HACKING ÉTICO PARA PROFUNDIZAR EN LA IMPLEMENTACIÓN DE SISTEMAS INFORMÁTICOS SEGUROS.

Gabriel Villarrubia González

15 de jun. de 2021

Contenido

Tabla de figuras

PARTICIPANTES

NIF/NIE/Pasap.	Nombre y apellidos	E-mail
76125754D	Juan Francisco De Paz Santana	fcofds@usal.es
07843490F	María Navelonga Moreno	mmg@usal.es
Y56001546W	André Filipe Sales Mendes	andremendes@usal.es
08112406F	Raúl García Ovejero	raulovej@usal.es
08104619V	Juan Ramón Muñoz Rico	rico@usal.es
08104276L	Juan José Bullón Pérez	perbu@usal.es
16799405K	José Torreblanca González	torre@usal.es
12420865Z	Vivian Félix López Batista	vivian@usal.es
70900083F	Daniel Hernández de la Iglesia	danihiglesias@usal.es
16610056P	David Peral García	daveral@usal.es
X2834542E	Xuzeng Mao	xuzengmao@usal.es
70901148Z	Héctor Sánchez San Blas	hectorsanchezsanblas@usal.es
21120042Q	Francisco García Encinas	frangaren@usal.es
Y8119893X	Luis Augusto Silva	Luisaugustos@usal.es
22758868T	Diego Manuel Jiménez Bravo	dmjimenez@usal.es
78508409W	Yanira Navarro Marrero	marnayan@usal.es
70971324V	Javier Vidal Ruano	javiervidrua@usal.es
70901709T	Miguel Hernández Corral	miguelhc95@usal.es

ASIGNATURAS Y TITULACIONES QUE SE BENEFICIAN DEL PROYECTO DE INNOVACIÓN

Este proyecto es válido para todas las asignaturas de la Universidad donde se imparte tanto docencia práctica como teórica relativa a asignaturas relacionadas con sistemas informáticos. Por economizar el espacio, únicamente se han enumerado las asignaturas del grado en Ingeniería Informática y grado en Ingeniería Informática en Sistemas de Información. Además, nuestro proyecto puede impartirse en auditorios, congresos ponencias y eventos de la universidad relativos al contenido impartido.

ASIGNATURAS Y TITULACIONES QUE SE BENEFICIARÁN DEL PROYECTO DE INNOVACIÓN
Computadores I *Grado Informática*
Computadores II *Grado Informática*
Sistemas operativos I *Grado Informática / Ingeniería en sistemas de información*
Sistemas operativos II *Grado Informática / Ingeniería en sistemas de información*
Administración de sistemas *Grado Informática*
Redes de computadores I *Grado Informática*
Redes de computadores II *Grado Informática*
Seguridad de sistemas informáticos *Grado Informática*
Teoría de la información y teoría de códigos *Grado Informática*
Sistemas Informáticos *Ingeniería en sistemas de información*
Criptografía *Ingeniería en sistemas de información*
Administración de sistemas de información *Ingeniería en sistemas de información*
Seguridad informática *Ingeniería en sistemas de información*

MOTIVACIÓN DE ESTE TRABAJO

Hacking ético es una forma de referirse al acto de una persona, mejor conocido como hacker, que utiliza sus conocimientos de informática y seguridad para encontrar **vulnerabilidades o fallas de seguridad en el sistema** con el objetivo normalmente de **reportarlas en la organización** para que se tomen todas las medidas necesarias que posibiliten prevenir una catástrofe cibernética, como el robo de información.

Los profesionales que se dedican al **Hacking ético** practican una serie de pruebas o test denominados "**Test de penetración**" cuyo objetivo es poder **burlar las diferentes fallas de seguridad** que tiene la red para diferentes organizaciones, con la única intención de probar su efectividad o, por el contrario, demostrar la vulnerabilidad de aquel sistema.

Si encuentran alguna falla del sistema de seguridad, el hacker ético reporta la situación a través de un completo informe a la empresa y se procede a mejorar la seguridad contratando los servicios de este profesional de la ciberseguridad.

La aplicación que se ha desarrollado en este proyecto **puede resultar ampliamente beneficiosa en el contexto universitario**. Es de vital importancia incorporar a las tradicionales clases magistrales una **mínima parte lúdica**, hackear aprendiendo, crear un nuevo universo "colectivo" que permite olvidar las reglas y **limitaciones individuales**, los prejuicios y las dudas, potenciando el completo desarrollo de las propias capacidades y fijando de mejor forma los conocimientos. Los expertos señalan que la única solución viable para cubrir estas necesidades pasa por asociarla a la **educación digital**. Es por ello que se debe hacer foco en la importancia de dotar de capacidades tecnológicas a los alumnos de informática, ya que los puestos relacionados con la ciberseguridad son los más demandados. Además, es muy necesario formar a los alumnos a desarrollar aplicativos libre de fallos y seguros ya que en la carrera no existe una asignatura que vea estos conceptos en profundidad. Ante esta previsión, se están implementando y buscando soluciones basadas en las posibilidades que ofrece la tecnología y que, según fuentes de Telefónica Educación Digital, la división especializada en soluciones de educación de Telefónica, permiten desarrollar programas de transformación tecno-pedagógicas y avanzar hacia nuevas modalidades de capacitación.

Ilustración 1 Sistemas informáticos en educación y tipos de malware.

Con el desarrollo de esta herramienta novedosa, alumnos y profesores pueden desarrollar contenidos académicos con el objetivo de detectar vulnerabilidades de sistemas informáticos, todo ello desde una novedosa plataforma puesta en marcha. Los alumnos pueden, de esta forma, complementar la información transmitida el profesor, en tiempo real, mediante una serie de elementos diseñados exclusivamente ofreciendo un aprendizaje más inmersivo y de una forma más gráfica. La aplicación pone a disposición del estudiante una serie de sistemas

informáticos que presentan ciertas vulnerabilidades en los ámbitos de criptografía, escalada de privilegios o informática forense. Para motivar al estudiante, en cada uno de los sistemas informáticos se ha implementado la metodología "Capture the flag". Esta metodología consiste en alojar un clave o contraseña dentro del sistema, ya sea en un fichero de texto o en un registro de base de datos, al cual, en un principio, el usuario normal no tiene acceso. Para la obtención de esta clave, el estudiante tiene que aplicar diferentes técnicas de hacking que le permiten profundizar en los contenidos de administración y seguridad de sistemas informáticos.

De esta forma, se ha creado un espacio virtual mediante la utilización de servidores virtuales tanto UNIX como Windows con diferentes servicios preinstalados que presentan vulnerabilidades explotables de modo que permiten al estudiante aumentar el contenido académico relacionado con los conocimientos que son impartidos en las distintas ramas de las asignaturas de sistemas informáticos de la Universidad de Salamanca.

TAREAS Y TEMPORIZACIÓN DE LAS TAREAS REALIZADAS

#	OBJETIVO	DESCRIPCIÓN
1	Especificaciones y definición de la herramienta.	***Descripción:*** Concepción y elaboración del diseño del sistema informático. ***Resultado:*** Diseño y elaboración de los objetivos y características vulnerables del sistema.
1.1	Especificación de requisitos funcionales.	***Descripción*:** Elicitación de los contenidos funcionales que la herramienta debe cumplir en el proceso de desarrollo. ***Resultado*:** Conjunto de requisitos funcionales.
1.2	Especificación de requisitos no funcionales y de interoperabilidad en una clase.	***Descripción*:** Análisis de los requisitos necesarios para el correcto desarrollo de la herramienta y su incorporación en una clase. ***Resultado*:** Conjunto de requisitos no funcionales y medidas para la correcta incorporación del sistema en cualquier clase.
2	Investigación de técnicas.	***Descripción*:** Investigación de técnicas para la elaboración de la herramienta. ***Resultado*:** Técnicas apropiadas la elaboración de la herramienta.
2.1	Investigación de técnicas de administración de sistemas.	***Descripción*:** Análisis de las técnicas existentes para la elaboración sistemas informáticos. ***Resultado*:** Estado del arte de técnicas de administración de sistemas.
2.2	Investigación y estudio de técnicas de seguridad informática.	***Descripción:*** Análisis de las técnicas existentes para la implantación de tecnología de seguridad informática. ***Resultado:*** Estado del arte de técnicas de seguridad informática.

2.3	Investigación y estudio acerca de técnicas de búsqueda de vulnerabilidades.	***Descripción:*** Análisis de diferente documentación para búsqueda de vulnerabilidades. ***Resultado:*** Elección del framework y tecnologías de implementación.
3	Elaboración de la herramienta.	***Descripción:*** Desarrollo del software para el funcionamiento del sistema. ***Resultado:*** Software necesario para la integración de los diferentes componentes del sistema.
3.1	Desarrollar el sistema informático Unix o Windows que contenga diferentes vulnerabilidades y la clave a obtener.	***Descripción:*** Programación de servidores Unix o Windows con diferentes vulnerabilidades. ***Resultado:*** Servidor Unix o Windows son vulnerabilidades.
3.2	Creación de diferentes servidores virtuales para la simulación de diversos escenarios.	***Descripción:*** Diseño y modelado de diferentes servidores virtuales. ***Resultado:*** Conjunto de servidores virtuales que pueden ser usadas para el simulado de las clases y prueba de las diferentes vulnerabilidades del sistema.
4	Instalación y entono de pruebas.	***Descripción:*** Instalación de sistemas alojados en servidores remotos con vulnerabilidades. ***Resultado:*** Versión alfa del sistema, para el testeo y realización de pruebas y retos.
4.1	Generación de contenido de ejemplo para clases de sistemas informáticos.	***Descripción:*** Diseño del contenido multimedia y manuales para formar a los profesores. ***Resultado:*** Contenido de pruebas.
4.2	Instalación de la aplicación en servidores remotos para las pruebas.	***Descripción*:** Instalación de la aplicación en servidores remotos. ***Resultado*:** Servidores con la aplicación preparada para su uso en una clase virtual.
4.3	Detección de posibles errores.	***Descripción:*** Ensayo del funcionamiento de la aplicación, para hallar posibles defectos. ***Resultado:*** Estado de la aplicación para realizar posibles correcciones de errores.
4.4	Implementación de posibles mejoras en la usabilidad de la aplicación.	***Descripción*:** Percepción de posibles mejoras que aumenten las características del sistema. ***Resultado*:** Incremento de la funcionalidad del sistema.
5	Aplicación de la herramienta en escenario real	***Descripción:*** Comenzar pruebas en cursos reales en las titulaciones de ingeniería. ***Resultado:*** Resultados general y amplio es curso académico.

