para las vulnerabilidades de estudio. Esto se consigue realizando diagramas a partir de los elementos y de la estructura del prototipo. En el momento en el que dicho prototipo de modelo no se adapta a alguna de las vulnerabilidades, se retorna a la fase 1 (Apartado 3.2.1) para identificar los elementos faltantes y sobrantes del modelo.

## 3.3. Ejecución de la metodología

Las formas, colores, disposiciones, relaciones, elementos y demás componentes ya mencionados que han conformado el modelo final han sido resultado de iteraciones y transformaciones derivadas de las pruebas del prototipo en los casos de estudio tomados como referencia. En esta sección se describen las iteraciones sobre las vulnerabilidades seleccionadas (Apartado 1.3.5).

Primer prototipo de modelo:

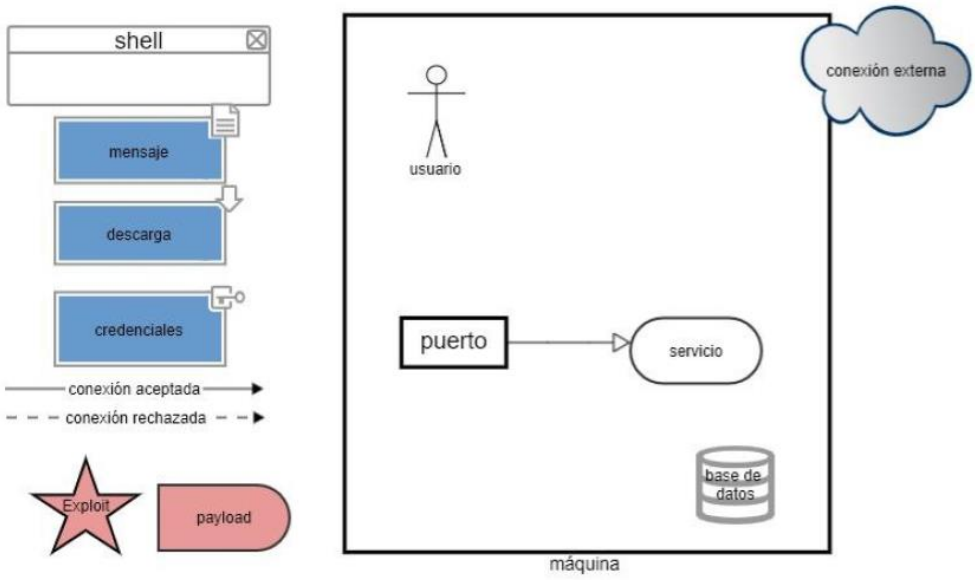



Ilustración 13. Primer prototipo.

Los elementos para este primer prototipo fueron elegidos con anterioridad a las pruebas y tras recopilar información sobre las vulnerabilidades a representar. El modelo se ha visto modificado tras cada prueba para representar las vulnerabilidades. Es por ello que en las posteriores versiones del modelo se incluyeron nuevos elementos y/o se eliminaron o modificaron algunos otros. Podemos observar que este primer prototipo difiere mucho del modelo final.

Con el primer prototipo de modelo no se hizo un diagrama para cada uno de los puertos de estudio. Esto se debe a que bastó con aplicarlo únicamente para las vulnerabilidades en los puertos 21, 22 y 23 para identificar carencias del prototipo que permitieron la elaboración del siguiente. De nuevo, siguiendo la metodología de creación del modelo (Ilustración 9),

con la prueba de este para cada vulnerabilidad se identifican sus carencias y se elabora el siguiente.

A continuación, se muestran los diagramas resultantes tras el análisis de las vulnerabilidades para los servicios en los puertos 21, 22 y 23 a partir del primer prototipo:

### 3.3.1. Iteración sobre VSP21 (CVE-2011-0762)

La vulnerabilidad identificada por OpenVAS sobre el servicio vsftpd que usa el puerto 21 permite un acceso a la shell de comandos mediante una puerta trasera que se abre en el puerto 6200. Esto es posible introduciendo los caracteres especiales ':)' en el nombre de usuario que es solicitado por el servicio vsftpd al conectarse por el puerto 21.

Si el nombre de usuario contiene ':)' entonces se abre una conexión al puerto 6200 desde la víctima con el exterior. Cuando te conectas desde el exterior a través de ese puerto se ejecuta la shell, con la que ganas acceso a comandos en modo root.

Como se ha mencionado, el servicio vulnerable identificado por OpenVAS que se encuentra en el puerto 21 y que permite este acceso es vsftpd, concretamente la versión 2.3.4.

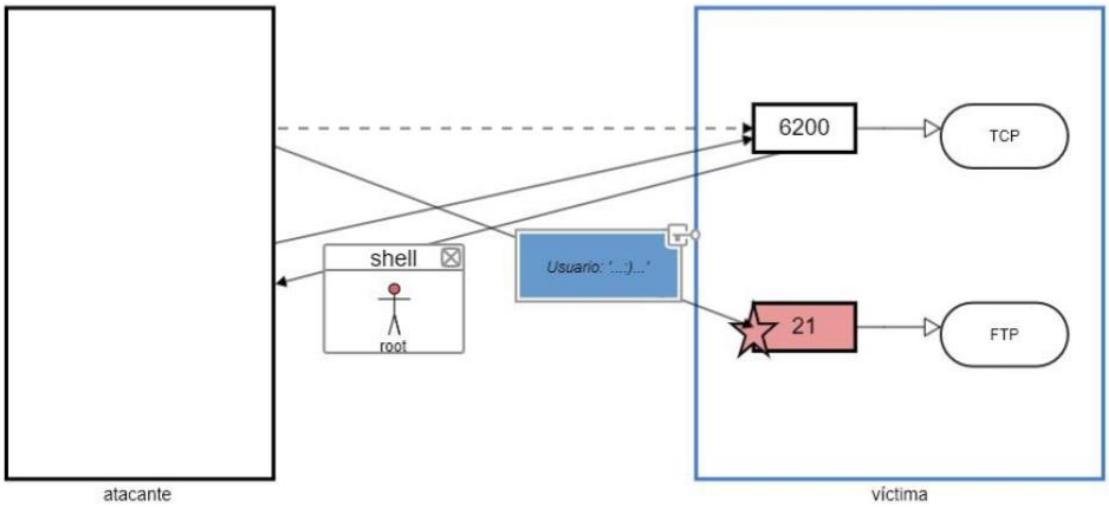



Ilustración 14. Primer prototipo - VSP21.

### 3.3.2. Iteración sobre VSP22 (CVE-2018-15473)

En este caso, en el puerto 22 se encuentra el servicio vulnerable OpenSSH (versiones desde la 2.2 hasta la 7.7). Al intentar autenticar a un usuario con un paquete SSH_MSG_USERAUTH_REQUEST malformado, si el usuario no existe, el servidor no tarda en enviar de vuelta un error SSH2_MSG_USERAUTH_FAILURE. Si el usuario es válido, el servidor tiene entonces que comprobar el paquete y al no ser este válido, cierra la conexión con un error fatal. Por lo tanto, es más rápida la respuesta a un usuario inválido que la de uno válido

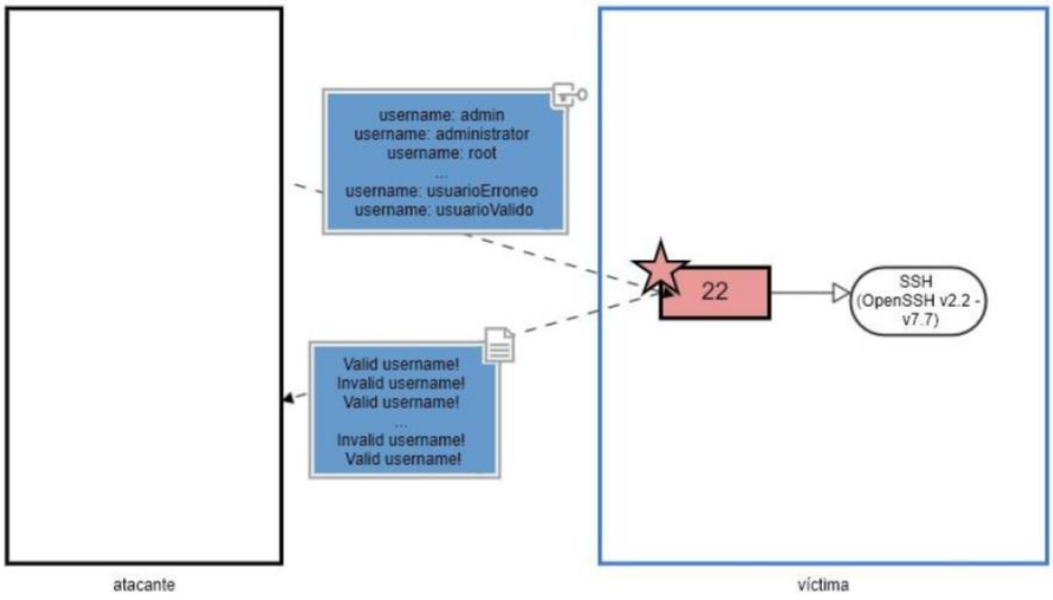



Ilustración 15. Primer prototipo - VSP22

### 3.3.3. Iteración sobre VSP23

La vulnerabilidad identificada por OpenVAS sobre el servicio Telnet que usa el puerto 23 permite obtener datos de terceros. La comunicación via Telnet se transmite en texto en claro. Al no ser encriptada es vulnerable a ser interceptada por otros. Al enviar credenciales de inicio de sesión por telnet, la información queda vulnerable a ser obtenida por el atacante.

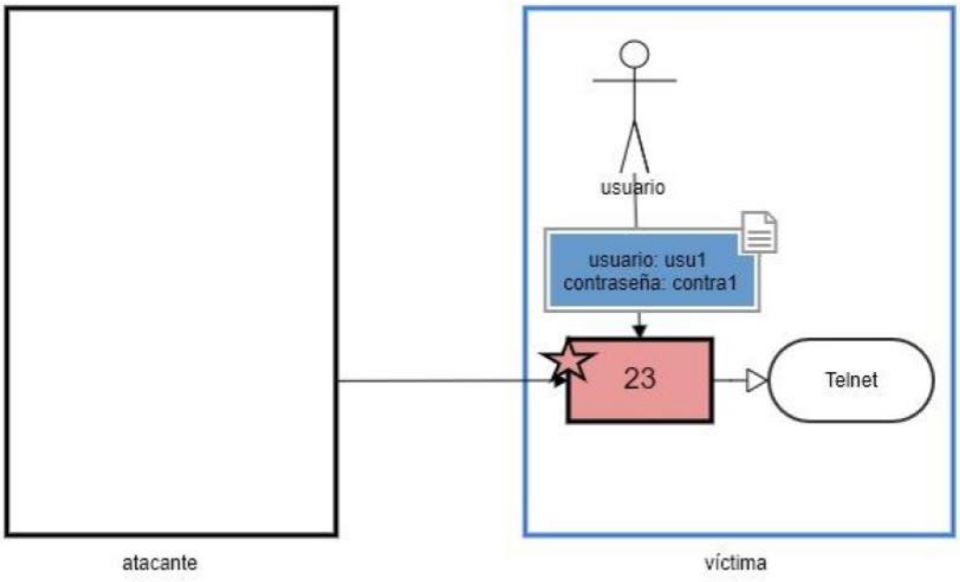



Ilustración 16. Primer prototipo - VSP23

El diseño del primer prototipo fue suficiente para representar las vulnerabilidades de los dos primeros puertos. Sin embargo, para la del puerto 23 fue necesario incluir la forma de especificar acciones y aplicaciones para conferir sentido y contexto a las relaciones.

Estos elementos por tanto se incluyen en el segundo prototipo:

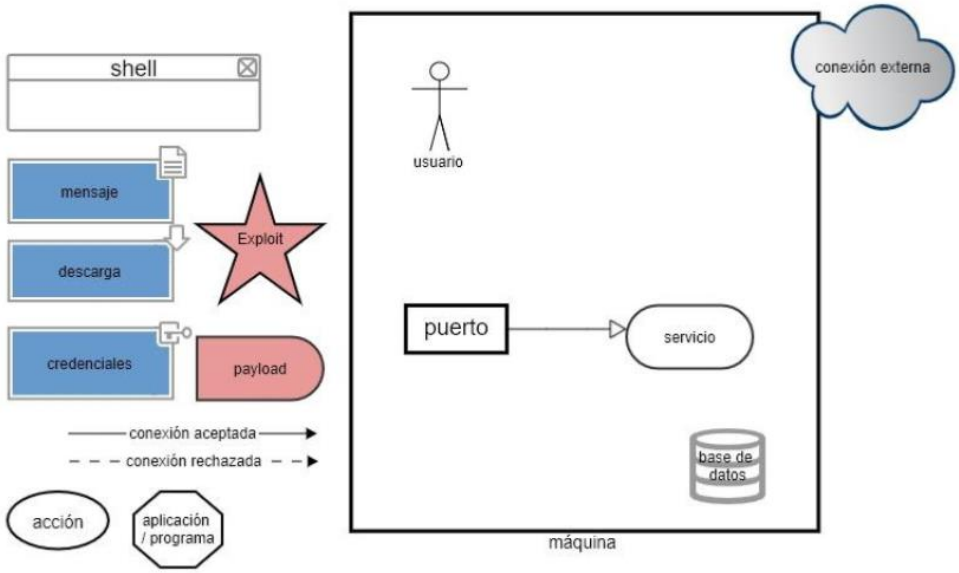



Ilustración 17. Segundo prototipo.

Los diagramas hechos hasta el momento se modificaron teniendo en cuenta estos dos elementos, consiguiendo así mayor nivel de especificidad y refuerzo a la comprensión visual para analizar las relaciones conceptuales [19].

Podemos comprobar cómo queda representada la vulnerabilidad del servicio Telnet (VSP23) mediante la aplicación del segundo prototipo de modelo en la Ilustración 18.