5.1	Evaluación de la herramienta en un caso de estudio real.	***Descripción:*** Evaluar el rendimiento en las asignaturas y de la herramienta. ***Resultado:*** Obtención de resultados en asignaturas de plan académico.
5.2	Medición del impacto mediante indicadores de eficiencia.	***Descripción*:** Medir si la herramienta presenta una eficiencia en el rendimiento de los alumnos. ***Resultado*:** Obtención de resultados académico gracias a la implantación del proyecto.
5.3	Obtención de resultados obtenidos mediante el empleo de la aplicación.	***Descripción*:** Generar resultados de compromiso de los estudiantes con los planes de estudio para el análisis de resultados académicos en entregas. ***Resultado*:** Documentación de la evolución de la mejora en las técnicas de estudio aplicadas por los alumnos.
6	Difusión de los resultados.	***Descripción*:** Difundir los resultados obtenidos gracias a la implantación de la herramienta en el campus. ***Resultado*:** Difusión general y científica del empleo del sistema.
6.1	Recogida y evaluación de la implantación del sistema.	***Descripción*:** Evaluación de todos los resultados obtenidos mediante el desarrollo e implantación del sistema. ***Resultado*:** Documentar todos los resultados obtenidos a nivel de usabilidad, desarrollo software, empleo de los alumnos y beneficios académicos.
6.2	Publicación de los resultados obtenidos en revistas científicas.	***Descripción*:** Exposición de los resultados obtenidos en Congreso y revistas del proyecto para la difusión de los beneficios de este tipo de herramientas. ***Resultado*:** Publicación de artículo en congreso internacional.

	NOV	DIC	ENE	FEB	MAR	ABR	MAY
1. Especificaciones y definición de la herramienta.							
1.1 Especificación de requisitos funcionales.							
1.2 Especificación de requisitos no funcionales y de interoperabilidad en una clase.							
2 Investigación de técnicas.							
2.1 Investigación de técnicas de administración de sistemas.							
2.2 Investigación y estudio de técnicas de seguridad informática.							
2.3 Investigación y estudio acerca de técnicas de búsqueda de vulnerabilidades.							
3 Elaboración de la herramienta.							
3.1 Desarrollar el sistema informático Unix o Windows que contenga diferentes vulnerabilidades y la clave a obtener.							
3.2 Creación de diferentes servidores virtuales para la simulación de diversos escenarios.							
4. Instalación y entono de pruebas.							
4.1 Generación de contenido de ejemplo para clases de sistemas informáticos.							
4.2 Instalación de la aplicación en servidores remotos para las pruebas.							
4.3 Detección de posibles errores.							
4.4 Implementación de posibles mejoras en la usabilidad de la aplicación.							
5 Aplicación de la herramienta en escenario real.							
5.1 Evaluación de la herramienta en un caso de estudio real.							
5.2 Medición del impacto mediante indicadores de eficiencia.							
5.3 Obtención de resultados obtenidos mediante el empleo de la aplicación.							
6. Difusión de los resultados.							
6.1 Recogida y evaluación de la implantación del sistema.							
6.2 Publicación de los resultados obtenidos en revistas científicas.							

PUNTO DE PARTIDA, ENCUESTA A ESTUDIANTES

Para justificar la necesidad de implantación de una herramienta de que ayude a profundizar en la implementación de sistemas informáticos seguros es de vital importancia conocer la opinión de profesores y alumnos acerca de los métodos de enseñanza tradicionales que son llevados a cabo en el centro. Es por ello por lo que durante la realización de este proyecto se ha realizado una encuesta anónima y con carácter voluntario a personal docente y alumnos de las carreras de ingeniería con un total de **38** participantes. A continuación, se listan las preguntas que se hicieron y los resultados para cada una de ellas.

- ¿Sabes que es el hacking ético? *Respuesta SI/NO*
- ¿Crees que puedes adquirir conocimientos con métodos o técnicas de hacking ético? *Respuesta SI/NO*
- ¿Conoces algún profesor que utilice algún sistema basado en hacking ético para impartir algún contenido académico en la Facultad de Ciencias? *Respuesta SI/NO*
- ¿Crees que dispones de las capacidades o técnicas para empezar a explotar vulnerabilidades de sistemas? *Respuesta SI/NO*

- ¿Conoces alguna herramienta que te permita profundizar sobre contenidos de vulnerabilidades en los sistemas que haga uso de hacking ético? ***Respuesta SI/NO***
- ¿Has probado alguna vez algún sistema que te permita realizar tareas de hacking ético? ***Respuesta SI/NO***
- ¿Crees que las plataformas de hacking online son seguras para empezar a experimentar con estas vulnerabilidades? ***Respuesta SI/NO***
- ¿Crees que una herramienta de hacking ético podría aumentar tus capacidades a la hora crear nuevos sistemas informáticos seguros? ***Respuesta SI/NO***
- ¿Te parecen suficientes los esfuerzos que realizan los profesores para la incorporación de nuevas tecnologías en clase? ***Respuesta SI/NO***
- Del 1 (desfavorable) al 5 (muy favorable), indica el grado de satisfacción con los métodos de aprendizaje que utilizan tus profesores en el día a día.

A continuación, se muestran los resultados finales de la encuesta:

- ¿Sabes que es el hacking ético? ***Respuesta SI/NO***

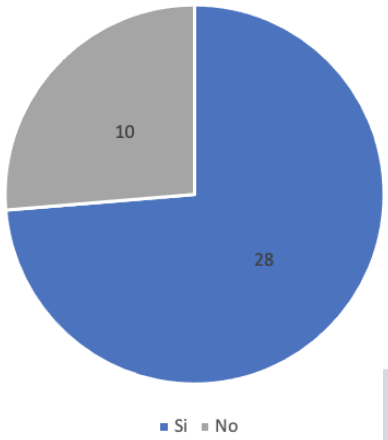

Ilustración 2 : Estadística ¿Sabes que el hacking ético?

- ¿Crees que puedes adquirir conocimientos con métodos o técnicas de hacking ético? ***Respuesta SI/NO***

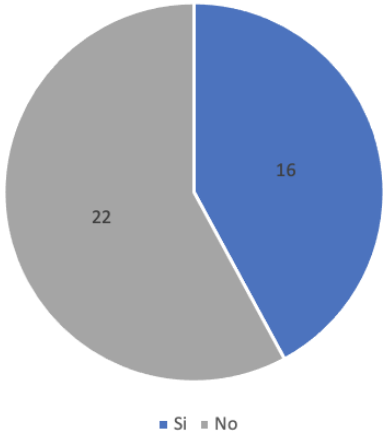

Ilustración 3 : Estadística ¿Crees que puedes adquirir conocimientos con métodos o técnicas de hacking ético?

- ¿Conoces algún profesor que utilice algún sistema basado en hacking ético para impartir algún contenido académico en la Facultad de Ciencias? ***Respuesta SI/NO***

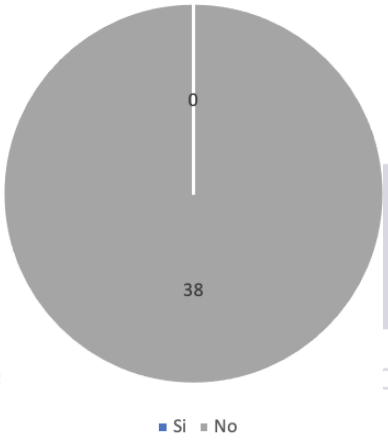

Ilustración 4: Estadística ¿Conoces algún profesor que utilice algún sistema basado en hacking ético para impartir algún contenido académico en la Facultad de Ciencias?

- ¿Crees que dispones de las capacidades o técnicas para empezar a explotar vulnerabilidades de sistemas? ***Respuesta SI/NO***

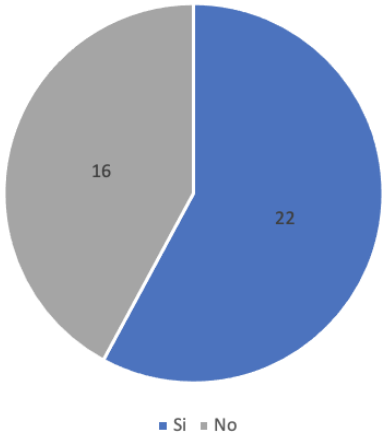

Ilustración 5: Estadística ¿Crees que dispones de las capacidades o técnicas para empezar a explotar vulnerabilidades de sistemas?

- ¿Conoces alguna herramienta que te permita profundizar sobre contenidos de vulnerabilidades en los sistemas que haga uso de hacking ético? ***Respuesta SI/NO***

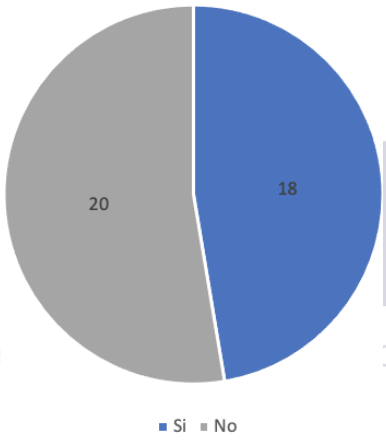

Ilustración 6: Estadística ¿Conoces alguna herramienta que te permita profundizar sobre contenidos de vulnerabilidades en los sistemas que haga uso de hacking ético?

- ¿Has probado alguna vez algún sistema que te permita realizar tareas de hacking ético? ***Respuesta SI/NO***

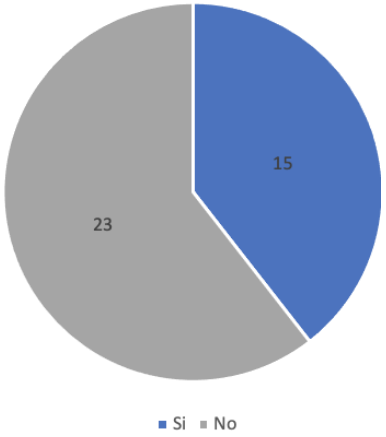

Ilustración 7: Estadística ¿Has probado alguna vez algún sistema que te permita realizar tareas de hacking ético?

- ¿Crees que las plataformas de hacking online son seguras para empezar a experimentar con estas vulnerabilidades? ***Respuesta SI/NO***

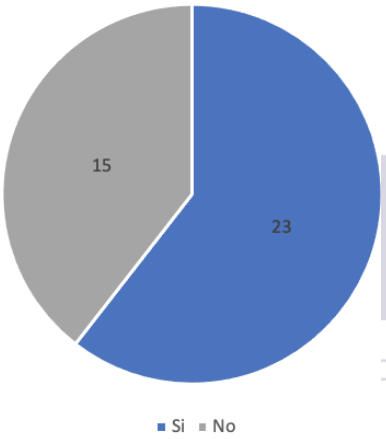

Ilustración 8: ¿Crees que las plataformas de hacking online son seguras para empezar a experimentar con estas vulnerabilidades?

- ¿Crees que una herramienta de hacking ético podría aumentar tus capacidades a la hora crear nuevos sistemas informáticos seguros? ***Respuesta SI/NO***

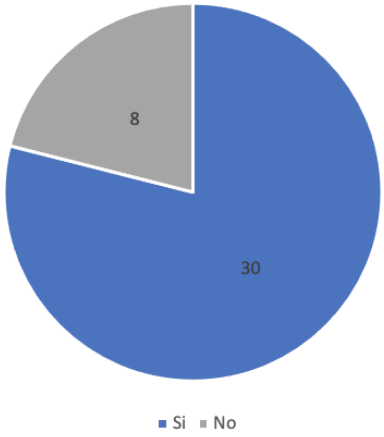

Ilustración 9: Estadística ¿Crees que una herramienta de hacking ético podría aumentar tus capacidades a la hora crear nuevos sistemas informáticos seguros?

- ¿Te parecen suficientes los esfuerzos que realizan los profesores para la incorporación de nuevas tecnologías en clase? ***Respuesta SI/NO***

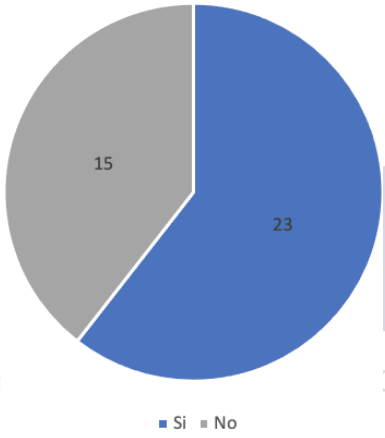

Ilustración 10: ¿Te parecen suficientes los esfuerzos que realizan los profesores para la incorporación de nuevas tecnologías en clase?

- Del 1 (desfavorable) al 5 (muy favorable), indica el grado de satisfacción con los métodos de aprendizaje que utilizan tus profesores en el día a día.

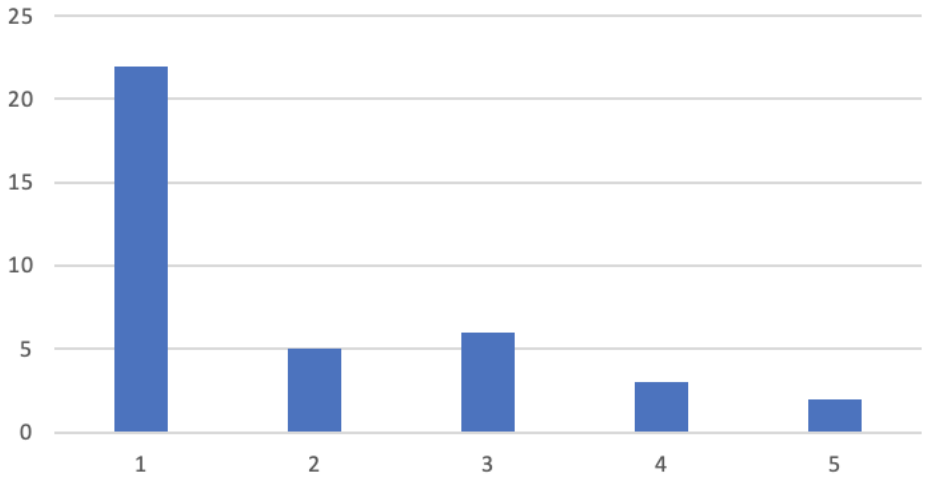

Ilustración 11: Estadística indica el grado de satisfacción con los métodos de aprendizaje que utilizan tus profesores en el día a día.

Observando los resultados de los encuestados, se puede decir que el 73% de los encuestados saben lo que es el hacking ético, pero apenas el 42% cree que puede adquirir conocimiento con métodos de hacking ético. Se puede deducir de las repuestas que la mitad de los encuestados conoce alguna herramienta que permita poner en práctica estas técnicas la mayoría de ellos han usado alguna de estas herramientas.
Aunque el 72% de los encuestados creen que el uso de herramienta de hacking ético podría aumentar sus capacidades a la hora de crear sistemas si antes han realizado pruebas y estudiar las vulnerabilidades haciendo uso de sistemas de este tipo, ellos no conocen a ningún profesor que en su centro educativo utilice algún método de hacking ético para la enseñanza de contenidos o la profundización de contenidos.

TRABAJOS RELACIONADOS

La generación de escenarios de seguridad ya ha sido estudiada por diferentes autores. A continuación, se puede ver un articulo científico donde se crea un framework que tiene la capacidad de la generar este tipo de escenarios de forma aleatoria para el aprendizaje de este tipo de contenidos.

Z. Cliffe Schreuders, Thomas Shaw, Mohammad Shan-A-Khuda, Gajendra Ravichandran, and Jason Keighley, Leeds Beckett University; Mihai Ordean "Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events", ASE'17, https://www.usenix.org/conference/ase17/workshop-program/presentation/schreuders

Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events

Z. Cliffe Schreuders, Thomas Shaw, Mohammad Shan-A-Khuda, Gajendra Ravichandran, and Jason Keighley, *Leeds Beckett University*
Mihai Ordean, *University of Birmingham*

Abstract

Computer security students benefit from hands-on experience applying security tools and techniques to attack and defend vulnerable systems. Virtual machines (VMs) provide an effective way of sharing targets for hacking. However, developing these hacking challenges is time consuming, and once created, essentially static. That is, once the challenge has been "solved" there is no remaining challenge for the student, and if the challenge is created for a competition or assessment, the challenge cannot be reused without risking plagiarism, and collusion.

Security Scenario Generator (SecGen) can build complex VMs based on randomised scenarios, with a number of diverse use-cases, including: building networks of VMs with randomised services and in-the-wild vulnerabilities and with themed content, which can form the basis of penetration testing activities; VMs for educational lab use; and VMs with randomised CTF challenges. SecGen has a modular architecture which can dynamically generate challenges by nesting modules, and a hints generation system, which is designed to provide scaffolding for novice security students to make progress on complex challenges. SecGen has been used for teaching at universities, and hosting a recent UK-wide CTF event.

1. Introduction

Computer security students benefit from hands-on experience applying security tools and techniques to attack and defend vulnerable systems. Practical lab work and pre-configured hacking challenges are common practice both in security education and also as a pastime for security-minded individuals. Competitive hacking challenges, such as Capture the Flag (CTF) competitions have become a mainstay at industry conferences and are the focus of large online communities. CTF activities have been used in education as an effective way of providing and assessing engaging hands-on security challenges, and is often the focus of student hacking society activity (see e.g. [1]–[3]). Virtual machines (VMs) provide an effective way of sharing targets for hacking, and can be designed in order to test the skills of the attacker. Websites such as Vulnhub [4] host pre-configured hacking challenge VMs and are a valuable resource for those learning and advancing their skills in computer security. However, developing these hacking challenges is time consuming, and once created, essentially static. That is, once the challenge has been "solved" there is no remaining challenge for the student, and if the challenge is created for a competition or assessment, the challenge cannot be reused without risking plagiarism, and collusion.

Delivering hacking scenarios to students involves a number of existing challenges, which we aim to overcome: existing pre-configured hacking challenges (such as Metasploitable and those on VulnHub) are typically static and therefore they suffer from limited re-play and reuse, since they only need to be solved once before a solution/write-up is available; and, as a consequence, academic or competitive assessment via pre-developed scenarios is fraught with the risk of hard to detect or prevent plagiarism and collusion.

The typical attempted solution to these issues is the time-consuming process of manually configuring hacking scenarios as vulnerable learning scenarios are required, typically on an event-by-event basis, accepting that each student has the same challenge and the same CTF flags to find.

This is not practical at scale: the network infrastructure and staff costs of running a single two day event is large (see e.g. [5]) and it can be argued that providing a whole cohort of students with appropriate and randomised assessment tasks, across a 12 week course is not practical using traditional methods.

Recently, there has been some related work to randomise security challenges or flags (such as [2], [6], [7]); however, these approaches are focussed on adding randomness to specific challenges or generating random flags that are inserted into static challenges.

We have created Security Scenario Generator (SecGen)[1] which provides a robust framework that can build complex VMs based on randomised scenarios, with a number of diverse use-cases, including: building networks of complex VMs with randomised services and in-the-wild vulnerabilities and with themed content such as business names, employees, and so on, which can form the basis of penetration testing activities; VMs for educational lab use; and VMs with randomised CTF challenges, with randomised (yet meaningful) challenges including real-word vulnerabilities. SecGen has a number of unique features, including a modular architecture which can dynamically generate challenges by nesting modules, and a hints generation system, which is designed to provide scaffolding for novice security students to make progress on complex challenges.

In this paper we describe our aims, present the SecGen framework, including its architecture, configuration language, and use cases, and present evaluation based on using the system for teaching at universities, and hosting a recent UK-wide CTF event.

2. Related Literature

Capture The Flag (CTF) competitions have been popular in the computer security community since the 90s, including the first DEFCON CTF [8]. Other popular annual CTF competitions include those that target university students, such as CSAW CTF [9], [10] and RuCTF [11], those that target high schools, such as PicoCTF [7], [12], others include Ghost in the Shellcode [13], Codegate [14], and UCSB iCTF [15]. The website ctftime.org [16] tracks these CTF events and many more, and lists thousands of teams that take part in competitions on an almost weekly basis. Many CTF events are conducted entirely online, such as DEFCON CTF Qualifiers, and online CTFs often feature a write-up submission, while others are conducted in-person, such as DEFCON CTF, and typically include a live leaderboard.

The most common style of CTF is based on *jeopardy challenges,* where competitors are typically presented with a board of independant challenges, typically with downloads of files for each challenge. Other styles of CTF include attack-defence, where the focus is on attacking or defending systems from attack while keeping services available [17]. In some cases, such as CCDC [18], competing student teams focus entirely on defence, while in other cases, such as RuCTFE [11], teams both patch and defend their systems while attacking others. Attack-defence CTFs often distribute vulnerable systems in the form of VMs. Various forms of games-based learning and gamification (such as leveling-up and leaderboards) have been applied to security education [19], [20]. Gondree *et. al* [21] emphasise diversity of the variety of approaches taken and describe security games as being on a continuum based on task variety and adversarial dynamicity (such as whether teams interact with each other); while acknowledging that this is an over-simplification and that many other game attributes are important.

Capture The Flag (CTF) competitions are a popular means of engaging students with cyber security. The pedagogical benefits of CTF competitions have been widely reported. Efforts to incorporate CTF in higher education (HE) include engaging students in out-of-class CTF activity to cultivate "informal learning spaces" [1], [22], delivering the lab work exercises in the form of CTF-style challenges [2], where flags are revealed where tasks are completed or challenges are solved, and Class Capture-the-Flag Exercises (CCTFs) [23], where teams play-off against each other in regular in-class competitions.

Challenges in running CTF events include the effort required to design and test challenges for quality and appropriate difficulty level, especially where the aim is to ensure accessibility for beginners [9]. The effort required to create challenges and attack scenarios (whether CTF-style or other vulnerable scenarios such as Metasploitable or VMs posted to Vulnhub) is substantial, and time consuming, and as stated earlier, essentially static, making reuse problematic.

Various frameworks for hosting CTFs have been published, such as Facebook CTF (FBCTF) [24], CTFd [25], HackTheArch [26], Mellivora [27], NightShade [28], and picoCTF-Platform 2 [29]. These frameworks typically present jeopardy challenges and scoreboards, and provide administrators a web interface for managing challenges. The iCTF Framework [15] can be used to host attack-defence CTFs, and can generate VirtualBox VMs for each team using setup scripts and vulnerable services that are manually created for each event. The InCTF Framework [17] builds on iCTF Framework to deploy CTFs and teams' exploits on Docker containers. While these various frameworks lower the barrier for hosting CTF events, the challenges are typically static, and as a result challenges are often not publicly published, and as such each new CTF event involves manually creating new challenges. Furthermore, most frameworks are geared towards jeopardy-style CTFs with either one or no hints, and no existing framework mets our aims, as discussed in the following section.