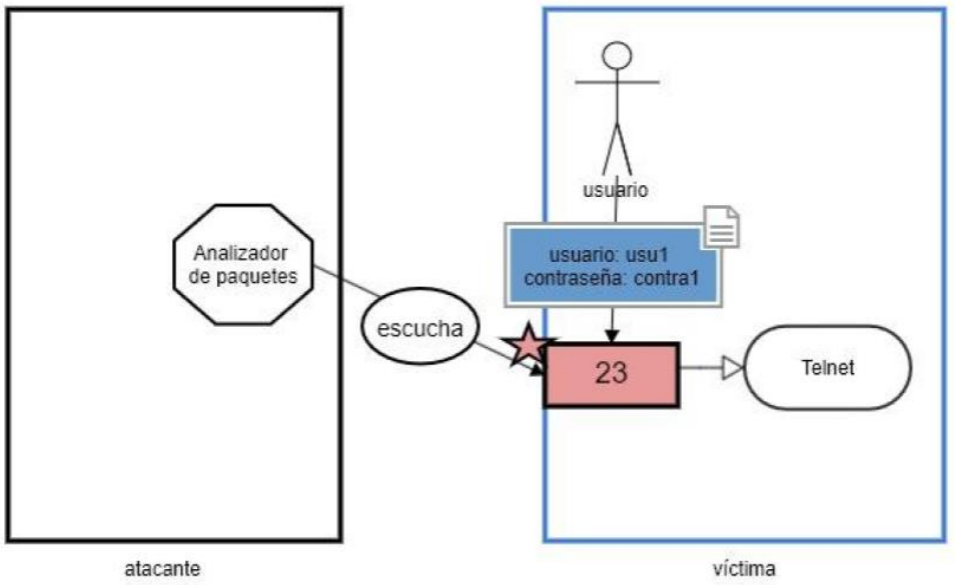



Ilustración 18. Segundo prototipo - VSP23.

Concretamente, debido a la inclusión de las acciones se consigue esquematizar complejos escenarios de ataque como el de la VSP25, en el que se requiere un mayor nivel de detalle.

### 3.3.4. Iteración sobre VSP25 (CVE-2015-4000)

La vulnerabilidad identificada por OpenVAS sobre el servicio Telnet (con versión 1.2 y anteriores de TLS) que usa el puerto 23 permite aceptar conexiones mediante el conjunto de cifrado DHE_EXPORT. Esto posibilita a un atacante realizar un man-in-the-middle y degradar la fortaleza de las claves usadas en el intercambio de claves Diffie-Hellman al forzar el uso del conjunto de cifrado DHE_EXPORT, facilitando el descifrado de las claves. Para ello el atacante debe reenviar al servidor el ClientHello reemplazando el conjunto de cifrado DHE por DHE_EXPORT, que trabaja con claves débiles de 512 bits (las más usadas son incluso ya conocidas) y a continuación debe reenviar al cliente el ServerHello reemplazando DHE_EXPORT por DHE. De esta forma el atacante puede obtener la clave de 512 bits sin que el cliente sepa siquiera que la conexión con el servidor ha sido mediante DHE_EXPORT.

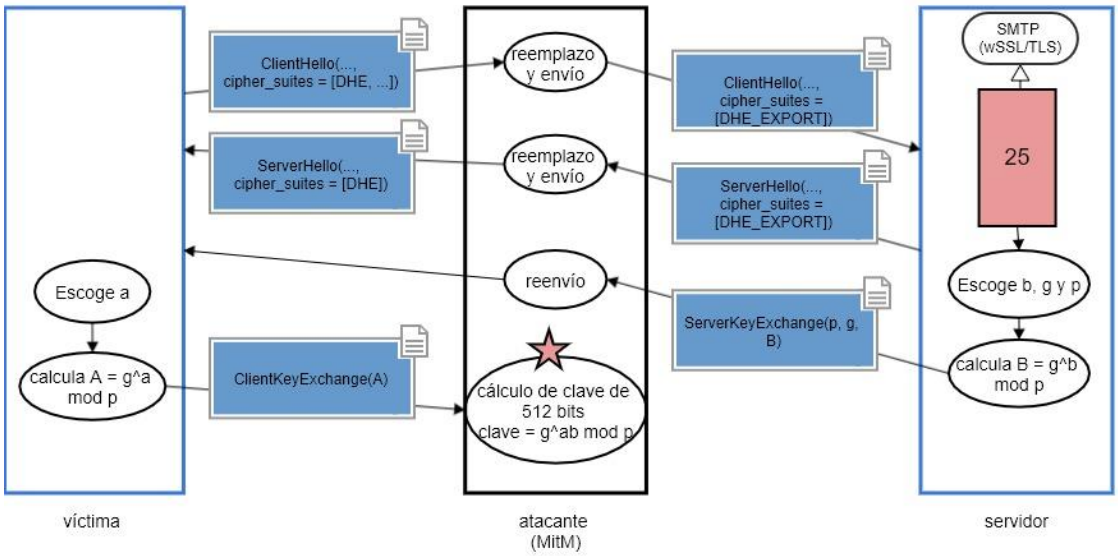



Ilustración 19. Segundo prototipo - VSP25.

Con el segundo prototipo (Ilustración 17) se podría conseguir la representación de prácticamente el 100% de las vulnerabilidades de estudio. Sin embargo, es en el estudio de las vulnerabilidades de los servicios en los puertos 53 y 3306 (VSP53 y VSP3306) cuando se identifica una nueva forma de distinción para los mensajes: las malformaciones, para las que anteriormente no había forma de representación asociada (p.ej. Primer prototipo - VSP22). Asimismo, se observa que tanto las acciones como las herramientas siempre van a ir incluidas dentro del contexto de la máquina.

Por otro lado, algo que también se saca en claro tras el estudio y representación de las vulnerabilidades es el desuso de tres de los elementos mantenidos desde el primer prototipo: la representación específica de una base de datos, la de una conexión externa y la de una descarga. Esto se debe a que las conexiones a terceros se pueden ver representadas como una nueva máquina, tal y como se observa en la VSP25, que las bases de datos se pueden representar como un servicio más, como se observa en la VSP3306, y que las descargas se pueden escenificar con una relación entre la máquina que ofrece la descarga y la que la recibe con una acción que indique descarga en su interior. Hay que

tener en cuenta que el objetivo es crear un modelo que además de ser representativo sea lo más sencillo posible.

Tercer prototipo para el modelo de diagrama:

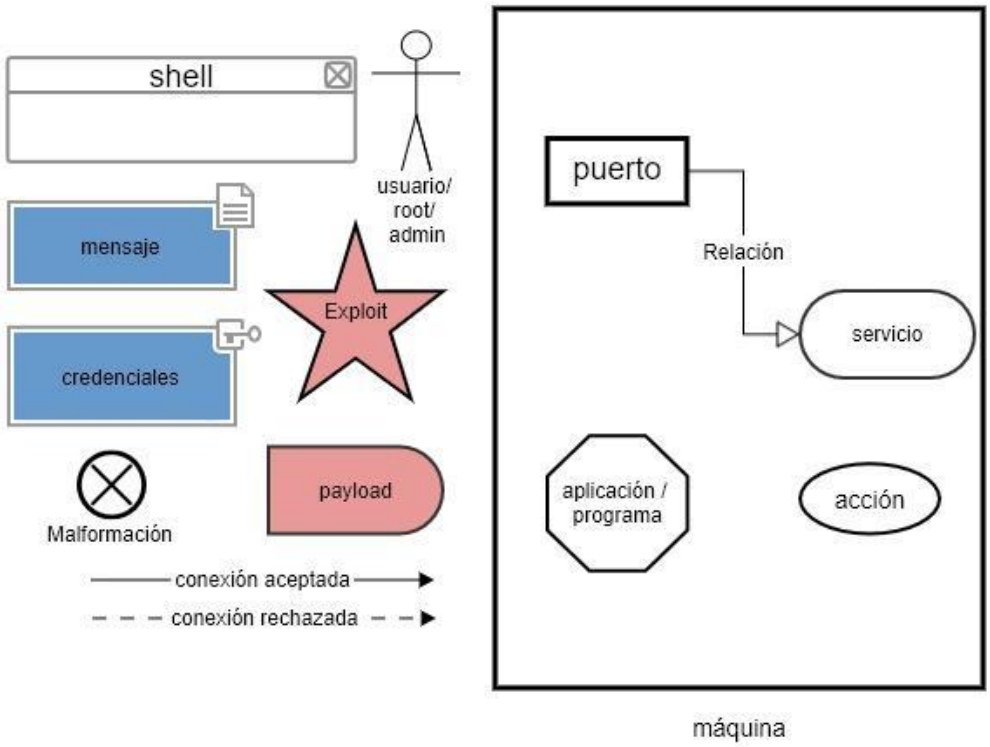



Ilustración 20. Tercer prototipo.

Aparecen por lo tanto en este tercer prototipo integradas en la máquina las aplicaciones y las acciones. Además, se añade una figura que representa las malformaciones y se eliminan los elementos excedentes, que son las bases de datos, las descargas y las conexiones externas.

### 3.3.5. Iteración sobre VSP53 (CVE-2016-2776)

En el puerto 53 se identifica una versión vulnerable del servicio DNS Bind. Múltiples versiones de Bind 9 albergan un defecto al crear una respuesta a determinadas peticiones. Dicho defecto aparece cuando la respuesta tiene un tamaño que supera el reservado por el servicio, y acepta entonces consultas de origen desconocido. Esto permite a los atacantes realizar ataques de denegación de servicio (DoS).

El error se debe a las comprobaciones de tamaño del buffer reservado para la respuesta. Tienen lugar dos comprobaciones: que el tamaño reservado supere el de las cabeceras (estándar 12 Bytes) y que supere también el tamaño de la respuesta preparada, sin tener en cuenta las cabeceras en esta última. Para el tamaño estándar de respuesta reservado (512 B), puede preparar una respuesta de 501B que aprueba las comprobaciones (reservado > cabeceras (512 > 12) y reservado > respuesta (512 > 501)), sin embargo, se

procederá a escribir 501 + 12 B = 513B de respuesta en el buffer reservado de 512B, provocando de esta forma un error en el servicio.

Es posible asegurar una respuesta mayor a 500B enviando una petición con una firma inválida de gran tamaño, al ir esta incluida en el mensaje.

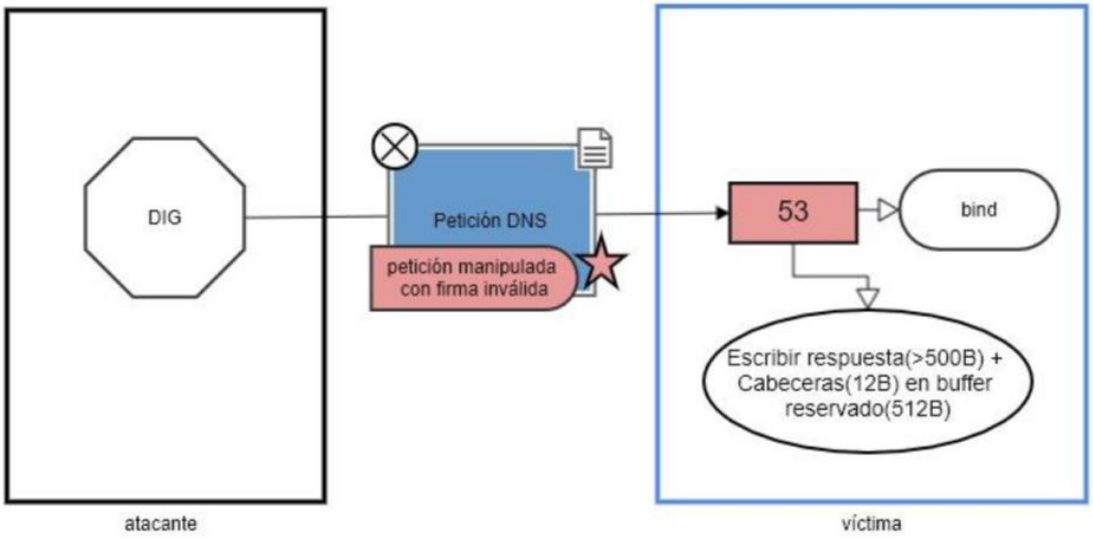



Ilustración 21. Tercer prototipo - VSP53.

### 3.3.6. Iteración sobre VSP3306 (CVE-2009-4484)

El servicio vulnerable en este caso es MySQL (versiones 5.0.x-5.0.90, 5.1.x-5.1.43 y 5.5.x-5.5.0-m2), debido a múltiples vulnerabilidades de sobreescritura del búfer en la función CertDecoder::GetName. Es posible establecer una conexión SSL con el servicio MySQL a través del puerto 3306 y enviar un certificado X.509 de cliente en el que el campo del nombre está corrompido, lo que permite a un atacante ejecutar código arbitrario o causar una denegación de servicio (Ilustración 22).

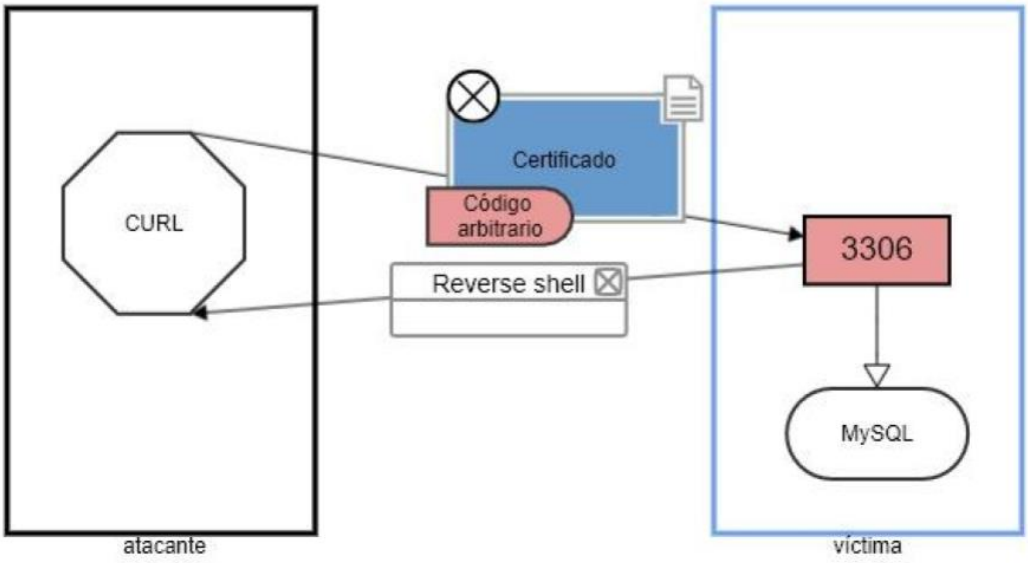



Ilustración 22. Tercer prototipo - VSP3306.