CyTrONE is a framework that aims to automate environment setup tasks for security education [30]. CyTrONE has a management UI, integrates with LVEs, and YAML specifications state software to install and run, and can include questions and answers to present to

1 SecGen is free and open source software (FOSS) available at http://github.com/cliffe/SecGen

users.

Previous work to provide randomisation to security challenges includes PicoCTF-Platform 2, which includes automatic problem generation (APG), where permutations of challenges can be generated on a per-team basis (or allocated from polls of instances for improved scalability), and served to teams via the web interface [7]. This approach could be used to generate dynamic challenge content (as APG been applied in other disciplines [31]); however, PicoCTF 2014 solely used APG to detect and prevent cheating by generating different flags per-team [7]. Attempts to share flags were detected 1081 times (0.84%). Chothia et. al [2] devised a CTF system to prevent flag sharing by automatically generating separate flags (based on public key cryptography) per-student in a single VM that is distributed to students. Feng developed MetaCTF [6], which provides polymorphic and metamorphic reverse engineering challenges so that students are given unique challenges, and which is designed around jeopardy-style CTF challenges for a HE curriculum, with a scaffolded progression of exercises. Other randomisation of reverse engineering security challenges includes Tigress, which provides dynamic obfuscation of C code [32].

The authors are not aware of any other projects that provide randomisation at the system/VM level for generating randomly vulnerable systems.

3. Aims and Methods

3.1. Overall Aim

The overall aim for for this work was to provide a randomizable, flexible, and general purpose method for specifying and generating VMs for security education and training purposes.

3.2. Use Cases

The educational use-cases include:

- simulations of organisations with a mix of secure and insecure services; with desktop and servers; for simulated security audits;
- security lab exercises; and,
- challenges for CTF events or CTF-style lab work.

3.3. Rich-Scenarios

To achieve these ambitious use-cases, rather than focus on generating standalone *jeopardy challenges* in the form of individual files, the aim of this work is to output a set of VMs, representing *rich-scenarios*. Each rich-scenario can include:

- One or more systems (VMs)
- Complete operating systems, including server and desktop systems
- Networked configuration, including multiple network segments
- Network services (such as FTP, IRC, HTTP, NFS)
- System configuration (such as users and accounts, and software installed)
- Files representing thematic content, such as themed websites
- Software vulnerabilities (including in-the-wild software vulnerabilities, and randomly generated vulnerabilities in protocols or software/websites)
- Configuration vulnerabilities (including misconfigured access controls and services, weak passwords, and so on)
- Data interpretation challenges including steganography, encryption and encoding
- "Loot", such as flags or simulated sensitive data
- CTF-style challenges (where solving challenges or compromising vulnerabilities such as any of the above leads leads directly to the discovery of flags)

3.4. Randomisation

Randomisation and modular reuse of the above elements is a primary goal. Our aim is to randomise the following:

- *Selection*: randomised selection of the above elements. For example, randomly choosing operating system(s), network configuration(s), service(s), system configuration(s), including user accounts and passwords, with random selection of in-the-wild vulnerabilities or security challenges.
- *Parameterisation*: all of the elements should be able to be configured (for example, the ports services should use, strength of passwords, theme of the scenario), and this configuration will be randomizable.
- *Nesting*: data generation (such as the generation of random flags) and interpretation challenges (such as encoding) should be able to be combined/nested in randomised ways. For example, a flag can be randomly generated, and then encoded in some random way before being leaked via a random software vulnerability.

3.5. Specification Language and Constrained Randomisation

A further aim is to design and implement a scenario specification language, that will randomly generate rich-scenarios for these use cases. Given the significant diversity in potential randomisation implied by our

randomisation aims, it is important that the specification language can specify the inclusion of elements and constrain randomisation to meaningful and context appropriate selection, parameterisation, and nesting.

The specification language will be capable of representing the generation of unique security scenarios based on a configurable set of optional constraints: for example, a network of servers, with specific kinds of services (such as a Web server and a file server) with specific kinds of software or misconfiguration vulnerabilities (such as remote code execution and local privilege escalation vulnerabilities). Vulnerabilities and services will be randomly selected and installed on VMs, as specified.

3.6. Student Engagement

The project aimed to engage students, both in development, and in using and evaluating the VMs and learning environments that were generated. We aimed to use our framework to provide rich-scenarios for penetration testing exercises, and to introduce new university student hacking teams to CTFs and as a stepping-stone to taking part in international competitions.

3.7. Development Methodology

Software design and development was led by the primary-author, with a cross-institutional software development team that over time included 10 undergraduate students (6 employed, others working on sub-projects), 1 postgraduate MRes student, and 1 postdoctoral researcher. Additionally, a team of students were employed to develop a range of CTF-style challenges, and adapt CTF challenges from existing security labs (such as [2]).

The software was developed open source using a relaxed Scrum methodology, with a backlog, regular sprint meetings, and task assignment. The current version was always available via Github, and typically members of the team tested each other's code before committing to the master branch.

This approach was designed to engage our students in developing their skills beyond their taught courses, giving them experience in software development, and developing learning materials.

4. Security Scenario Generator (SecGen)

4.1. Introducing SecGen

Here we present Security Scenario Generator (SecGen), which is designed to achieve all of the aims described in Section 3.

SecGen is a Ruby application, with an XML configuration language. SecGen reads its configuration, including the available vulnerabilities, services, networks, users, and content, reads the definition of the requested scenario, applies logic for randomising the scenario, and leverages Puppet and Vagrant to provision the required VMs.

SecGen generates randomised vulnerable VMs that are created based on a scenario specification, which describes the constraints and properties of the VMs to be created. For example, a scenario could specify the creation of a system with a remotely exploitable vulnerability that would result in user-level compromise, and a locally exploitable flaw that would result in root-level compromise. This would require the attacker to discover and exploit both randomly selected vulnerabilities in order to obtain root access to the system. Alternatively, the scenario that is defined can be more specific, specifying certain kinds of services (such as FTP or SMB) or even exact vulnerabilities (by CVE).

This work builds on an early prototype implementation that demonstrated the feasibility of the combination of technologies [33]. The system was re-architected and advanced features were implemented to achieve our ambitious set of aims, and which are described in the following sections.

4.2. Architecture and Modularity

SecGen leverages a number of virtualisation and automation technologies, including Vagrant and Puppet. Vagrant, which is typically used by developers to manage development environments [34], is used to provision VMs, Puppet, which is typically used to manage large scale deployments of servers [35], is used to configure the VMs, and Librarian-puppet is used to manage the deployment of the selected puppet modules. The final output currently includes VirtualBox VMs.

SecGen is designed to be highly modular, with a directory structure and general design philosophy loosely inspired by Metasploit's modular structure. For example, the `modules/vulnerabilities/` directory includes modules representing various vulnerabilities, sometimes directly relating to Metasploit Framework's corresponding `modules/exploits/` modules.

The underlying structure of SecGen is that of a number of "system" objects, which represent VMs (with a Vagrant basebox that is selected based on specified attributes), and each is associated with a list of SecGen "module" objects which are primarily selected based on specified attributes.

Each module has a type (such as vulnerability, service, utility, generator, or encoder), module path, and a associative array of attributes (such as CVE number, difficulty level, CVSS, and so on). Modules can receive data into named parameters (such as port_number or strings_to_leak), either from the output from another module or from data stored in a datastore (variable). Modules can output data, which can be directed at the

input of another module's parameters or into a datastore. Modules can include Puppet code which is deployed to and executed on the VMs (as in the case for vulnerability, service, and utility modules), or local code which provides randomisation or transformation of data (as with encoder and generator modules). Furthermore, modules can have default inputs, and dependencies on or conflicts with other modules.

Note that this modular structure is further explained with examples in the following sections.

There are two stages to running SecGen:

Stage 1) building the project output.

Stage 2) building VMs based on the project output.

At Stage 1, all available modules are read, and the scenario definition is also read. The scenario definition is used to select the modules to include for each system. In some cases modules will automatically add other modules to the scenario: either due to a dependency or as a default input to a parameter.

All randomisation happens at Stage 1. Modules that have local code are run to produce output, which is then fed into other modules' parameters.

Librarian-puppet is then used to deploy all of the puppet modules corresponding to the SecGen modules that have been selected into the project output directory. A Vagrantfile is created, which makes reference to all the generated data and puppet modules. Other outputs include files describing the generated scenario, including an XML file listing flags with corresponding hints.

Stage 2 simply involves invoking "vagrant up", which leverages Vagrant to generate and provision the VMs.

4.3. SecGen Modules

The types of SecGen modules are:

- base: a SecGen module that defines the OS platform (VM template) used to build the VM
- vulnerability: a SecGen module that adds an insecure, hackable, state (including realistic software vulnerabilities known to be in the wild or fabricated hacking challenges)
- service: a SecGen module that adds a (relatively secure) network service
- utility: a SecGen module that adds (relatively secure) software or configuration changes
- network: a virtual network card
- generator: generates output, such as random text
- encoder: receives input, such as text, performs operations on that to produce output (such as, encoding/encryption/selection)

The root of a module's directory always contains a secgen_metadata.xml file (illustrated in Figure 1), which defines the attributes of the module. In the case of vulnerability modules, this file contains information about the vulnerability, including CVE, privilege level the successful attacker gains, access level required in order to attack (remote vs local), any metasploit module that can be used to exploit the vulnerability, CVSS score and vector string, difficulty level, and description. This information can be used to filter module selection for scenarios, and also used to specify modules that conflict with each other or to satisfy dependencies between modules.

```
<?xml version="1.0"?>
<vulnerability [snip]>
  <name>DistCC Daemon Command Execution</name>
  <author>Lewis Ardern</author>
  <module_license>MIT</module_license>
  <description>Distcc has a documented security weakness
    that enables remote code execution.</description>
  <type>distcc</type>
  <privilege>user_rwx</privilege>
  <access>remote</access>
  <platform>unix</platform>
  <!--module inputs-->
  <read_fact>strings_to_leak</read_fact>
  <read_fact>leaked_filenames</read_fact>
  <default_input into="strings_to_leak">
    <generator type="message_generator"/>
  </default_input>
  <default_input into="leaked_filenames">
    <generator type="filename_generator"/>
  </default_input>
  <!--optional vulnerability details-->
  <difficulty>medium</difficulty>
  <cve>CVE-2004-2687</cve>
  <cvss_base_score>9.3</cvss_base_score>
  <cvss_vector>AV:N/AC:M/Au:N/C:C/I:C/A:C
  </cvss_vector>
  <reference>https://www.rapid7.com/db/modules/
  exploit/unix/misc/distcc_exec</reference>
  <reference>OSVDB-13378</reference>
  <software_name>distcc</software_name>
  <software_license>GPLv2</software_license>
  <!--optional hints-->
  <msf_module>exploit/unix/misc/distcc_exec
  </msf_module>
  <hint>On a non-standard port</hint>
  <solution>Distcc is vulnerable, and on a high port
    number.</solution>
  <!--Cannot co-exist with other installations-->
  <conflict>
    <software_name>distcc</software_name>
  </conflict>
</vulnerability>
```

Figure 1: secgen_metadata.xml

4.4. Scenario Specification

The selection logic for choosing the modules to fulfill the specified constraints can filter on any of the attributes in each module's secgen_metadata.xml file (for example, difficulty level and/or CVE), and any ambiguity results in a random selection from the remaining matching options (for example, any vulnerability matching a specified difficulty level). The filters specified are regular expression (regexp) matches.