Todas las vulnerabilidades son fácilmente representables con sus elementos mediante este tercer prototipo. No obstante, ciertos detalles se echan en falta para algunas de ellas y hacen que se constituya un último prototipo:

1) Importancia de los colores para distinguir rápidamente los elementos del diagrama que son comprometedores o comprometidos. Es el caso del puerto vulnerable, en rojo desde los primeros diagramas, y no sin embargo el de las malformaciones, que a pesar de ser elementos comprometedores no fueron incluidas en rojo.

2) Los exploits deben situarse en el elemento que explota la vulnerabilidad y mantener concordancia con el caso de estudio, Metasploit, identificando así los ataques que tienen exploit asociado en Metasploit. De esta forma los exploits antes situados en elementos comprometidos pasan a situarse en los elementos comprometedores. Asimismo, dichos exploits aparecen únicamente cuando están disponibles en Metasploit, por lo que los elementos de ataque que no se encuentran en esta herramienta no deben incluir la estrella roja.

3) Inclusión del tipo de riesgo que provoca la vulnerabilidad representada, que puede ser alto, medio o bajo. En este caso el tipo de riesgo viene determinado por la herramienta de análisis de vulnerabilidades utilizada, OpenVAS, que clasifica y puntúa las vulnerabilidades que detecta según el riesgo que suponen dado su CVSS.

4) Especificación de las versiones de los servicios que son vulnerables. No todas las versiones de un servicio son vulnerables; Hay servicios que son afectados por una vulnerabilidad para algunas de sus versiones en concreto. Por ello, es necesario especificar qué versiones del servicio pueden ser atacadas de la forma representada.

5) Establecimiento de reglas que indican una manera de utilizar el modelo, a modo de manual.

Como se mostraba anteriormente en el apartado 3.1 y a modo de recordatorio, el modelo de diagrama final tiene el siguiente aspecto:

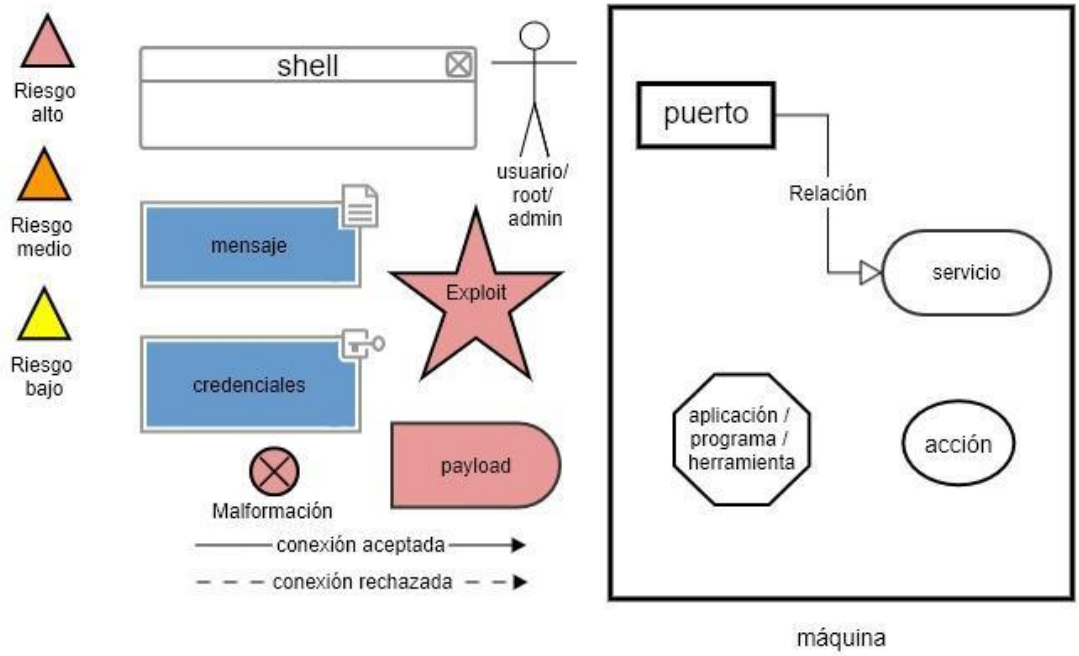



Ilustración 23. Modelo final propuesto

De igual forma, las reglas resultantes quedan se pueden resumir de la siguiente forma:

- Color negro para la máquina atacante. Color azul para máquinas inofensivas.
- Interacciones y relaciones de elementos dispuestas en orden de escritura (de izquierda a derecha y de arriba hacia abajo)
- Especificación de las versiones vulnerables del servicio que propician el ataque.
- Representación de las acciones dentro de la máquina que las realiza
- Representación de las aplicaciones dentro de la máquina que las posee
- Especificación del tipo de usuario cuando es conocido
- Color rojo para elementos comprometedores/comprometidos
- Representación de exploits disponibles en Metasploit dispuesta sobre el elemento o elementos final/es que explota/n la vulnerabilidad.
- Especificación del riesgo de la vulnerabilidad estudiada en el margen superior izquierdo.

# 4. Modelo final aplicado a las vulnerabilidades

En este apartado se muestran los diagramas finales, que ayudan a describir para cada puerto la vulnerabilidad y ataque escogidos. Esto nos va a servir para ver la utilidad del modelo propuesto y justificar el diseño final creado.

Como se mencionó anteriormente, no todas las vulnerabilidades ni todos los puertos de estudio fueron representados, ya que algunos no se encontraban en el informe de vulnerabilidades de OpenVAS o no tenían exploit o ataque asociado. Las vulnerabilidades finalmente estudiadas y representadas son la VSP21, VSP22, VSP23, VSP25, VSP53, VSP80, VSP514, VSP2121, VSP3306, VSP5432, VSP5900 y VSP6200. En la Tabla 1 y Tabla 2 se muestra la información recopilada para cada puerto y servicio, partiendo del análisis de OpenVAS. En ellas se resaltan las vulnerabilidades finalmente representadas.

La estructura que se utiliza sucesivamente para exponer los resultados es la siguiente: primero se especifica el identificador de la vulnerabilidad (Apartado 1.3.5), a continuación se muestra el diagrama final que representa la vulnerabilidad y el ataque escogidos, se indica cuál es la vulnerabilidad tal y como fue identificada por OpenVAS (siguiendo su nomenclatura) y por último se describe el diagrama.

## 4.1. VSP21 (CVE-2011-0762)

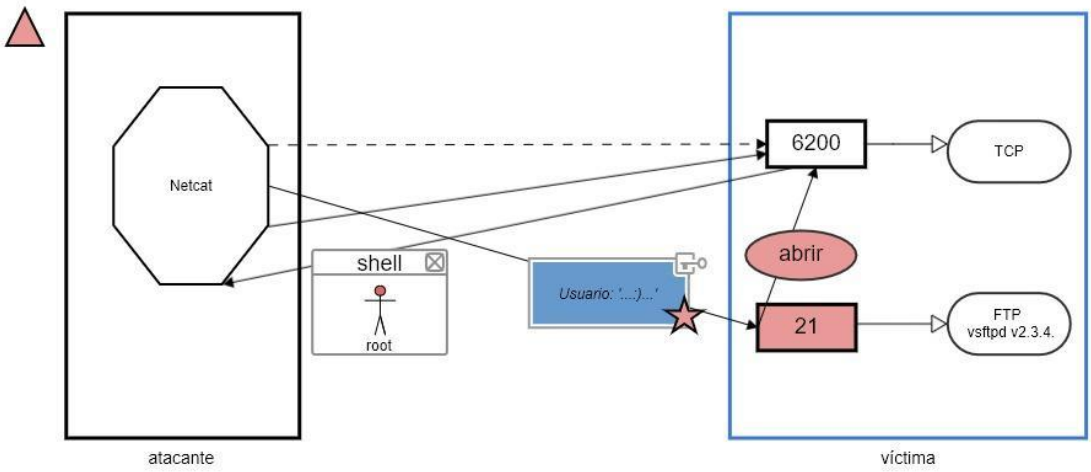



Ilustración 23. Diagrama final. VSP21

Vulnerabilidad detectada por OpenVAS: “CVE-2011-0762 - vsftpd Compromised Source Packages Backdoor Vulnerability”.

En primer lugar se aprecia el nivel de riesgo. La forma triangular del margen superior izquierdo del diagrama revela por su color rojo que la vulnerabilidad encontrada para el servicio del puerto 21 es de alto riesgo.

En segundo lugar, se distinguen dos máquinas que intervienen en el ataque: una máquina atacante (recuadro de borde negro, a la izquierda) y una máquina víctima (recuadro de borde azul, a la derecha). Dentro de la máquina atacante solo se observa un componente, una herramienta, que como sabemos se representa mediante un octágono. Se trata de la herramienta Netcat, que permite el envío y recibo de datos a través de comandos y abriendo puertos TCP/UDP. En cuanto a la máquina víctima, los elementos que se han utilizado son dos puertos con sus respectivos servicios y una acción que parte del puerto 21. Este puerto (21) aparece destacado en rojo, lo que indica que es el puerto comprometido. Se trata del puerto que alberga el servicio vsftpd v2.3.4 y es por lo tanto el foco de la vulnerabilidad y el que da paso al ataque. El otro puerto que entra en juego en la máquina víctima es el puerto 6200, representado en blanco.

Siguiendo el orden de los elementos y las relaciones, se observa que primero el atacante intenta conectarse a la máquina víctima por el puerto 6200 mediante Netcat, pero esta conexión es rechazada dado que este puerto se encuentra cerrado, por lo que esta conexión se representa con una flecha con línea discontinua. Seguidamente, intenta conectarse al puerto 21 introduciendo como credenciales un nombre de usuario que contiene la cadena ':)'. Al tratarse de credenciales, se ha utilizado el recuadro azul con llave para su representación. La estrella roja en este elemento indica que es un exploit que se encuentra en Metasploit. Debido a una vulnerabilidad en el código del servicio vsftpd, esta conexión es aceptada (flecha con línea continua) y se abre el puerto 6200 (acción comprometedora en rojo) con acceso a una shell en modo root.

El atacante puede ahora conectarse al puerto 6200 y tener acceso a la shell, por lo que esta vez, a diferencia de la primera conexión hacia este puerto, la conexión es aceptada y por lo tanto se representa con una flecha con línea continua.

Se concluye que el ataque tras enviar el exploit al puerto 21 permite que si el atacante se vuelve a conectar al puerto 6200 de la víctima, esta conexión le devuelva una shell en modo root, representada con el recuadro de shell y el usuario root contenido en su interior.

Como se presume por la estrella roja en el diagrama, se podría usar la herramienta Metasploit para buscar el exploit asociado a la versión del servicio estudiada o directamente a la vulnerabilidad y dirigirlo contra la máquina víctima. Exploit: exploit/unix/ftp/vsftpd_234_backdoor.

## 4.2. VSP22 (CVE-2018-15473)

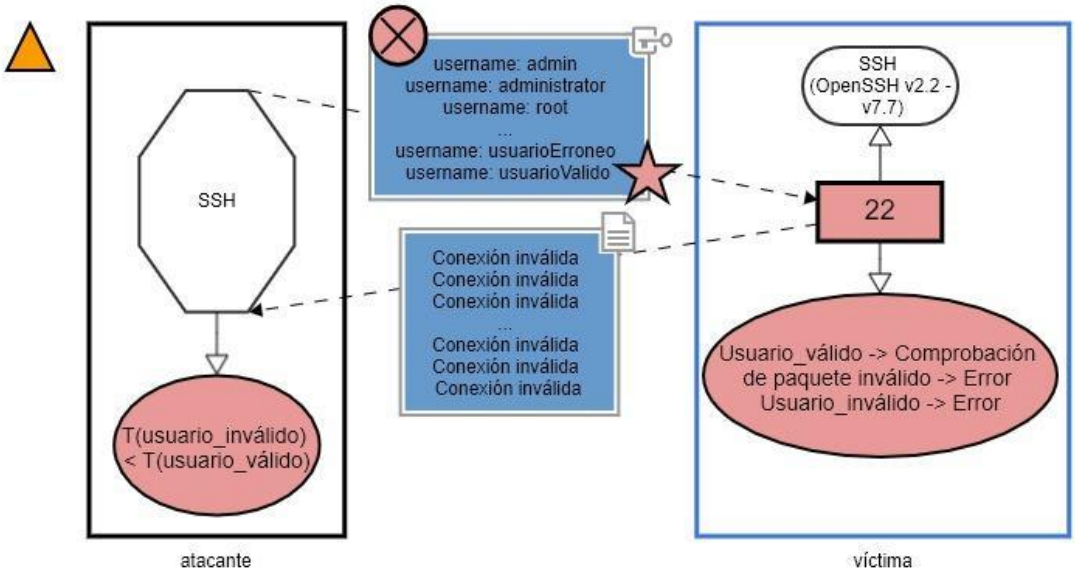



Ilustración 24. Diagrama final. VSP22

Vulnerabilidad detectada por OpenVAS: "CVE-2018-15473 - Open SSH User Enumeration Vulnerability-Aug18(Linux)"

Comenzando por el margen superior izquierdo, se observa que en este caso el nivel de riesgo que entraña la vulnerabilidad es medio (forma triangular de color naranja).

Entran en juego dos máquinas, la atacante y la víctima. Para representarlas se ha utilizado para la atacante un recuadro en negro a la izquierda y para la víctima un recuadro en azul a la derecha.

En el lado del atacante solo es necesaria una única herramienta, SSH. Por ello, en la figura de la izquierda se ha utilizado un octágono que representa SSH. Esto se debe a que el servicio vulnerable en este caso es OpenSSH, por lo que el atacante puede iniciar la conexión mediante SSH.

En la máquina víctima se encuentra un único puerto, el 22. Este puerto aparece en rojo, por lo cual se entiende que el puerto sirve de entrada para el ataque debido a una vulnerabilidad en el servicio que en él se encuentra, OpenSSH versiones 2.2-7.7 (servicio simbolizado con un rectángulo redondeado).

El servicio SSH requiere de nombre de usuario y contraseña, pero para efectuar el ataque son necesarios únicamente los nombres de usuario a probar, que son enviados en paquetes malformados, representados mediante el recuadro azul con llave al tratarse de credenciales de usuario y el círculo rojo con cruz para indicar la malformación. Al no incluir contraseña, la conexión es rechazada para todos los nombres de usuario (para conectarse se debe

indicar la contraseña). Por esta razón, la conexión rechazada se puede representar con una flecha con línea discontinua.

Refrescando lo ya comentado en el apartado Pruebas, desde la versión 2.2 hasta la 7.7 de OpenSSH. al intentar autenticar a un usuario con un paquete malformado, es más rápida la respuesta a un usuario inválido que a uno válido. Esto lo podemos ver en el óvalo rojo relacionado con el puerto 22, que indica los procesos que sigue el servicio de la víctima para procesar las peticiones. Por ello, al recibir respuesta del servidor comunicando las conexiones sin éxito (recuadro de mensaje en azul), el atacante recoge los tiempos de respuesta para los usuarios y los compara teniendo en cuenta que la respuesta para los usuarios válidos es la más lenta, lo que ha sido posible representar mediante el símbolo que indica una acción, el óvalo rojo, relacionado con el servicio SSH que realiza las conexiones y recibe las respuestas.

Como indica la estrella roja en el diagrama, para esta vulnerabilidad Metasploit tiene un exploit conocido. Se trata de "auxiliary/scanner/ssh/ssh_enumusers".

## 4.3. VSP23

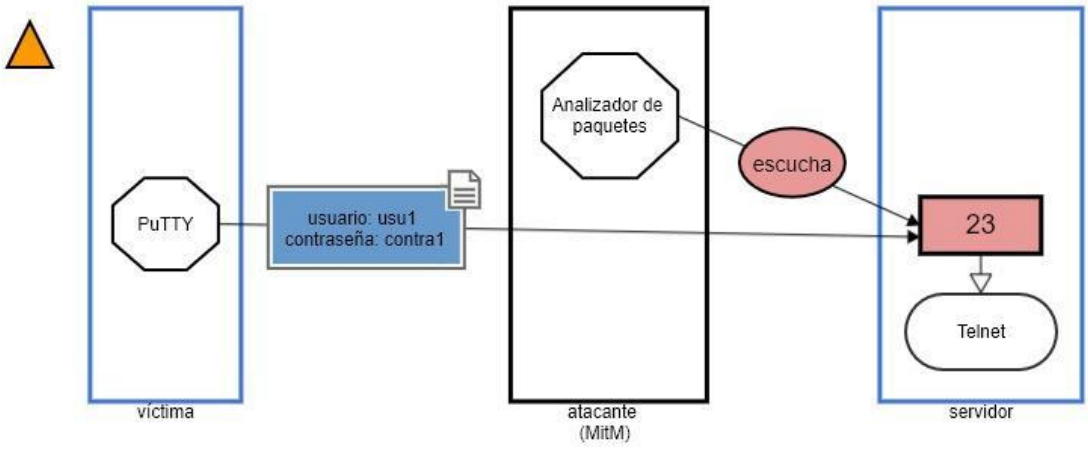



Ilustración 25. Diagrama final. VSP23

Vulnerabilidad detectada por OpenVAS: “Telnet Unencrypted Cleartext Login”

El riesgo de esta vulnerabilidad ha sido clasificado como medio, lo que es posible representar, siguiendo el modelo, mediante el triángulo naranja en la esquina superior izquierda.

En esta ocasión, se observa tres máquinas distintas: a la izquierda y en azul, la máquina víctima, en medio y en negro, la máquina atacante, que actúa como Man in the Middle, y a la derecha y en azul, el servidor, aunque este último podría tomarse también como máquina víctima (es la máquina con la que interactúa el atacante). En la máquina víctima hace falta una herramienta como PuTTY, que es un cliente que permite realizar conexiones Telnet y SSH, entre otras. Tal y como indica el modelo, esta herramienta se ha representado con un

octágono en la máquina víctima. En la máquina atacante se necesita otra herramienta, en este caso que permita la recogida y análisis de tráfico de red. Por ello encontramos un octágono que representa a un analizador de paquetes. Por último, en la máquina servidor se encuentra el puerto vulnerable 23 (rectángulo rojo), con el servicio Telnet.

Como se ha comentado y se percibe en el diagrama, el ataque es de tipo MitM (Man in the Middle). El atacante escucha (acción en óvalo rojo) el puerto del servidor telnet mediante el analizador de paquetes. La comunicación vía Telnet no es encriptada, por lo que puede ser interceptada por un tercero, que obtiene los datos en claro. La víctima envía credenciales de inicio de sesión por telnet, que al no ser encriptadas se tratan como un mensaje de texto en claro (recuadro de mensaje en azul). Por eso mismo, la información está comprometida a ser obtenida por el atacante.

En este caso no hay un exploit para esta vulnerabilidad en Metasploit. Se puede explotar mediante la escucha y obtención de paquetes dirigidos al puerto 23 con un analizador de paquetes, como por ejemplo WireShark.

## 4.4. VSP25 (CVE-2015-4000)

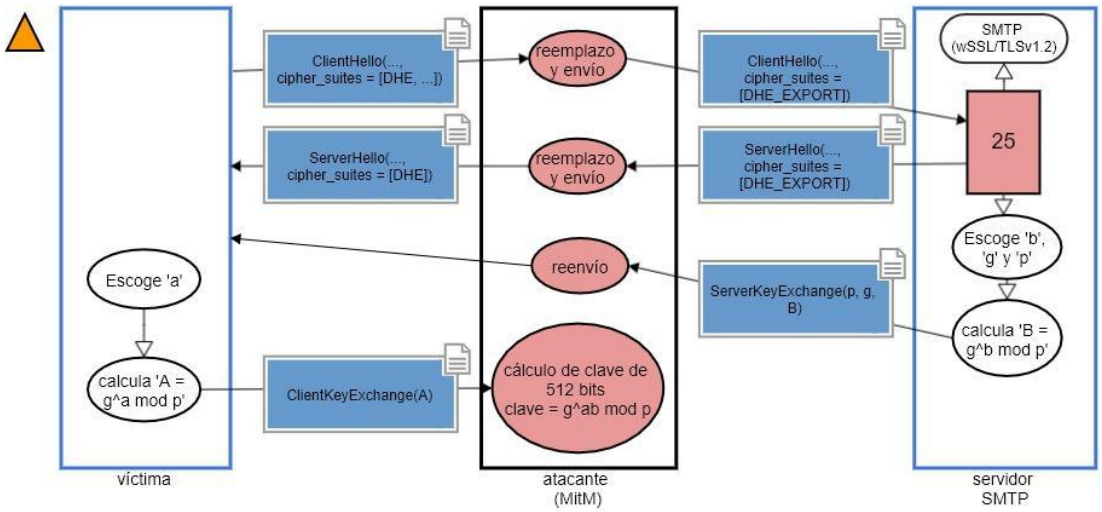



Ilustración 26. Diagrama final. VSP25

Vulnerabilidad detectada por OpenVAS: CVE-2015-4000 – “SSL/TLS: 'DHE_EXPORT' Man in the Middle Security Bypass Vulnerability (LogJam)”

El riesgo detectado que supone esta vulnerabilidad es clasificado como medio, por lo que se observa un triángulo naranja en el margen superior izquierdo.

Se percibe que se trata de nuevo una vulnerabilidad de tipo Man in the Middle, con la representación de las siguientes máquinas: a la izquierda y en azul, la máquina víctima, en medio y en negro, la máquina atacante, que actúa como Man in the Middle, y a la derecha y en azul, el servidor de correo SMTP. En este caso, debido a la complejidad de la

vulnerabilidad, los únicos elementos que hay en las máquinas son acciones, a excepción de la máquina víctima, que alberga el puerto vulnerable 25 (rectángulo rojo).

Esta vulnerabilidad es hallada en la versión 1.2 y en anteriores de TLS, que aceptan conexiones mediante el conjunto de cifrado DHE_EXPORT. Esto permite a un atacante realizar un man-in-the-middle y degradar la fortaleza de las claves usadas en el intercambio de claves Diffie-Hellman al forzar el uso del conjunto de cifrado DHE_EXPORT, facilitando el descifrado de las claves.

Este es un claro ejemplo de diagrama con múltiples acciones, que permiten explicar el proceso por el que pasa la información en el lado del cliente, el servidor y el atacante.

Primero, la víctima envía un ClientHello al servidor con un conjunto de posibles cifrados para el intercambio de claves, tal y como vemos en la Ilustración 27, en el primer recuadro de mensaje azul. El atacante debe reemplazar en el ClientHello el conjunto de cifrado DHE por DHE_EXPORT y reenviarlo al servidor (indicado en el óvalo rojo), que trabaja con claves débiles de 512 bits (las más usadas son incluso ya conocidas públicamente).

A continuación, el servidor recibe el mensaje, con el conjunto de cifrado cambiado. Lo recibe por el puerto 25, por lo que la flecha apunta directamente a este, que da al servicio SMTP. El servidor entonces responde con un ServerHello, representado como es de esperar con un recuadro azul de mensaje que el atacante debe reenviar al cliente reemplazando DHE_EXPORT por DHE, acción también representada con un segundo óvalo rojo en la máquina atacante.

El servidor calcula ‘B = g^b mod p’, acción que se ha representado mediante óvalos tal y como indica el modelo. Tras ello envía los valores de ‘B’, ‘g’ y ‘p’ (recuadro de mensaje azul). El siguiente óvalo rojo en la máquina atacante indica que el atacante recibe dichos valores y los reenvía a la víctima. La víctima con estos valores escoge ‘a’ y calcula ‘A = g^a mod p’ (acción en óvalo blanco) y envía el valor de ‘A’ de vuelta (recuadro azul de mensaje). Como refleja el último óvalo rojo del atacante, de esta forma puede obtener la clave de 512 bits sin que el cliente sepa siquiera que la conexión con el servidor ha sido mediante DHE_EXPORT.

No hay exploit asociado en Metasploit. Sin embargo, tal y como se ha comentado y tal y como se puede observar en el diagrama, es posible efectuar un ataque de tipo Man in the Middle.

## 4.5. VSP53 (CVE-2016-2776)

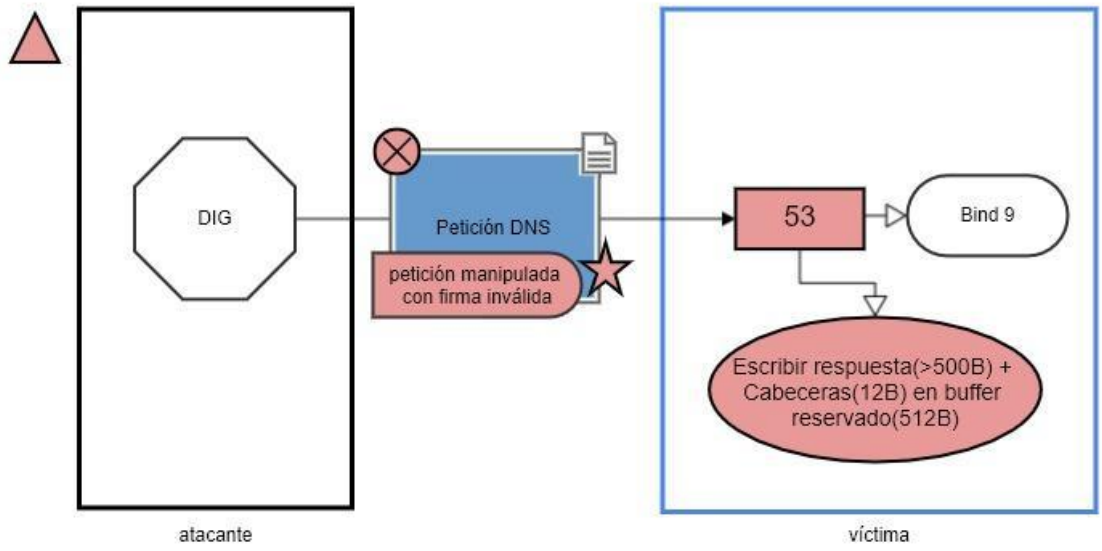



Ilustración 27. Diagrama final. VSP53

Vulnerabilidad detectada por OpenVAS: CVE-2016-2776 – “ISC BIND 'buffer.c' Assertion Failure Denial of Service Vulnerability (Linux)”

Como se indica en el triángulo rojo en la esquina superior izquierda, el riesgo de esta vulnerabilidad ha sido calificado como alto.

Se observan dos máquinas partícipes. A la izquierda y en negro está la máquina atacante, que tiene solamente una herramienta, representada mediante el correspondiente octágono. A la derecha y en azul se encuentra la máquina víctima, en la que se indica el puerto vulnerable 53 (rectángulo en rojo) y el servicio Bind 9 (rectángulo redondeado).

En el diagrama se observa que el atacante efectúa una petición DNS mediante la utilidad Unix DIG que permite efectuar consultas DNS. Dicha petición (recuadro de mensaje en azul) está malformada, lo que se identifica, siguiendo el modelo, con un círculo rojo con cruz, y lleva un payload con una firma inválida. El payload viene representado con el símbolo de la bala roja y en su interior indica qué tipo de payload es (firma inváilida). Al recibir la petición malformada, el servicio bind albergado en el puerto 53 efectúa una acción comprometedora (óvalo rojo) que provoca un error en el servicio que lo inhabilita. Para el tamaño estándar de respuesta reservado (512 B), puede preparar una respuesta de 501B que aprueba las comprobaciones (reservado > cabeceras (512 > 12) y reservado > respuesta (512 > 501)), sin embargo, se procederá a escribir 501 + 12 B = 513B de respuesta en el buffer reservado de 512B, provocando un error. Esto se incluye dentro de la representación de la acción comprometedora, es decir, dentro de el óvalo rojo, que aparece como último elemento que escenifica el ataque, situado en concreto en el lado de la víctima.