As illustrated in Figure 2, the default scenario defines a scenario with a remotely exploitable vulnerability that grants access to a user account, and a locally exploitable root-level privilege escalation vulnerability.

```
<?xml version="1.0"?>
<scenario [snip]>
  <!-- an example remote storage system, with a
    remotely exploitable vulnerability that can then
    be escalated to root -->
  <system>
   <system_name>storage_server</system_name>
    <base platform="linux"/>
    <vulnerability privilege="user_rwx"
      access="remote"/>
    <vulnerability privilege="root_rwx"
      access="local"/>
    <service/>
    <network type="private_network" range="dhcp"/>
  </system>
</scenario>
```

Figure 2: default_scenario.xml

```
<?xml version="1.0"?>
<scenario [snip]>
  <system>
    <system_name>file_server</system_name>
    <base platform="linux"/>
    <vulnerability module_path=".*nfs.*">
    <input into="strings_to_leak">
      <value>Leak this text and a flag</value>
      <generator type="flag_generator"/>
    </input>
    </vulnerability>
    <network range="dhcp"/>
  </system>
</scenario>
```

Figure 3: Module parameterisation

Parameterisation enables modules to be fed input. For example, a vulnerability can be fed information to leak as output. And modules can be nested, so that the output from nested modules are passed into the input for the parent modules. SecGen module parameters are analogous to named and (always) optional parameters. For example, Figure 3 shows a system with a NFS share that will host a publicly exported file containing leaked text, including a generated flag.

Figure 4 illustrates how the flag generator can be nested within an encoder to first encode the flag before it is leaked.

Generators and encoders will always produce/return an (unnamed) array of strings, which can be directed to input parameters for other modules (by parameter name into modules they are nested under, as illustrated in Figure 4). All string encoders will accept and process the "strings_to_encode" parameter, so it's safe to pass input into any randomly selected encoder. It is also possible to direct the output from multiple modules to input to the same module parameter, by nesting multiple modules under an `<input>` element. In which case each of the nested inputs to that same parameter are concatenated into the same array of strings.

```
[snip]
  <vulnerability module_path=".*nfs.*">
    <input into="strings_to_leak">
      <encoder name="BASE64 Encoder">
        <input into="strings_to_encode">
          <value>Leak this text</value>
          <generator type="flag_generator"/>
        </input>
      </encoder>
    </input>
  </vulnerability>
[snip]
```

Figure 4: Nesting encoders

Note that module definitions can specify a set of (potentially nested) modules that should be selected for input to a parameter, if an input is not specified in the scenario. This is illustrated in Figure 1, where strings_to_leak has a generated message as it's default value.

Other advanced features include methods for ensuring modules selected are unique, and using datastores (variables) to hold values for reuse. Datastores are similar to variables in other languages. However, a datastore always holds an array of strings, and writing to the datastore concatenates to the array of strings. Datastores can be used to store generated information for complex scenarios, such as the organisation's name, employees, etc, which can then be fed through to websites, and services, user accounts, and so on.

This specification language has proven to be a powerful method for generating meaningful challenges and systems. However, through our experience with collaborative software development we concede it has a steep learning curve to development.

Access to existing scenarios makes SecGen's barrier for entry low. This removes the requirement for end users of the framework to understand SecGen's configuration specification. Scenarios can be found in the `scenarios/` directory. Developed scenarios include a

set of VMs for a randomly generated fictional organisation, with a desktop system, webserver, and intranet server, ready for a security audit; and a set of VMs for hosting a CTF competition; and many other example scenarios.

4.5. Implemented Functionality

Over 100 modules have been implemented to date, which provides functionality that makes the SecGen framework practically useful. 11 service modules provide a range of secure services including NFS, IRC, NTP, SMB, FTP, database, and web servers. 11 utility modules provide various system configurations such as user accounts, firewalls, and desktop environment configuration. 24 vulnerability modules provide a range of vulnerable services, such as vulnerable NFS, IRC, SMB, FTP, SSH, web servers and web apps, vulnerable desktop configurations, access control and system configurations, the majority of which can be deployed either as CTF challenges or to provide open-ended simulations. 45 generator modules can provide content, such as business and user names, addresses and email addresses, messages, filenames and directories, images, ssh keys, passwords, and CTF flags. 13 encoder modules provide various forms of encryption, conversion between data formats, and encoding methods. Network modules provide network cards for scenarios with multiple network segments. The focus has been on deploying Linux systems; however, we have had success testing Windows functionality, which is in development.

4.6. Front End: CTF Website and Hints

A website has been developed to provide a front end to SecGen generated VMs for CTF events. The website provides a scoreboard, timer, flag submission, progress indication, and hints.

SecGen automatically generates a `marker.xml` file, listing all the flags, and for each flag a list of corresponding hints, based on the metadata for the module. Hints range from general hints, such as trying port scans, to progressively more specific hints all the way through to the description of a solution. The approach taken for hints was to penalise points for each hint taken, although the penalties for hints will never exceed the reward for submitting the flag. Where multiple flags are behind the same challenge (for example, differently encoded flags behind the same vulnerability), submitting any of those flags unlocks repeated hints (such as how to exploit the vulnerability).

5. Evaluation

5.1. Rich-scenarios and randomisation

SecGen provides a platform that uniquely and demonstrably achieves the aims described in Sections 3.1 to 3.5. The framework can demonstrably generate highly-randomised VMs based on rich-scenarios.

5.2. Experience Teaching Using SecGen

SecGen has been applied in HE to provide security exercises, from small-scale exploitation exercises through to open-ended audits of a complex set of VMs. Recently a rich-scenario was developed which was used to create targets for team-based security audit projects. The scenario includes a web server, intranet server, and desktop system. The attacker (Kali Linux) VM was placed on the same network segment as the webserver (ie. sharing the same virtual network card), which in turn was connected to the intranet and desktop systems. The students were required to breach the webserver before pivoting attacks through to the other systems. The scenario includes a generated business name, manager, and employees, and involves a random selection of secure and vulnerable services and configurations. A security audit remit was also generated for each team. Student teams followed a security audit methodology and completed a writeup. The output from SecGen was used to assist marking.

5.3. CTF Using SecGen

SecGen was used to generate a set of VMs for use in hosting a UK-wide full-day in-person CTF event. 59 students from 10 universities competed. 3 VMs were generated for the event using SecGen, including one with random decoding challenges, one with a random set of vulnerabilities and image steganography, and another with a root-level privilege escalation. At the end of the event SecGen was presented to participants.

An evaluation survey was run to gauge success of the framework and the event. The response rate was 21 of the 59 participants from 8 of the 10 universities that took part. 52% were postgraduate students, 43% undergraduate (1 reported “N/A”). Many were completing the first year of their degree (38%).

Satisfaction of the event was good, with only one participant responding negatively on the scale of satisfaction. A multiple linear regression analysis was conducted to understand whether the level of satisfaction with the event was impacted by the level of study (not applicable/undergraduate/postgraduate), year of current course (not applicable/first year/mid-course/final year), whether they had taken part in a Capture The Flag (CTF) or other hacking challenge before (yes/no), level of knowledge and understanding of cybersecurity, and sex of the participants (male/female/prefer not to say). All assumptions such as independence of residuals, evidence of multicollinearity, and assumptions of normality were met. Examining all of the independent variables, the overall model that was found to have best fit of the data has $F(3,17) = 3.313$, $p < 0.05$, $R^2 = 0.369$, two of the independent variables (stage of study and whether they had taken part CTF before) have statistically significant

contribution in explaining variation (nearly 37%) of the dependent variable (satisfaction with the event) with $p < 0.05$. Table 1 (below) represents regression coefficients with standard errors. The model suggests that in general the participants who are at a later year of study were less satisfied with the event compared to participants at earlier stages of study. The result might support the view expressed in previous research that there is a need to preserve balance between difficulty and ease for designing security competitions with respect to the target audience [5]. The higher satisfaction amongst those that had participated in CTF previously perhaps supports the findings from qualitative data that indicated an appreciation for the uniqueness of the event such as the "attack-format" and use of attack tools, which could be appreciated more by the participants with past hacking challenge experience.

Table 1: Summary of Multiple Regression Analysis on Satisfaction

Variable	B	SE_B	β
Intercept	5.432	0.656	
Level of study	-0.322	0.302	-0.213
Year of study	-0.425	0.169	-0.503*
CTF experience	-0.825	0.355	-0.460*

Note: *p<.05; B=unstandardized regression coefficient;
SE_B=Standard error of the coefficient; β=standardized coefficient

81% (n=18) reported that their level of knowledge and understanding of cyber security increased as a result of participating in the event. 81% also expressed an interest in competing in similar events in the future (on a 1-5 Likert scale M=4.43, SD=1.12), with positive but slightly lower interest in online team competitions (M=4.14, SD=1.10), online individual (M=4.10, SD=1.13), and offline (M=3.67, SD=1.35).

The difficulty level was good. During the one day event no team completed every SecGen flag (min=1, max=18, out of 21 possible flags). On a 5 point Likert scale of too easy to too hard, 67% (n=14) selected '3' (not too easy or too hard) (M=3.10, SD=0.7).

The hints system received a mixed response, with participants largely divided over how hints should be implemented in a CTF event. 19% thought the best approach to hints was to have multiple hints per flag - at a penalty (as with the SecGen VMs), another 19% prefered having one hint per flag with no penalty, 19% prefer to have free hints from organisers directly, 14% to have one hint per flag at a penalty, and 29% "Other" with various comments, including an indication that teams avoiding making use of the hints, or that they found the hints unhelpful or too helpful.

Significantly, a large number of those who participated responded that they were interested in making use of the SecGen framework in the future. 86% (n=19) would compete in similar CTF events using SecGen (1 answered "No", 1 other "Not sure"), 72% responded they were interested in browsing the source code to understand the challenges, 63% would use SecGen to generate VMs as personal challenges, 59% were interested in hosting their own CTF events using the framework, and 55% were interested in contributing to SecGen development.

Qualitative data also indicates a positive experience. Multiple participants noted the uniqueness of the "attack-format", and use of attack tools, which was compared to the usual jeopardy format.

Negative comments were focussed on the networking issues that some teams faced, when configuring the VMs that were distributed to teams' own laptops.

Following the event the authors received significant interest in using SecGen to run further CTF events for universities and schools.

6. Future Work

SecGen benefits from the development of further modules to add functionality, such as more vulnerabilities, generated content, encoding methods, and CTF challenges. The authors are developing further SecGen modules and still in the process of converting CTF challenges that have been developed.