Para esta vulnerabilidad existe el siguiente exploit en Metasploit: auxiliary/dos/dns/bind_tsig.

## 4.6. VSP80 (CVE-2017-9798)

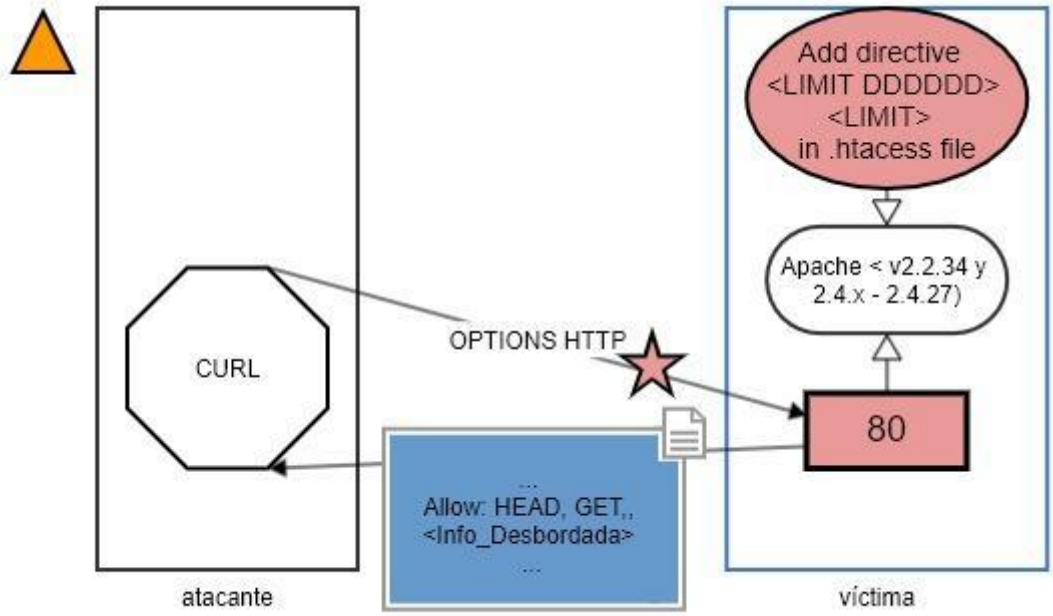



Ilustración 28. Diagrama fina. VSP80

Vulnerabilidad detectada por OpenVAS: CVE-2017-9798 – "Apache HTTP Server OPTIONS Memory Leak Vulnerability (Optionsbleed)"

El triángulo naranja del margen superior izquierdo indica que el riesgo de la vulnerabilidad es calificado como medio.

Aparece a la izquierda la máquina atacante y a la derecha la máquina víctima, en azul. Para reproducir el ataque escenificado se necesita en la máquina atacante únicamente la herramienta CURL, que permite hacer peticiones HTTP. Es por ello que solo se reconoce una forma en esta máquina: un octágono que indica que se trata de la herramienta CURL. En la máquina víctima se identifica el puerto 80 como vulnerable (en rojo) con el servicio Apache. Esta vulnerabilidad, conocida como Optionsbleed afecta a versiones de Apache hasta la versión 2.2.34 y desde la 2.4 a la 2.4.27 (indicado en el rectángulo redondeado).

La vulnerabilidad es de tipo use after free, explotada cuando se usa un limitador para un tipo de petición inexistente en el servidor en el archivo .htaccess de Apache. Por ello, se indica en la máquina víctima una acción comprometedora (óvalo rojo): añadir la directiva LIMIT para limitar una opción HTTP inexistente (DDDDDD) en el archivo de configuración .htacess.

En el diagrama se escenifica el ataque llevado a cabo mediante un comando CURL para enviar una petición aceptada (flecha de línea continua) hacia el puerto 80 de la máquina víctima. Dicha petición es de tipo Options, que sirve para obtener los tipos de peticiones que acepta el servicio que recibe la petición, ya que los devuelve listados en la respuesta.

Sin embargo, debido a la vulnerabilidad en este servicio, en este caso es posible que la respuesta contenga información desbordada, tal y como se aprecia en el diagrama representativo, en el mensaje que recibe el atacante (recuadro de mensaje en azul), concretamente en la sección "allow" de la respuesta.

Existe un exploit asociado en Metasploit (estrella roja sobre la petición options), auxiliary/scanner/http/apache_optionsbleed.

## 4.7. VSP514 (CVE-1999-0651)

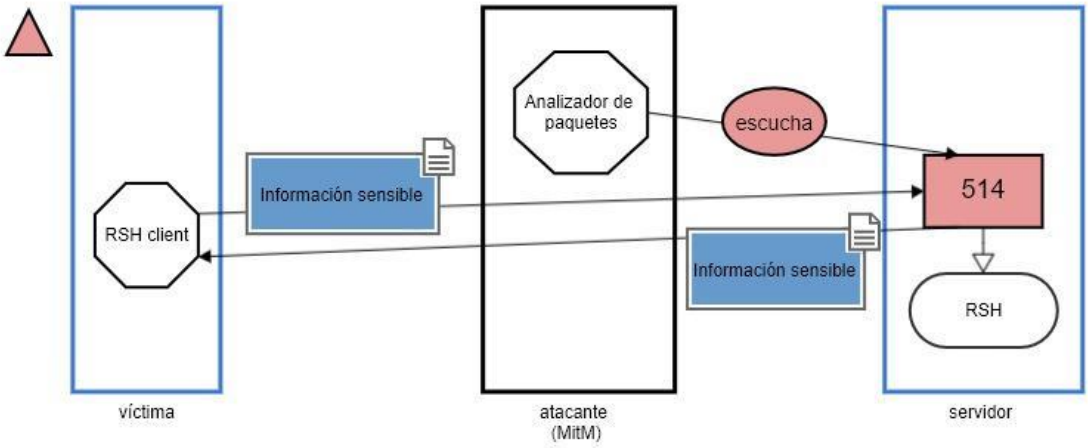



Ilustración 29. Diagrama final. VSP514 - Mensajes de texto en claro.

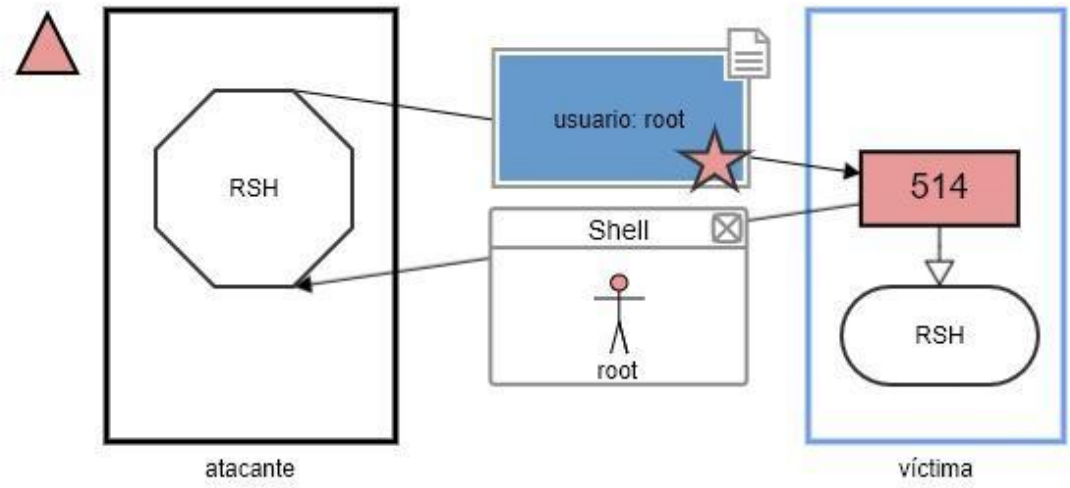



Ilustración 30. Diagrama final. VSP514 - Inicio de sesión sin contraseña

Vulnerabilidad detectada por OpenVAS: CVE-1999-0651 – “rsh Unencrypted Cleartext Login”

El riesgo de esta vulnerabilidad ha sido calificado como alto, como se indica con las formas triangulares rojas.

Este servicio encontrado tiene dos vulnerabilidades asociadas al código CVE-1999-0651. La primera está representada en la Ilustración 30. Consta de tres máquinas: a la izquierda la máquina víctima en azul, al centro la máquina atacante en negro y a la derecha el servidor en azul. Esta distribución de máquinas se ha visto anteriormente y facilita identificar que se trata de un ataque de tipo Man in the Middle debido a la disposición, número y colores de las máquinas representadas.

El atacante utiliza un analizador de paquetes para escuchar en el puerto 513 de la máquina víctima, puerto vulnerable con el servicio RSH. Es por ello que en la máquina atacante encontramos únicamente un octógono que representa al analizador de paquetes.

Cliente y servidor intercambian información en mensaje en texto claro (recuadros de mensaje en azul) por RSH, con lo cual el atacante (MitM) puede obtener datos de inicio de sesión e información sensible.

Además, esta vulnerabilidad tiene una segunda consecuencia (Ilustración 31), la misma que se encuentra en el puerto 513 (rlogin), ya que rsh usa rlogin para los inicios de sesión, que no requiere de contraseña. El atacante utiliza RSH para intentar conectarse al servicio por el puerto 514 indicando el usuario root en texto en claro y sin contraseña, lo cual está disponible en un exploit de Metasploit (estrella roja). Encontramos esto representado mediante un octógono que identifica la herramienta RSH en el lado del atacante, un recuadro de mensaje en azul (por tratarse de texto en claro) con un nombre de usuario root y una flecha con línea continua, dado que esta conexión es aceptada. El atacante obtiene shell con privilegios de root.

Por lo tanto, se concluye que debido a esta vulnerabilidad, es posible para un atacante tanto ver en texto en claro los mensajes que se intercambian entre cliente y servidor, haciendo de MitM, como también es posible para un atacante conectarse al servicio sin contraseña.

En el segundo diagrama para esta vulnerabilidad se observa la existencia de exploit en Metasploit. En este caso hay tres exploits presentes: auxiliary/scanner/rservices/rsh_login, auxiliary/scanner/rservices/rlogin_login y auxiliary/scanner/rservices/rexec_login, que permiten iniciar sesión en el shell remoto (RSH).

## 4.8. VSP2121 (CVE-2010-4221)

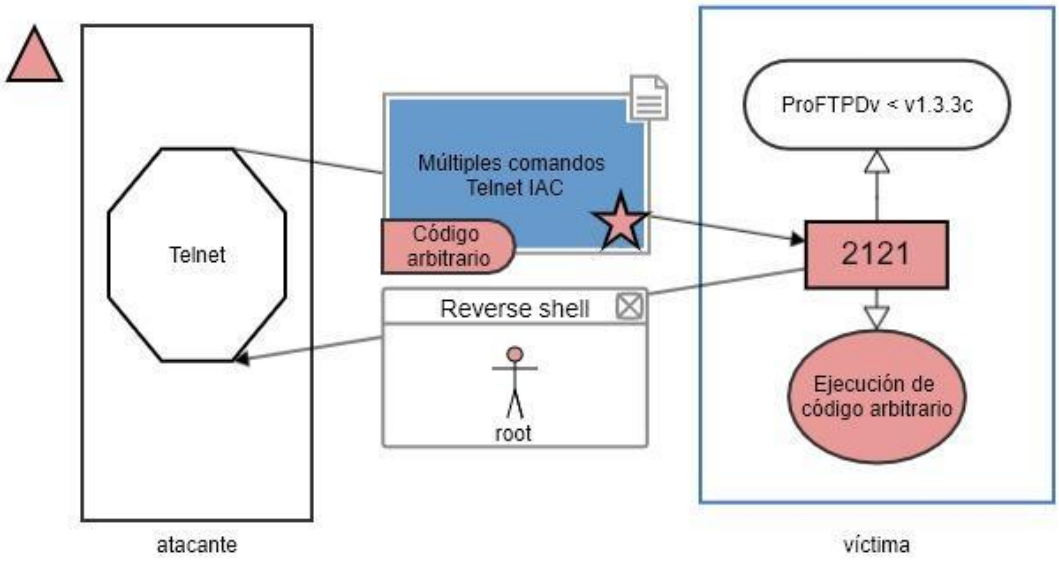



Ilustración 31. Diagrama final VSP2121

Vulnerabilidad detectada por OpenVAS: CVE-2010-4221 – "ProFTPD Multiple Remote Vulnerabilities"

Como se indica con el triángulo rojo en el margen superior izquierdo del diagrama, el riesgo de esta vulnerabilidad ha sido calificado como alto.

Se distinguen dos máquinas: atacante a la izquierda y víctima a la derecha, en azul. La máquina atacante actúa mediante Telnet para enviar comandos Telnet IAC, por lo que encontramos el octógono que representa a esta herramienta en la máquina atacante. En la máquina víctima se encuentra únicamente el puerto 2121, representado en rojo ya que es vulnerable debido al servicio ProFTPD, versiones menores a la v1.3.3c, y que realizará una acción comprometedora que se describe posteriormente, representada con su óvalo rojo siguiendo el modelo.

El envío de múltiples comandos Telnet IAC (recuadro de mensaje azul del diagrama) a la víctima permite que un atacante corrompa la memoria de la pila y ejecute código arbitrario, acción comprometida mostrada mediante un óvalo rojo. Dicho código arbitrario es el que envía el atacante en un payload, que como podemos ver ha podido ser incluido en el diagrama siguiendo nuestro modelo. Vemos el payload dentro del mensaje y en forma de una bala roja, al igual que encontramos una estrella roja. Esta estrella indica que es posible enviar estos comandos con payload a través de la herramienta Metasploit.