Work is in progress to incorporate further digital forensics challenges, and output to forensic disk images, such as E01 files. Related work includes incorporating Microsoft Windows baseboxes and vulnerabilities into SecGen. Work is also ongoing to add cloud deployment of SecGen VMs, specifically to an oVirt-based lab infrastructure. Work is also ongoing to further integrate lab sheet based lab exercises, with randomised worksheets. The platform will be extended with further gamification and immersive scenarios.

7. Conclusion

SecGen provides a flexible and highly modular framework that generates VMs based on scenario definitions that can include randomisation of vulnerabilities (from in-the-wild software vulnerabilities and misconfiguration, to randomised CTF-style challenges), secure services and configuration, and content that can be generated and encoded to provide meaningful *rich-scenario* style challenges. SecGen has been successfully used to enhance security education, by providing randomised targets for lab exercises, large team project security audits, and for generating CTF competition VMs. SecGen can be used to overcome the challenges of generating unique security challenges (and the issues inherent when not randomising tasks given to students), and is free and open source software (FOSS), ready for use in security education. The authors have clear plans for continued development and future work.

Acknowledgements

This project is supported by a Higher Education Academy (HEA) learning and teaching in cyber security grant (2015-2017). Tom Chothia managed University of Birmingham's contributions to the project.

References

[1] A. Mansurov, "A CTF-Based Approach in Information Security Education: An Extracurricular Activity in Teaching Students at Altai State University, Russia," *Modern Applied Science*, vol. 10, no. 11, p. 159, Aug. 2016.

[2] T. Chothia and C. Novakovic, "An Offline Capture The Flag-Style Virtual Machine and an Assessment of Its Value for Cybersecurity Education," in *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., 2015.

[3] C. Eagle and J. L. Clark, "Capture-the-Flag: Learning Computer Security Under Fire," Jul. 2004.

[4] "Vulnerable By Design ~ VulnHub." [Online]. Available: https://www.vulnhub.com/. [Accessed: 05-May-2017].

[5] N. Childers *et al.*, "Organizing Large Scale Hacking Competitions," in *Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Berlin, Heidelberg, 2010, pp. 132–152.

[6] W. Feng, "A Scaffolded, Metamorphic CTF for Reverse Engineering," in *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., 2015.

[7] J. Burket, P. Chapman, T. Becker, C. Ganas, and D. Brumley, "Automatic Problem Generation for Capture-the-Flag Competitions," in *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., 2015.

[8] DEF CON Communications, Inc., "DEF CON Hacking Conference - Capture the Flag Archive," *https://www.defcon.org/html/links/dc-ctf.html*, 2013. [Online]. Available: https://www.defcon.org/html/links/dc-ctf.html. [Accessed: 17-Dec-2013].

[9] K. Chung and J. Cohen, "Learning Obstacles in the Capture The Flag Model," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, 2014.

[10] E. Gavas, N. Memon, and D. Britton, "Winning Cybersecurity One Challenge at a Time," *IEEE Security Privacy*, vol. 10, no. 4, pp. 75–79, Jul. 2012.

[11] "RuCTF." [Online]. Available: https://ructf.org/index.en.html. [Accessed: 05-May-2017].

[12] P. Chapman, J. Burket, and D. Brumley, "PicoCTF: A Game-Based Computer Security Competition for High School Students," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, 2014.

[13] "Ghost in the Shellcode." [Online]. Available: http://ghostintheshellcode.com/. [Accessed: 05-May-2017].

[14] "Codegate CTF." [Online]. Available: http://ctf.codegate.org. [Accessed: 05-May-2017].

[15] G. Vigna *et al.*, "Ten Years of iCTF: The Good, The Bad, and The Ugly," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, 2014.

[16] "CTFtime.org / All about CTF (Capture The Flag)." [Online]. Available: https://ctftime.org/. [Accessed: 05-May-2017].

[17] A. S. Raj, B. Alangot, S. Prabhu, and K. Achuthan, "Scalable and Lightweight CTF Infrastructures Using Application Containers (Pre-recorded Presentation)," in *2016 USENIX Workshop on Advances in Security Education (ASE 16)*, Austin, TX, 2016.

[18] NCCDC, "Collegiate Cyber Defense Competition (CCDC)? About." [Online]. Available: http://www.nationalccdc.org/index.php/competition/about-ccdc. [Accessed: 08-May-2017].

[19] J. A. Amorim, M. Hendrix, S. F. Andler, and P. M. Gustavsson, "Gamified Training for Cyber Defence: Methods and Automated Tools for Situation and Threat Assessment," in *NATO Modelling and Simulation Group (MSG) Annual Conference 2013 (MSG-111)*, 2013.

[20] Z. C. Schreuders and E. Butterfield, "Gamification for Teaching and Learning Computer Security in Higher Education," in *2016 USENIX Workshop on Advances in Security Education (ASE 16)*, Austin, TX, 2016.

[21] Mark Gondree, Zachary N J Peterson, and Portia Pusey, "Talking about Talking about Cybersecurity Games," *;login:*, vol. 41, no. 1, 2016.

[22] A. R. Schrock, "Education in Disguise: Culture of a Hacker and Maker Space," *InterActions: UCLA Journal of Education and Information Studies*, vol. 10, no. 1, 2014.

[23] J. Mirkovic and P. A. H. Peterson, "Class Capture-

the-Flag Exercises," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, 2014.

[24] "Facebook CTF is Now Open Source!" [Online]. Available: https://www.facebook.com/notes/facebook-ctf/facebook-ctf-is-now-open-source/525464774322241/. [Accessed: 08-May-2017].

[25] "CTFd/CTFd," *CTFd - CTFs as you need them (GitHub)*. [Online]. Available: https://github.com/CTFd/CTFd. [Accessed: 08-May-2017].

[26] "mcpa-stlouis/hack-the-arch," *HackTheArch: A free open source scoring server for cyber Capture the Flag competitions (GitHub)*. [Online]. Available: https://github.com/mcpa-stlouis/hack-the-arch. [Accessed: 08-May-2017].

[27] "Nakiami/mellivora," *Mellivora is a CTF engine written in PHP (GitHub)*. [Online]. Available: https://github.com/Nakiami/mellivora. [Accessed: 08-May-2017].

[28] "UnrealAkama/NightShade," *NightShade - A simple capture the flag framework (GitHub)*. [Online]. Available: https://github.com/UnrealAkama/NightShade. [Accessed: 08-May-2017].

[29] "picoCTF/picoCTF-Platform-2," *PicoCTF-Platform-2: A genericized version of picoCTF 2014 that can be easily adapted to host CTF or programming competitions (GitHub)*. [Online]. Available: https://github.com/picoCTF/picoCTF-Platform-2. [Accessed: 08-May-2017].

[30] Razvan Beuran, Cuong Pham, Dat Thanh Tang, Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda, "CyTrONE: An Integrated Cybersecurity Training Framework," presented at the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017), Porto, Portugal, 2017, pp. 157–166.

[31] D. Sadigh, S. A. Seshia, and M. Gupta, "Automating Exercise Generation: A Step Towards Meeting the MOOC Challenge for Embedded Systems," in *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education*, New York, NY, USA, 2013, p. 2:1–2:8.

[32] Christian Collberg, "The Tigress C Diversifier/Obfuscator." [Online]. Available: http://tigress.cs.arizona.edu/. [Accessed: 05-May-2017].

[33] Z. C. Schreuders and L. Ardern, "Generating randomised virtualised scenarios for ethical hacking and computer security education: SecGen implementation and deployment," in *1st UK Workshop on Cybersecurity Training & Education (VIBRANT 2015)*, Liverpool, UK.

[34] M. Hashimoto, *Vagrant: Up and Running*. .

[35] S. Walberg, "Automate System Administration Tasks with Puppet," *Linux Journal*, vol. 2008, no. 176, Dec. 2008.

HERRAMIENTAS Y TÉCNICAS DE HACKING

NMAP

Se trata de un programa de código abierto usado para el rastreo de puertos, actualmente es usado como método de evaluación de la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática.
Para ello, envía paquetes definidos a otros equipos, espera la respuesta de esos equipos y la analiza.

Sus características principales son:

- Descubrimiento de servidores (capaz de listar computadoras conectadas a una red).
- Identificación de puertos abiertos en una computadora de destino (objetivo).
- Identificación de los servicios en ejecución en la máquina.
- Identificación del sistema operativo usado y su versión en la máquina (*fingerprinting)*
- Otras características del hardware de red usado por la máquina en cuestión

Se trata de una de las mejores herramientas de escaneo para Ethical Hacking, además, aunque su lanzamiento fue para sistemas Linux, actualmente es multiplataforma y su distribución y uso es gratuito.

Otras funciones que realiza son:

- Consultar a un determinado host para obtener sus subdominios y servidores DNS.
- Encontrar y explotar las vulnerabilidades de una red.
- Mapeado y enumeración de redes
- Realización de consultas DNS masivas contra dominios y subdominios.

Permite realizar tareas de administración relacionadas con la seguridad del equipo para detectar aplicaciones no autorizadas ejecutándose en la máquina.

Cuenta con la ventaja de que es una herramienta difícilmente detectable ya que evade los sistemas IDS para la detección de intrusos y no infiere apenas en las operaciones normales de las maquinas analizadas.

```
root@kali:~# nmap
Nmap 7.70 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
```

Ilustración 12: Ejemplo nmap

BURP SUITE

Muchas veces es llamada la navaja suiza de los hackers, dado que tiene un gran potencial y permite hacer un sinfín de ataques web, lo cual impresiona a la mayoría de las personas la primera vez que la usan. Tiene una estructura basada en componentes, como se puede ver en la Ilustración siguiente:

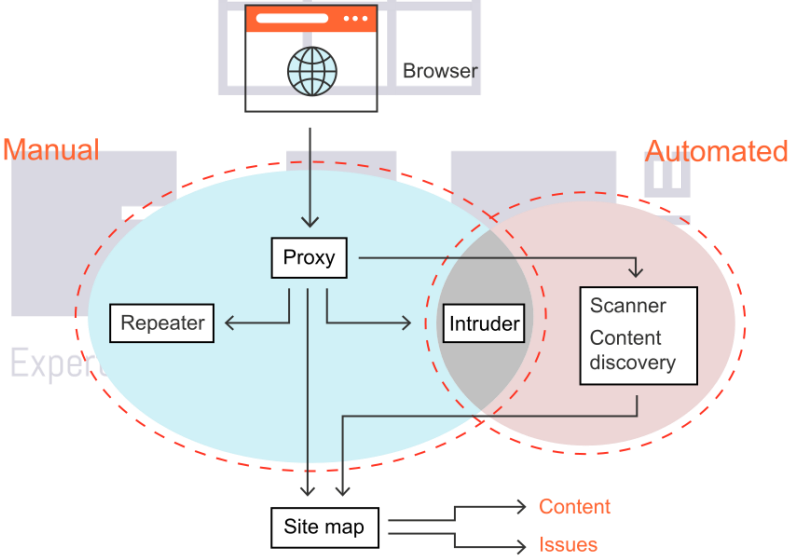

Ilustración 13: Burp Suite

Con ellos, se puede realizar desde una simple intercepción de una petición y modificación de las cabeceras, hasta ataques elaborados como el de predicción de tokens de sesión, o el volteador de bits de cookies. Todos ellos conforman esta potente herramienta, que cuenta con una versión gratis, con funcionalidad limitada destinada a los aficionados y para el uso personal, así como una versión de pago destinada a los profesionales.