Hay dos exploits asociados a esta vulnerabilidad en Metasploit, dependiendo del sistema operativo al que se dirige: exploit/freebsd/ftp/proftp_telnet_iac para freebsd y exploit/linux/ftp/proftp_telnet_iac para linux.

## 4.9. VSP3306 (CVE-2009-4484)

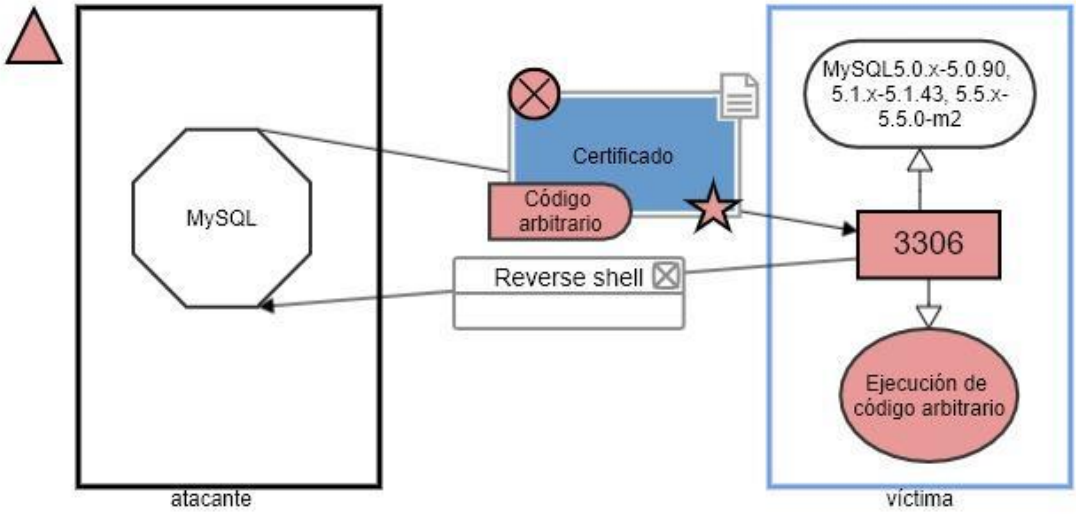



Ilustración 32. Diagrama final. VSP3306

Vulnerabilidad detectada por OpenVAS: CVE-2009-4484 – "MySQL 5.0.51a Unspecified Remote Code Execution Vulnerability"

Esta vulnerabilidad tiene un riesgo asociado que ha sido calificado como alto, como indica la estrella roja en la esquina superior izquierda.

Se observan dos máquinas, una atacante a la izquierda y una víctima a la derecha. Primero se ha representado la herramienta que necesita tener la máquina atacante y el puerto vulnerable de la máquina víctima. El atacante utiliza la herramienta MySQL (octágono) para conectarse con el servicio MySQL por el puerto 3306 de la máquina víctima. Este puerto es vulnerable (rectángulo rojo) debido a conectar el servicio MySQL en las versiones expuestas en el rectángulo redondeado del diagrama.

El ataque es posible mediante el envío de un certificado de cliente (recuadro de mensaje azul) corrompido (círculo rojo con cruz), disponible en Metasploit (estrella roja), concretamente mediante el exploit "exploit/linux/mysql/mysql_yassl_getname", que permite a un atacante ejecutar código arbitrario, tal y como indica el payload en forma de bala roja, o causar una denegación de servicio. En el escenario que muestra la Ilustración 33, el atacante mediante MySQL envía hacia el puerto 3306 un certificado corrompido que alberga código arbitrario. El servicio ejecuta dicho código, acción comprometedora y por lo tanto representada en forma de óvalo rojo, y devuelve el acceso a una shell reversa.

## 4.10. VSP5432

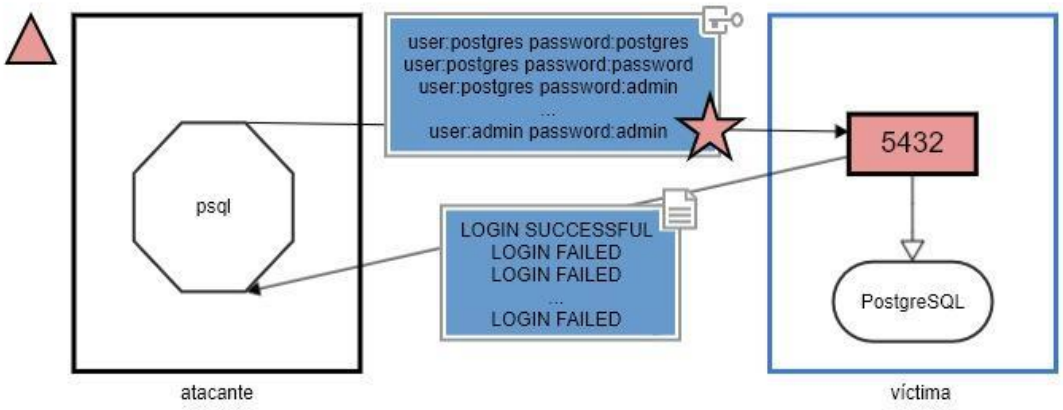



Ilustración 33. Diagrama final. VSP5432

Vulnerabilidad detectada por OpenVAS: PostgreSQL weak password

Como se indica en el diagrama, la vulnerabilidad provoca un alto nivel de riesgo, representado con una forma triangular roja.

A la derecha se observa una máquina atacante provista de únicamente una herramienta, psql (disponible para ejecutar comandos por terminal que comunican con el servicio PostgreSQL), y a la derecha una máquina víctima con el servicio PostgreSQL (rectángulo redondeado) en el puerto 5432, en forma de rectángulo rojo al tratarse del puerto vulnerable.

Se observa cómo un atacante se intenta autenticar en PostgreSQL con una serie de posibles credenciales, como observamos en el recuadro de mensaje de credenciales (sabemos que es de credenciales por el símbolo de llave en la esquina superior derecha) en azul en la Ilustración 34. Estas son fruto de combinar frecuentes nombres de usuario y contraseñas para el servicio en cuestión (disponible mediante el exploit "auxiliary/scanner/postgres/postgres_login" de Metasploit, como indica la estrella roja). Como consecuencia, la máquina atacante recibe el resultado de las credenciales que han funcionado para poder conectarse a PostgreSQL (recuadro de mensaje en azul).

Se concluye que un atacante puede obtener las credenciales de sesión mediante un ataque de fuerza bruta.

## 4.11. VSP5900

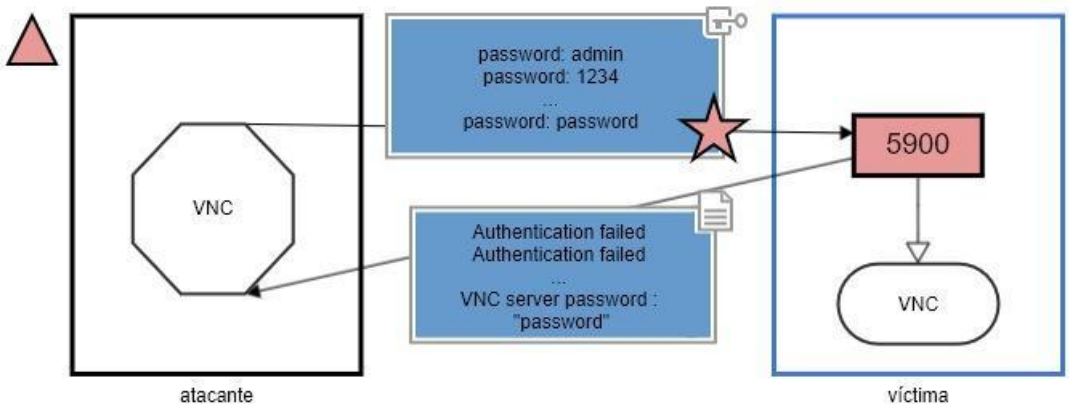



Ilustración 34. Diagrama final. VSP5900

Vulnerabilidad detectada por OpenVAS: VNC Brute Force Login

El riesgo asociado a esta vulnerabilidad es de nivel alto (triángulo rojo).

Se distinguen dos máquinas: a la izquierda la atacante, con solo una herramienta (octágono que refleja el software VNC), y a la derecha la máquina víctima, con el puerto 5900 vulnerable debido al servicio VNC, representados con el rectángulo rojo y el rectángulo redondeado respectivamente.

La vulnerabilidad expuesta permite conectarse a VNC con una contraseña común (“password”). En la ilustración se representa cómo un atacante puede obtener las credenciales de sesión mediante un ataque de fuerza bruta. Para ello, utiliza la herramienta VNC como cliente para conectarse al servidor VNC de la víctima por el puerto 5900, probando con múltiples combinaciones de contraseñas, tal y como hemos reflejado a través del rectángulo azul con llave, que en este caso no requieren de nombre de usuario, y que se pueden enviar a través de un exploit de Metasploit (estrella roja en el recuadro), "auxiliary/scanner/vnc/vnc_login". Como resultado, el atacante recibe respuesta positiva para las contraseñas válidas (recuadro de mensaje azul).

## 4.12. VSP6200 (CVE-2011-0762)

Este puerto se ve comprometido por la vulnerabilidad vista para el puerto 21 (CVE-2011-0762), de tal forma que el puerto 6200 actúa como puerta trasera que el atacante utiliza para establecer una conexión con la máquina víctima.

# 5. Discusión de resultados

En esta sección se describe aquello que se consigue con el modelo propuesto y la utilidad que tiene a la hora de crear diagramas que representen vulnerabilidades y ataques. Para cada una de las utilidades del modelo, primero se presenta una frase que la resume y a continuación se desarrolla el contenido de forma más extensa.

**Fácil identificar el tipo de ataque (MitM, DoS, Fuerza bruta, etc).**

Observando la disposición y el tipo de elementos incluidos en un diagrama determinado es posible identificar qué tipo de ataque está siendo representado. Por ejemplo, en un ataque de Man in the Middle la máquina atacante (en negro) se posiciona entre dos máquinas inofensivas (en azul), a menudo un servidor y una máquina víctima, haciendo de intermediaria recibiendo/enviando paquetes. Por su parte, un ataque de fuerza bruta se distingue mediante un elevado número de posibles nombres de usuario y/o contraseñas que se prueban contra la máquina víctima con objeto de obtener aquellos que resultan válidos. Los ataques de denegación de servicio suelen implicar el envío de paquetes malformados y como resultado el servicio atacado ejecuta una acción comprometedora, que puede ser el simple hecho de procesar dichos paquetes, por la cual queda inhabilitado.

**Fácil identificar el resultado u objetivo del ataque (Robo de credenciales, bloqueo de servicio, colapso de sistema, acceso remoto, etc).**

Las últimas interacciones que comprenden los diagramas revelan la finalidad o resultado del ataque. Así pues, el robo de credenciales se observa en una de las últimas interacciones desde la máquina víctima a la atacante, en el momento en el que el atacante recibe las credenciales a consecuencia de la explotación de la vulnerabilidad en cuestión. De igual forma, cuando el resultado es el acceso remoto a la máquina víctima, esto se observa en una de las últimas interacciones representadas, comúnmente cuando la máquina atacante recibe de la víctima el acceso a una shell de comandos.

**Visualización de los elementos o factores clave en la máquina víctima.**

En los diagramas se encuentran los elementos de la máquina víctima que posibilitan el ataque y que por tanto forman parte de la vulnerabilidad. Debido a la omisión de elementos no partícipes en la vulnerabilidad ni en el ataque, los diagramas reflejan en la máquina víctima únicamente aquellos elementos que sí influyen y es posible diferenciar concretamente aquellos comprometidos y/o comprometedores.

**Distinción entre vulnerabilidades explotadas con y sin Metasploit.**

Ciertas vulnerabilidades no cuentan con un exploit disponible en Metasploit. Los exploits de Metasploit tienen una representación concreta según el modelo propuesto (una estrella roja), por lo que la presencia de esta en el diagrama supone una forma de reconocer que existe la forma de llevar a cabo el ataque mediante un exploit de Metasploit.

**A mayor número de elementos, menor abstracción.**

El número de elementos es ilimitado, a gusto del creador de los diagramas. Para vulnerabilidades y ataques que requieren un mayor nivel de detalle habrá más formas visuales y descripciones que para aquellos más simples y generales. Por ejemplo, un flujo de ataque será más descriptivo cuantas más acciones incluya en el diagrama.

**Escalabilidad horizontal (aumento de número de máquinas).**

Si bien es cierto que el caso de uso es la iniciación de un usuario al pentesting mediante Metasploit (atacante) y Metasploitable2 (víctima), el modelo permite añadir terceras máquinas atacantes y/o víctimas para representar otro tipo de ataques. Por ejemplo, un ataque DDoS requiere de diversas máquinas controladas por el atacante para denegar un servicio en una máquina víctima, para lo que se podrían añadir varias máquinas partícipes de una botnet a la hora de elaborar el diagrama. Esto lo podemos ver, por ejemplo, en la ilustración X en el apartado TAL.

**Escalabilidad vertical (mayor número de elementos).**

Tal y como se comentó anteriormente, a mayor número de elementos, mayor es el nivel de detalle. Siempre se puede añadir más elementos al diagrama. Dependerá del peso que tenga la descripción frente a la simplicidad. Esto lo podemos ver, por ejemplo, en la ilustración X en el apartado TAL.

**Compatibilidad con otras herramientas de creación de diagramas.**

Se han escogido formas comunes en la creación de diagramas, por lo que hay muchas opciones de herramientas que permitirían la utilización del modelo propuesto.

# 6. Conclusiones y líneas futuras

Con esta sección se finaliza el cuerpo de la memoria.

En primer lugar, se desglosan las conclusiones del trabajo realizado. Para cada una de las conclusiones, primero se presenta una frase que la resume y a continuación se desarrolla el contenido de forma más extensa.

Por último, se describen algunas ideas de futuros trabajos o estudios que pueden realizarse a partir de lo aquí recopilado. Este apartado comparte la misma estructura que el anterior (Conclusiones).