Esta herramienta proporciona a los hackers un potente framework para la realización de ataques web dinámicos, ya que cuenta con funciones como:

- Edición de peticiones HTTP.
- Escaneo y filtrado automáticos.
- Gran variedad de informes especializados.
- Analizador de objetivos.
- Descubrimiento de contenido en servidores.
- Programador de tareas.
- Generador de tokens CSRF.

OWASP ZAP

Es la herramienta más usada del mundo para la realización de ataques web. Es gratis, y mantenida por un grupo de voluntarios internacionales. En su página web, tienen guías y tutoriales explicando su uso. Es la alternativa gratis a Burp Suite y, en varios aspectos, es más potente que la anteriormente mencionada. A continuación, se presenta una captura de la aplicación.

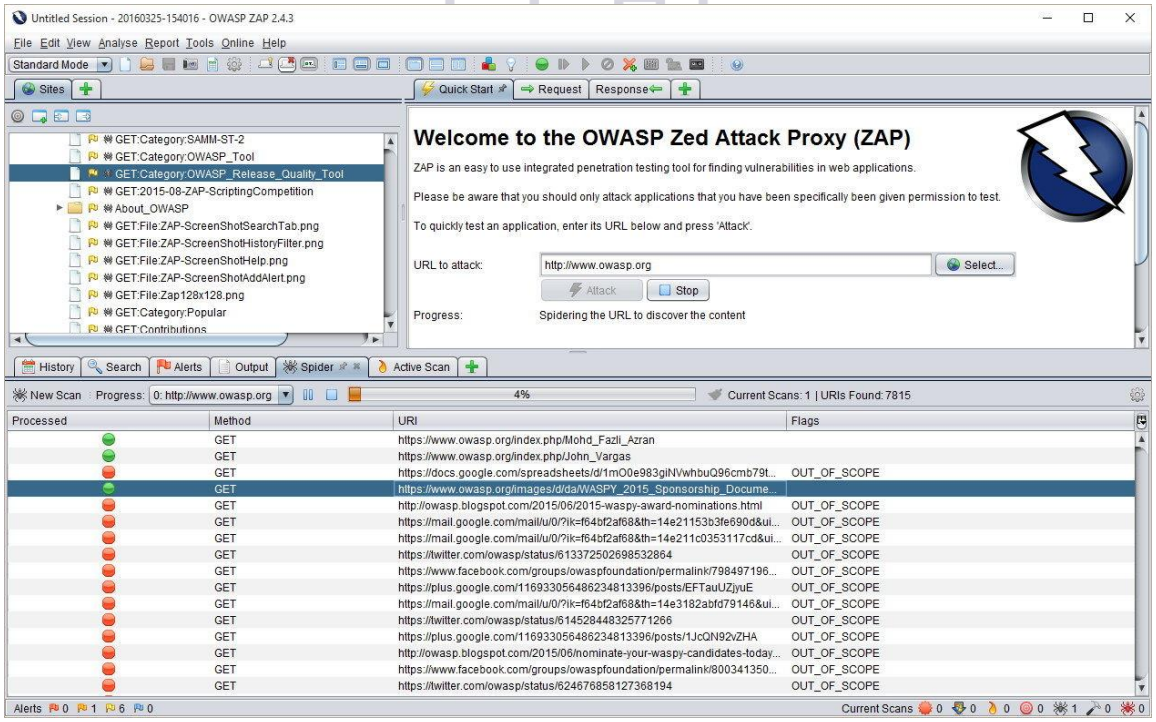

Ilustración 14. ZAP.

METASPLOIT

Se trata del framework de explotación más potente y usado en el mundo. Tiene 2 ediciones, una gratis, para la comunidad, y una de pago, pensada para los profesionales. Por lo general, la edición comunitaria es capaz de hacer todo lo que se puede desear, por lo tanto, lo

recomendable es usar la misma en cualquier situación que no sea de carácter profesional. Cuando se inicia la herramienta de línea de comandos, se puede ver una interfaz como la que se muestra en la siguiente Ilustración.

Ilustración 15. Interfaz de línea de comandos de Metasploit.

Desde ella podremos realizar todo tipo de ataques y usar la completa funcionalidad de la aplicación. Algunas de sus características son realizar e importar datos sobre escaneos realizados con la herramienta nmap, personalizar payloads, lanzar exploits y realizar técnicas post-explotación.

ETTERCAP & BETTERCAP

Ettercap es una suite dedicada al despliegue de ataques de tipo MITM. Cuenta con la posibilidad de hacer sniffing en conexiones de tiempo real, realizar un filtrado de contenido y una larga lista de características interesantes. Soporta ataques activos y pasivos sobre varios protocolos e incluye una amplia lista de herramientas para el análisis del equipo y de la red. Bettercap es el sucesor de Ettercap, escrita en Go, para ofrecer una mayor rapidez de funcionamiento, que amplía la funcionalidad de la predecesora. Ofrece la posibilidad de realizar ataques Wi-Fi, BLE, sobre dispositivos HID y redes Ethernet. Se puede ver una captura de la herramienta a continuación.

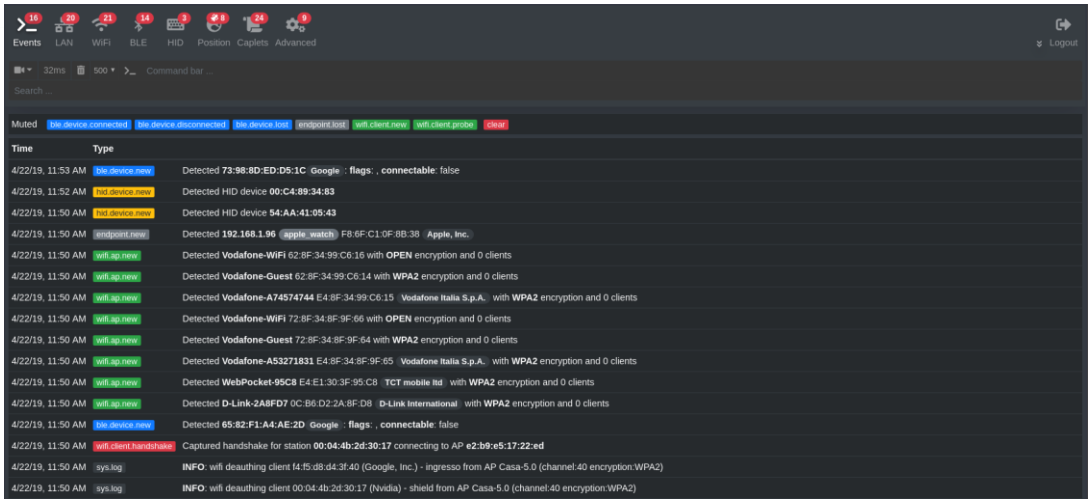

Ilustración 16. Bettercap.

JOHN THE RIPPER

Es un programa de criptografía que emplea los métodos de fuerza bruta o diccionario para el descifrado de contraseñas, mediante la fuerza bruta. Es capaz de romper algoritmos criptográficos de cifrado, hash, DES o SHA1.

Desarrollado inicialmente para sistemas Linux, actualmente es multiplataforma.

El uso más habitual de esta herramienta en administración es la comprobación de la seguridad de las contraseñas de los usuarios en cualquier máquina.

Sus características son:

- Se encuentra optimizado para multitud de procesadores
- Funciona en diversas arquitecturas y sistemas operativos
- Usa ataques por diccionario y por fuerza bruta
- Fácilmente personalizable y modificable (software libre)
- Permite la definición del rango de letras usado para elaborar palabras y sus longitudes
- Permite el pausado del proceso y su retorno más adelante
- Permite incluir reglas en el diccionario para las variaciones tipográficas
- Permite su automatización con herramientas como cron

Su funcionamiento se basa en ataques por diccionario, posee un diccionario de palabras usadas, bien total o parcialmente, en contraseñas típicas, básicas e inseguras, como por ejemplo "*contraseña123456",* va probando todas ellas comparándolas con el hash a descifrar hasta que coincidan dando así con la palabra correcta.
También emplea métodos de fuerza bruta donde prueba con todas las combinaciones posibles ya sean palabras, números o monosílabos; este método es más lento que el anterior y poco usado dado que los ataques por diccionario son más rápidos y eficientes.

```
$ /usr/sbin/john
John the Ripper password cracker, version 1.8.0.6-jumbo-1-bleeding [linux-x86-64-xop]
Copyright (c) 1996-2015 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION]        "single crack" mode
--wordlist[=FILE] --stdin wordlist mode, read words from FILE or stdin
                  --pipe  like --stdin, but bulk reads, and allows rules
--loopback[=FILE]         like --wordlist, but fetch words from a .pot file
--dupe-suppression        suppress all dupes in wordlist (and force preload)
--prince[=FILE]           PRINCE mode, read words from FILE
--encoding=NAME           input encoding (eg. UTF-8, ISO-8859-1). See also
                          doc/ENCODING and --list=hidden-options.
--rules[=SECTION]         enable word mangling rules for wordlist modes
--incremental[=MODE]      "incremental" mode [using section MODE]
--mask=MASK               mask mode using MASK
--markov[=OPTIONS]        "Markov" mode (see doc/MARKOV)
--external=MODE           external mode or word filter
--stdout[=LENGTH]         just output candidate passwords [cut at LENGTH]
--restore[=NAME]          restore an interrupted session [called NAME]
--session=NAME            give a new session the NAME
--status[=NAME]           print status of a session [called NAME]
--make-charset=FILE       make a charset file. It will be overwritten
--show[=LEFT]             show cracked passwords [if =LEFT, then uncracked]
--test[=TIME]             run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..] [do not] load this (these) user(s) only
--groups=[-]GID[,..]      load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]    load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX]    load salts with[out] COUNT [to MAX] hashes
--save-memory=LEVEL       enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL    this node's number range out of TOTAL count
--fork=N                  fork N processes
--pot=NAME                pot file to use
--list=WHAT               list capabilities, see --list=help or doc/OPTIONS
--format=NAME             force hash of type NAME. The supported formats can
                          be seen with --list=formats and --list=subformats
```

Ilustración 17: JOHN THE RIPPER

HASHCAT

Se trata de la herramienta de cracking the hashes más rápida y potente del mundo. Fue la primera y aún a día de hoy sigue siendo la única que integra las reglas de variaciones de caracteres en su kernel personalizado para un alto rendimiento. Es de código abierto, bajo la licencia MIT, multiplataforma, y se caracteriza porque usa todo el potencial que poseen las tarjetas gráficas. A continuación se muestra un ejemplo de su uso.

```
hashcat (v6.2.1) starting...

CUDA API (CUDA 11.3)
====================
* Device #1: NVIDIA GeForce RTX 2080 Ti, 10137/11264 MB, 68MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Precompute-Init
* Early-Skip
* Not-Iterated
* Prepended-Salt
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1100 MB

e983672a03adcc9767b24584338eb378:00:hashcat

Session..........: hashcat
Status...........: Cracked
Hash.Name........: SolarWinds Serv-U
Hash.Target......: e983672a03adcc9767b24584338eb378:00
Time.Started.....: Sun May 23 11:43:13 2021 (1 sec)
Time.Estimated...: Sun May 23 11:43:14 2021 (0 secs)
Guess.Mask.......: ?a?a?a?a?a?at [7]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 24620.9 MH/s (32.19ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 31606272000/735091890625 (4.30%)
Rejected.........: 0/31606272000 (0.00%)
Restore.Point....: 0/857375 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:35840-36864 Iteration:0-1024
Candidates.#1....: 4{,erat -> cyr ~}t
Hardware.Mon.#1..: Temp: 62c Fan: 31% Util:100% Core:1920MHz Mem:7000MHz Bus:16

Started: Sun May 23 11:43:12 2021
Stopped: Sun May 23 11:43:15 2021
```

Ilustración 18. Hashcat.