## 6.1. Conclusiones

**Identificación y clasificación de elementos**

Como primer resultado del trabajo podemos señalar la identificación y clasificación de elementos (Apartado Objetivos, O3). De acuerdo con la metodología que se ha seguido para crear el modelo (Apartado 3.2), se han identificado los elementos que deben formar parte a la hora de representar una vulnerabilidad y su ataque. Gracias a la recopilación de información sobre las metodologías de aprendizaje existentes (Apartado Objetivos, O1) y las vulnerabilidades (Apartado Objetivos, O2) y las diferentes iteraciones del modelo, se ha conseguido clasificar los tipos de elementos de forma que quedan distribuidos en grupos (p.ej. malformaciones) y se les asigna una forma que los represente (p.ej. círculo rojo con cruz) y de forma que los diagramas mantienen concordancia entre ellos.

Además de identificar los elementos que se usan para representar las vulnerabilidades (p.ej. puertos o archivos de configuración), se han identificado los elementos que intervienen en un vector de ataque (p.ej. payloads) para añadir la posibilidad de escenificar los ataques a las vulnerabilidades mediante el modelo propuesto (Apartado 3).

**Limitaciones y Adaptabilidad**

La naturaleza del modelo es adaptable, dada la metodología seguida para su creación (Apartado 3.2). Esto hace que el modelo se pueda mejorar con cada una de las iteraciones. Sin embargo, esto mismo hace que el modelo esté sesgado al conjunto de vulnerabilidades que finalmente se representan. Por lo tanto, nunca podremos decir que el modelo es capaz de representar cualquier vulnerabilidad hasta no haberlo probado con todas, lo cual resulta imposible debido al gran número de vulnerabilidades que existen y a las nuevas que se descubren.

**Facilitación del aprendizaje con Metasploit**

Tras comprobar los resultados de la aplicación del modelo (4) se puede concluir que el modelo consigue representar vulnerabilidades y ataques con y sin Metasploit. El hecho de haber tenido en cuenta todos los elementos que propician y forman parte de un ataque ha permitido que se pueda entender cómo, de dónde y por qué obtiene Metasploit los resultados que muestra al usuario, tal y como se pretendía lograr con el objetivo O4 (Apartado 1.2).

## 6.2. Líneas futuras

**Estudio de vulnerabilidades no incluidas y posteriores versiones del REVEX**

Tal y como se ha mencionado anteriormente en el trabajo, la inclusión de la metodología empleada para la creación del modelo permite adoptar los pasos seguidos para evolucionar el modelo según se utilice para nuevos casos de puertos, vulnerabilidades y ataques de estudio (Apartado Metodología de trabajo empleada en el TFG). Esto se debe a que hay miles de casos de estudio para los que se puede emplear el modelo y que no han sido estudiados en el contexto de este trabajo, por lo que REVEX se podría utilizar para estudiar y escenificar dichos casos de estudio y podrían identificarse carencias o mejoras en el modelo propuesto.

**Uso del modelo para la creación de ejercicios o actividades de aprendizaje de seguridad**

Mediante el análisis y clasificación de vulnerabilidades y ataques llevado a cabo en este trabajo se persigue una vía de aprendizaje de iniciación a la seguridad con un modelo representativo. Esto es debido a que visualmente resulta más sencillo familiarizarse y aprender las distintas causas de vulnerabilidad y las formas de atacarlas. Por ello, el presente trabajo podría utilizarse en actividades didácticas tales como el reconocimiento de vulnerabilidades, la creación de diagramas o la identificación de componentes que faltan y de errores expuestos a propósito, entre otras.

**Integración del modelo con Metasploit**

A partir de aquí se podría programar una herramienta que, a través de la información obtenida desde Metasploit al ejecutar cierto ataque, ofreciera un enfoque visual con los elementos que componen el modelo.

**Integración en modelos de ML**

Con objeto de realizar automáticamente numerosos diagramas de ataques y vulnerabilidades e incluso para nuevos casos, se podría entrenar un modelo de Machine Learning capaz de identificar y disponer los elementos en diagramas según el modelo propuesto. La mayor dificultad de esto es que se necesitaría un gran conjunto de diagramas realizados con anterioridad con el que entrenar y validar dicho modelo.

# Referencias


[1] imperva (Accedido el 24/09/2020) *Vulnerability assesment* https://www.imperva.com/learn/application-security/vulnerability-assessment/

[2] López Muñoz, Gema (2011) *Test de penetración* https://es.slideshare.net/gemalm/test-de-penetracin

[3] Almeida Coloma, C. L., & Pincay Párraga, J. A. (2018*). Implementación de un laboratorio de seguridad de informática para la realización de técnicas de ataque y defensa (Pentesting) en un ambiente real controlado, utilizando una distribución de Kali Linux dentro de la empresa industrial siderúrgica Andec SA* (Doctoral dissertation, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería En Networking y Telecomunicaciones).

[4] Uladzislau Murashka (2017). *Vulnerability Assessment vs. Penetration Testing: Know Who Is Who.* https://www.scnsoft.com/blog/vulnerability-assessment-vs-penetration-testing

[5] Munif Tanjim (2020). *Best Linux Distributions for Hacking and Penetration Testing.* https://www.oversight.network/forums/topic/4-best-linux-distributions-for-hacking-and-penetration-testing/

[6] Albaladejo Sánchez, J. G. (2019). *Desarrollo e Instalación y testeo de entorno de penetración:(pentesting environment).*

[7] andatux (2016). *Toolkits para hacking (III): Máquinas vulnerables para practicar*. https://fwhibbit.es/toolkits-para-hacking-iii-maquinas-vulnerables-para-practicar

[8] Rani, S., & Nagpal, R. (2019). *PENETRATION TESTING USING METASPLOIT FRAMEWORK: AN ETHICAL APPROACH*.

[9] Moore, M. (2017). *Penetration Testing and Metasploit*.

[10] Sheyner, O., & Wing, J. (2003, November). *Tools for generating and analyzing attack graphs. In International symposium on formal methods for components and objects* (pp. 344-371). Springer, Berlin, Heidelberg.

[11] Stolen & Erdogan (2015). http://coras.sourceforge.net/index.html

[12] Golnaz Elahi1, Eric Yu2 y Nicola Zannone (2009). *A Modeling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations*

[13] Gandarillas, Aurelio (2017) *Prototipado.* https://metodologia.es/prototipado/

[14] INTECO (2011). *Análisis de tráfico con WireShark*

[15] Herrero C., Daniel (2016). *Nessus - Escaneo de vulnerabilidades.* http://k-oox.blogspot.com/2016/05/nessus-escaneo-de-vulnerabilidades.html

[16] O'Connor, Debbie (2019). *The meaning of Shapes in Design.* https://www.whiteriverdesign.com/meaning-shapes-design/

[17] Velarde, Orana (Accedido en Septiembre de 2020). *The Meaning of Shapes and How to Use Them Creatively in Your Designs.* https://visme.co/blog/geometric-meanings/

[18] Wikipedia (2020). Estrella Roja. https://es.wikipedia.org/wiki/Estrella_roja

[19] Domínguez García, Judith (2018). *LA COMPRENSIÓN VISUAL*. https://centromiranda.es/la-comprension-visual/

# Anexo

## A.A. Funcionamiento de Metasploit

Metasploit está disponible a través de diversas interfaces y cuenta con múltiples módulos, exploits y payloads que el usuario puede utilizar para llevar a cabo pruebas de penetración.

Trabaja con los logs de algún programa de escaneo de vulnerabilidades. En este caso el programa usado para detectar vulnerabilidades es OpenVAS, que viene instalado por defecto en Kali Linux.

En cuanto a cargas maliciosas, cabe destacar el payload por excelencia que ofrece Metasploit: el intérprete de comandos Meterpreter. Es un payload multipropósito con funcionalidades de rootkit, que se instala en los sistemas explotados para interactuar remotamente sobre los mismos. Es una familia de plugins avanzados que encripta canales y permite controlar el SO (Sistema Operativo). Todo ello es cargado en la memoria del sistema sin crear ningún proceso adicional ni dejar rastros, permitiendo incluso la inyección dinámica de dlls o la migración entre procesos del intérprete.

### Módulos

Metasploit cuenta con módulos que utiliza para desempeñar distintas tareas. Son programas que permiten por ejemplo el escaneo o el ataque a una máquina. Dichos módulos son los siguientes: Payloads, Exploits, Encoders, NOPs y Auxiliary.

Los payloads son cargas maliciosas con las que sacar provecho de las vulnerabilidades. Tipos: Singles o inline (Todo en uno), Stagers (Establecimiento de conexión víctima-atacante), Stages (complemento del stager que ejecuta una tarea), meterpreter (payload potente y avanzado que se carga en memoria), passiveX (utiliza ActiveX de Internet Explorer para comunicarse por HTTP), NoNX (evita el Data Execution Prevention de Windows), etc [Ref10].

Los exploits son instrucciones que permiten explotar vulnerabilidades de un sistema y permiten acceder a él. Se pueden clasificar en dos conjuntos: exploits automatizados (la herramienta se las ingenia para entrar en el sistema por medio de un exploit y en base a la información que tiene de la víctima, como por ejemplo el sistema operativo o las vulnerabilidades de los servicios instalados) y exploits manuales (escoges un exploit en base a la información que conoces de la máquina víctima)[Ref11]. Los exploits también pueden clasificarse en otros dos conjuntos en base al momento de su ejecución: exploits activos (se envían y se ejecutan) y exploits pasivos (esperan que ocurra algo antes de ejecutarse, como por ejemplo la conexión de una víctima)[Ref12].

Los encoders son codificadores que permiten dificultar la detección del ataque. Las actividades maliciosas pueden ser descubiertas por motores antivirus o ciertos mecanismos

de seguridad. Es posible evadirlos mediante encoders. (puedes consultar los encoders disponibles mediante el comando “show encoders”) (Por ejemplo, msfvenom es una combinación de msfpayload y msfencode)

Por otro lado, tenemos los NOPs (No Operation Payloads). Es un mecanismo para conservar el funcionamiento de un payload en el que las instrucciones se mueven de posición en la memoria, que consiste en añadir operaciones NOP en los espacios que separan una posición de memoria con la que en realidad contiene la siguiente instrucción que debe ejecutarse o para reservar memoria en la que insertar código malicioso.

En cuanto a Auxiliary, es un tipo de exploit sin payload que se utiliza mayormente para funciones de escaneo, recogida de información y detección de vulnerabilidades (funciona mediante el comando ‘run’ en lugar de ‘exploit’)

### Interfaces

En lo que se refiere al modo de usar Metasploit, esta herramienta cuenta con varias interfaces que facilitan la interacción con ella dependiendo del uso que se le quiera dar y el entorno en el que se ejecute:

*Msfconsole*: es la interfaz “todo-en-uno” de línea de comandos de uso interactivo y la más popular de Metasploit. Permite acceder a todas las opciones que ofrece Metasploit de manera eficiente. Esta opción es la utilizada para este el trabajo.

*MsfGUI*: es la interfaz gráfica y el método más sencillo para usar Metasploit.

*Msfweb*: es la interfaz web útil para presentaciones o trabajos en equipo.

*Msfcli*: es la interfaz de línea de comandos no interactiva muy práctica para crear scripts y automatizaciones.

*Armitage*: es una interfaz gráfica que visualiza objetivos y ofrece recomendaciones de ataques. Mediante esta GUI se añaden elementos visuales a la información que emplea y muestra Metasploit, por lo que no añade más información sobre los ataques ni los representa, pero sí facilita su utilización. Se encuentra en el repositorio de Kali Linux (apt-get install armitage).

## A.B. Metasploitable2

Se trata de una máquina virtual Linux que intencionalmente contiene numerosas vulnerabilidades. Está pensada para ser explotada y mostrar fallos de seguridad que podrían encontrarse en sistemas que no han sido debidamente protegidos. Gracias a esto es posible hacer pruebas de penetración que muestren múltiples y variadas opciones de ataque en un entorno preparado específicamente para ello.

Ilustración 6. Inicio de Metasploitable2

Si combinamos Metasploitable2 y Kali Linux en un entorno de estudio es muy sencillo simular pruebas de penetración y aprender sobre numerosas vulnerabilidades que puede contener una máquina

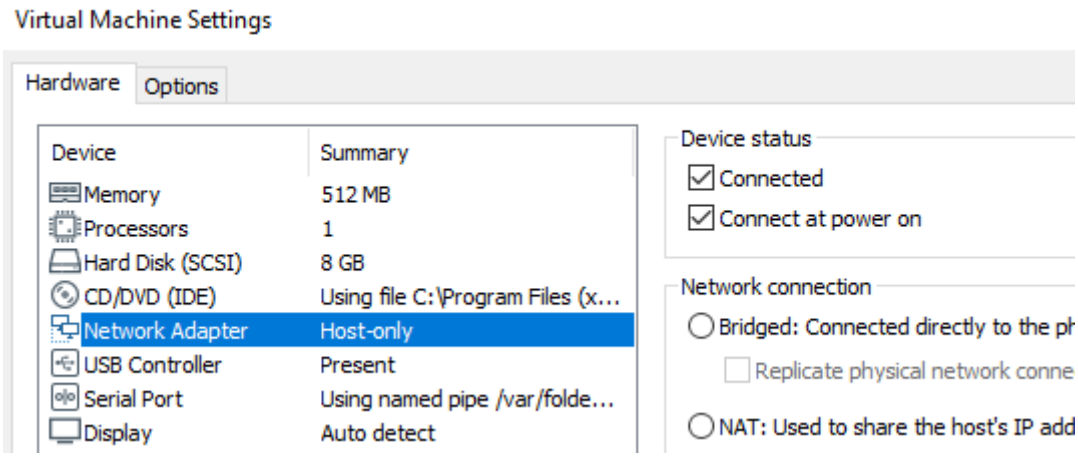
# A.C. Pasos para la instalación del entorno

## Requisitos:

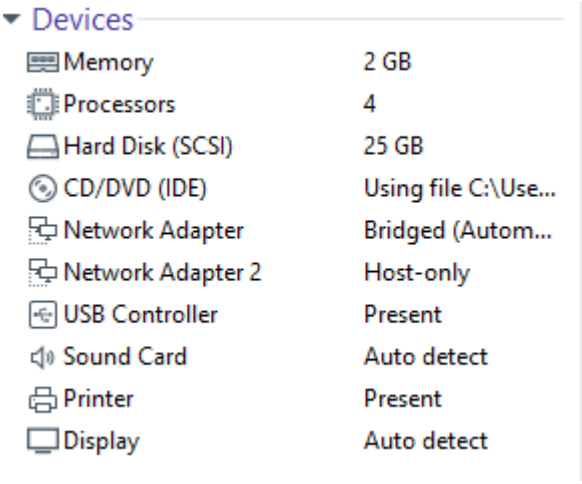
Para realizar las pruebas en un entorno seguro, usaremos los siguientes recursos:
1) Máquina virtual con Sistema Operativo Kali Linux, que cuenta con herramientas para pentesting, entre ellas Metasploit. Esta será la máquina atacante y enfocada a las funciones de seguridad.
2) Máquina virtual con vulnerabilidades: Metasploitable2. Esta será la máquina objetivo, ya que fue creada para ser destino de pruebas de seguridad y ataques, conteniendo multitud de vulnerabilidades.
3) VMware Workstation Pro para lanzar las máquinas virtuales mencionadas anteriormente.

## Configuración:

Las máquinas virtuales van a estar configuradas en el hipervisor de VMware en modo host-only. El modo host-only restringe el acceso de la máquina al exterior, de tal forma que sólo tiene visibilidad para el resto de las máquinas conectadas a la red host-only y para la propia máquina host. Este modo es el más apropiado para construir entornos de pruebas aislados:




Además del modo host-only, la máquina Kali va a contar con un segundo adaptador de red en modo Bridged, de modo que tendrá acceso al exterior y será posible descargar y actualizar herramientas:



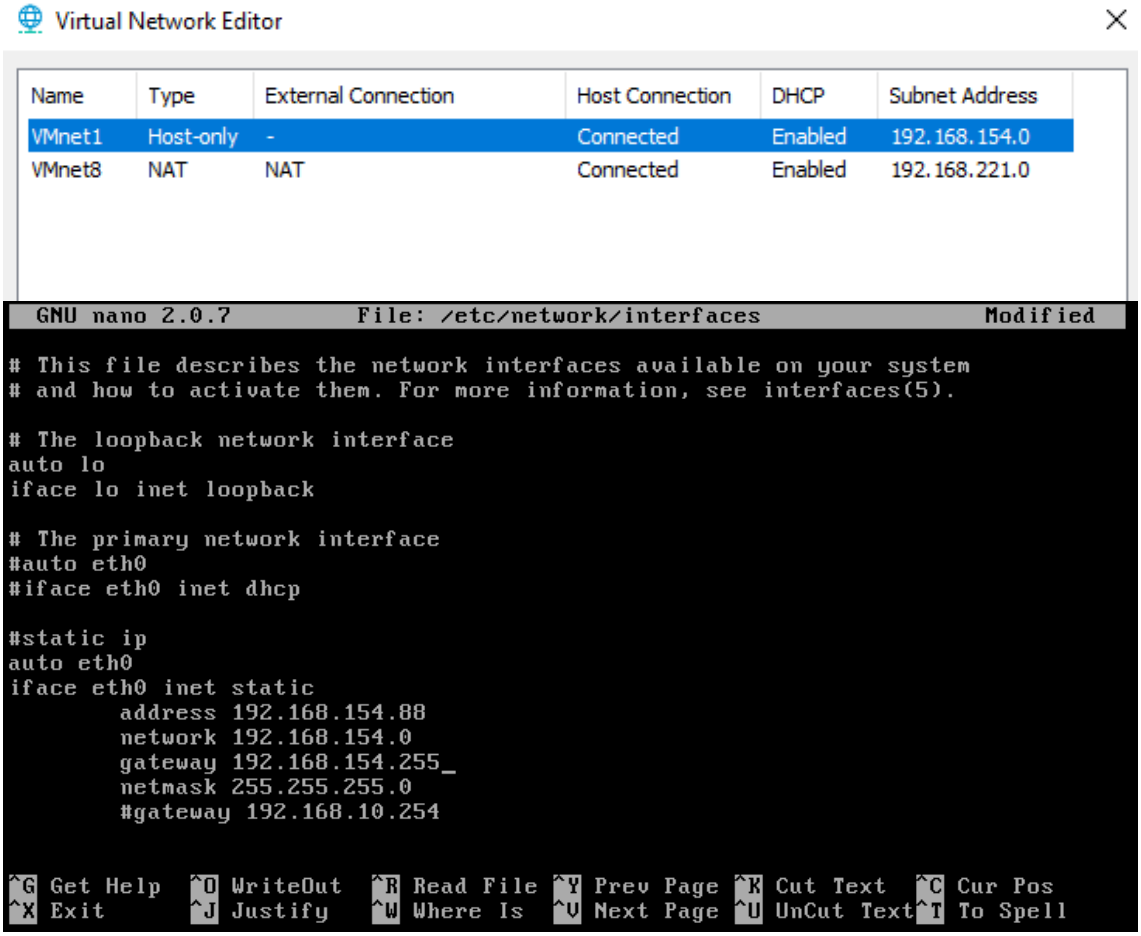

Para comprobar la visibilidad de ambas máquinas primero es necesario saber la IP que tiene cada una de ellas en la red host-only, que se puede obtener mediante el comando ifconfig:

```
root@kaliAna:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.154.128  netmask 255.255.255.0  broadcast 192.168.154.255
        inet6 fe80::d260:78d8:15cf:e6e  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:56:ef:43  txqueuelen 1000  (Ethernet)
        RX packets 13657  bytes 20111579 (19.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5745  bytes 363446 (354.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:17:96:1c
          inet addr:192.168.154.88  Bcast:192.168.154.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe17:961c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2954 (2.8 KB)
          Base address:0x2000 Memory:ef5c0000-ef5e0000
```

Al tratarse de un entorno que va a tener un uso prolongado en el tiempo, se puede establecer una IP estática, como en este caso, modificando el archivo

/etc/network/interfaces y ejecutando el comando “systemctl restart networking”. Hay que tener en cuenta que deberán coincidir los números correspondientes a la máscara de red de ambas máquinas según la dirección asignada para la red host-only, en este caso ambas tienen el formato 192.168.154.X:




A continuación, se verifica la visibilidad de ambas mediante el comando ping de envío y recepción de paquetes:

```
root@kaliAna:~# ping 192.168.154.88
PING 192.168.154.88 (192.168.154.88) 56(84) bytes of data.
64 bytes from 192.168.154.88: icmp_seq=1 ttl=64 time=0.534 ms
64 bytes from 192.168.154.88: icmp_seq=2 ttl=64 time=0.708 ms
64 bytes from 192.168.154.88: icmp_seq=3 ttl=64 time=0.592 ms
64 bytes from 192.168.154.88: icmp_seq=4 ttl=64 time=1.00 ms
^C
--- 192.168.154.88 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 82ms
rtt min/avg/max/mdev = 0.534/0.709/1.004/0.183 ms
root@kaliAna:~#
```

```
msfadmin@metasploitable:~$ ping 192.168.154.128
PING 192.168.154.128 (192.168.154.128) 56(84) bytes of data.
64 bytes from 192.168.154.128: icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from 192.168.154.128: icmp_seq=2 ttl=64 time=0.786 ms
64 bytes from 192.168.154.128: icmp_seq=3 ttl=64 time=0.263 ms
64 bytes from 192.168.154.128: icmp_seq=4 ttl=64 time=0.765 ms

--- 192.168.154.128 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.000/0.453/0.786/0.335 ms
msfadmin@metasploitable:~$
```

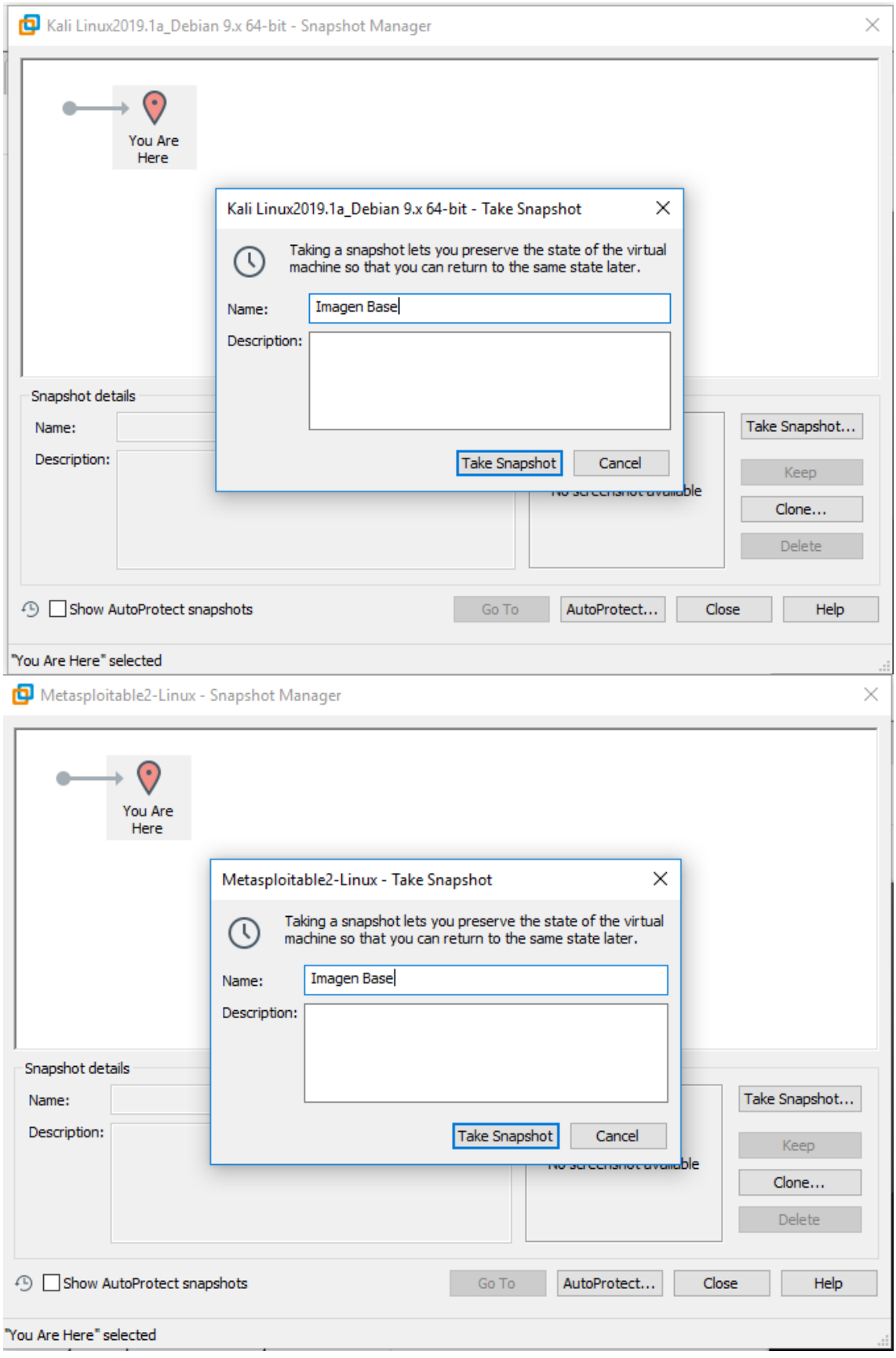
Como ambas tienen visibilidad, el siguiente paso es descargar las herramientas que necesitemos como atacantes en la máquina Kali para poder probarlas contra la máuina víctima metasploitable. En este caso, las herramientas utilizadas son Metasploit y OpenVAS. La primera viene por defecto, por lo que basta con comprobar que la versión instalada es adecuada para nuestro propósito (la utilizada en este trabajo es la 5.0.36, lanzada en Julio de 2019)

```
msf5 > version
Framework: 5.0.36-dev
Console  : 5.0.36-dev
```

OpenVAS se puede instalar mediante el comando “sudo apt-get install openvas” y se configura mediante “sudo openvas-setup”.

```
● openvas-scanner.service - Open Vulnerability Assessment System Scanne
   Loaded: loaded (/lib/systemd/system/openvas-scanner.service; disable
 preset: disabled)
   Active: active (running) since Thu 2019-07-18 00:00:12 CEST; 5s ago
     Docs: man:openvassd(8)
           http://www.openvas.org/
  Process: 6880 ExecStart=/usr/sbin/openvassd --unix-socket=/var/run/op
ock (code=exited, status=0/SUCCESS)
```

Una vez configurado el entorno con las herramientas y opciones deseadas, es recomendable guardar una captura de ambas máquinas. De esta forma, cuando ocurren cambios indeseados en alguna de ellas, como por ejemplo desconfiguraciones, es posible revertir los cambios volviendo al estado en el que estaban cuando se tomaron las capturas.


Kali Linux2019.1a_Debian 9.x 64-bit - Snapshot Manager
You Are Here
Kali Linux2019.1a_Debian 9.x 64-bit - Take Snapshot
Taking a snapshot lets you preserve the state of the virtual machine so that you can return to the same state later.
Name:
Imagen Base
Description:
Take Snapshot
Cancel
Snapshot details
Name:
Description:
Take Snapshot...
Keep
Clone...
Delete
Show AutoProtect snapshots
Go To
AutoProtect...
Close
Help
"You Are Here" selected
Metasploitable2-Linux - Snapshot Manager
You Are Here
Metasploitable2-Linux - Take Snapshot
Taking a snapshot lets you preserve the state of the virtual machine so that you can return to the same state later.
Name:
Imagen Base
Description:
Take Snapshot
Cancel
Snapshot details
Name:
Description:
Take Snapshot...
Keep
Clone...
Delete
Show AutoProtect snapshots
Go To
AutoProtect...
Close
Help
"You Are Here" selected