Esta herramienta soporta, prácticamente, todos los tipos de cifrados que existen a día de hoy, lo cual hace que sea la herramienta más usada para el password cracking.

WIRESHARK

Es una herramienta de análisis de paquetes y protocolos usado en comunicaciones de red para solucionar problemas de intercomunicación, pérdidas de mensajes, análisis de los datos y protocolos, se compara con la herramienta *tcdump* incorporando interfaz gráfica, opciones de filtrado de mensajes y diferentes configuraciones.

Permite observar gráficamente todo el tráfico que circula por una determinada red, el contenido de los mensajes, emisores y destinatarios de estos en comunicaciones TCP.

Esta desarrollado y distribuido en software libre, actualmente es multiplataforma y su distribución es gratuita.

Algunos aspectos importantes de esta herramienta son:

- Se encuentra bajo licencia GPL
- Robusto tanto en modo promiscuo como en modo no promiscuo
- Puede capturar datos de la red directamente o almacenados en un archivo (captura guardada)
- Reconstrucción y traducción de sesiones TCP

Cabe destacar que WireShark se ejecuta en modo *superusuario,* para poder capturar paquetes directamente de la interfaz de red y evitar poner en riesgo la maquina con la ejecución, por ejemplo, de código externo.

Sus usos más frecuentes son:

- Captura de tramas directamente desde la red
- Mostrar y filtrar las tramas capturadas con toda su información
- Edición y retransmisión de las tramas por la red
- Captura de tramas desde un ordenador remoto
- Realizar análisis y estadísticas de la información transmitida
- Exportación y captura de las tramas en diferentes formatos
- Seguimiento de todo tipo de flujos, parámetros o patrones de diseño

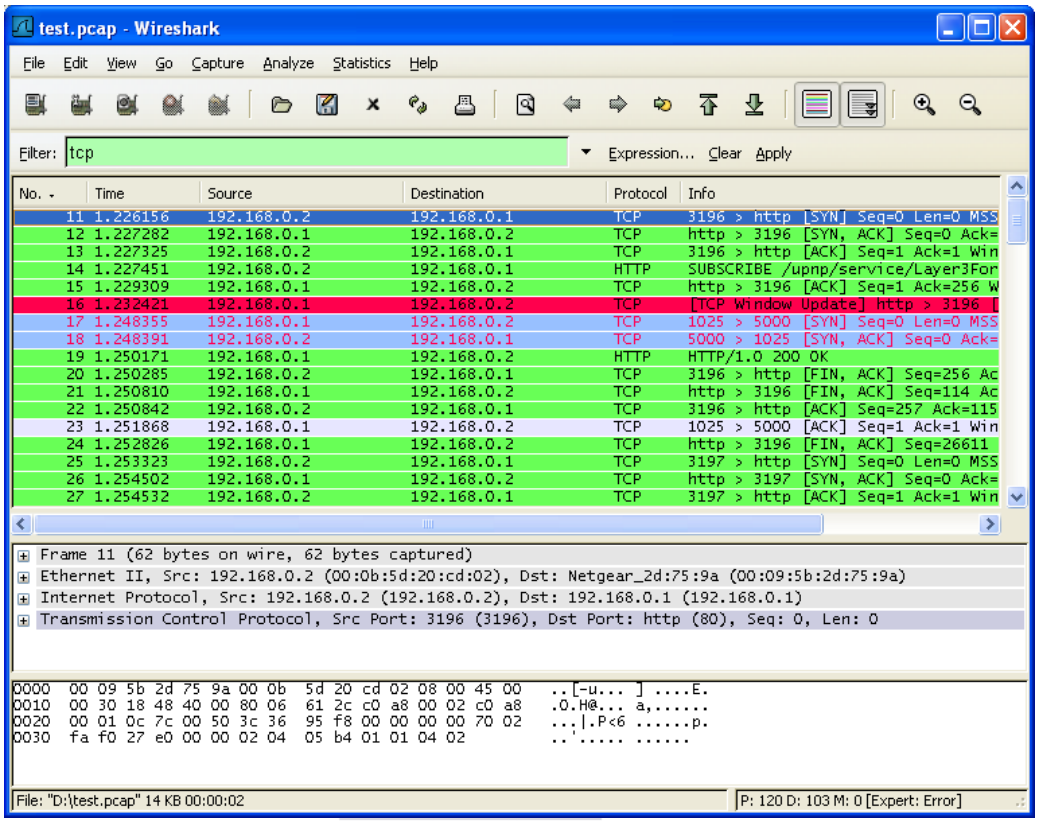

Ilustración 19. Wireshark

SQL injection

Es un método de infiltración de código intruso que se aprovecha de las vulnerabilidades de los sitios web o en las aplicaciones en los datos de entrada que maneja una base de datos o en las operaciones sobre ella.

El procedimiento de inyección SQL se basa en insertar código SQL malicioso dentro del código SQL benigno programado con anterioridad, con el fin de alterar el normal comportamiento del programa y conseguir que se ejecute el código SQL invasor en la propia base de datos teniendo acceso a toda la información allí almacenada.

Su finalidad suele ser de espionaje de la información almacenada en la base de datos, aunque también con fines malignos de sustracción de datos, eliminación de datos o corromper y dejar inservible la base de datos.

Aunque es un método bastante dañino en el almacenamiento de información, existen métodos preventivos que pueden evitar la infiltración en la base de datos, como por ejemplo el uso de lenguaje PERL con el método *DBI::quote* , en Java el uso de *PreparedStatement,* o similares.

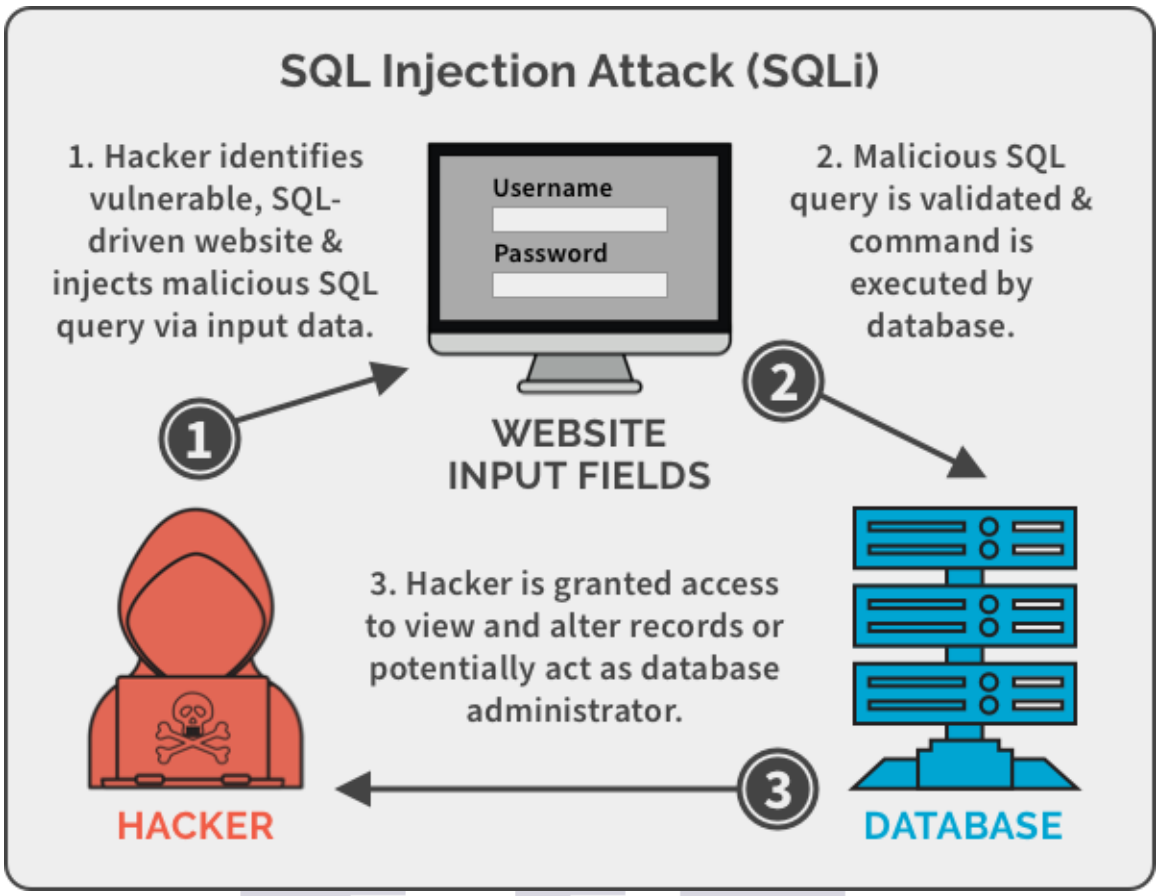

Ilustración 20: SQL Injection

Bate and Switch

Se trata de una técnica que usa anuncios con una apariencia que aporta relativa confianza para engañar a los usuarios y hacer que se dirijan a páginas web maliciosas. Suelen darse en páginas de dudable reputación y que a menudo se dedican a realizar actividades ilegales, como puede ser, por ejemplo, el streaming de películas, algo totalmente ilegal. En algunos casos, después de hacer que el usuario visite la página maliciosa, hacen que se le redirija a la página auténtica para que el usuario tenga la sensación de que todo ha ido correctamente, cuando no es así.

Cross site scripting (XSS)

Estos ataques son un tipo de inyección de código, por la cual scripts (pequeños trozos de código) son acoplados a páginas con buena reputación y aparentemente seguras. Ocurren cuando un atacante usa una aplicación web para mandar o incrustar código malicioso, debido a una mala programación de la misma, con el objetivo de conseguir llegar hasta otro usuario final. Los ataques de este tipo que triunfan, consiguen llegar de manera muy rápida a un gran

número de usuarios, lo cual es un gran problema, y se debe poner gran empeño en prevenirlos.

Debido a que el navegador no tiene idea de si el código JavaScript que le manda la página es o no es legítimo, debe limitarse a simplemente ejecutarlo. Estos programas maliciosos suelen tener como objetivo obtener las cookies, sesiones y datos almacenados en el navegador del usuario como pueden ser contraseñas o nombres de usuario. A continuación, se muestra un diagrama del ataque.

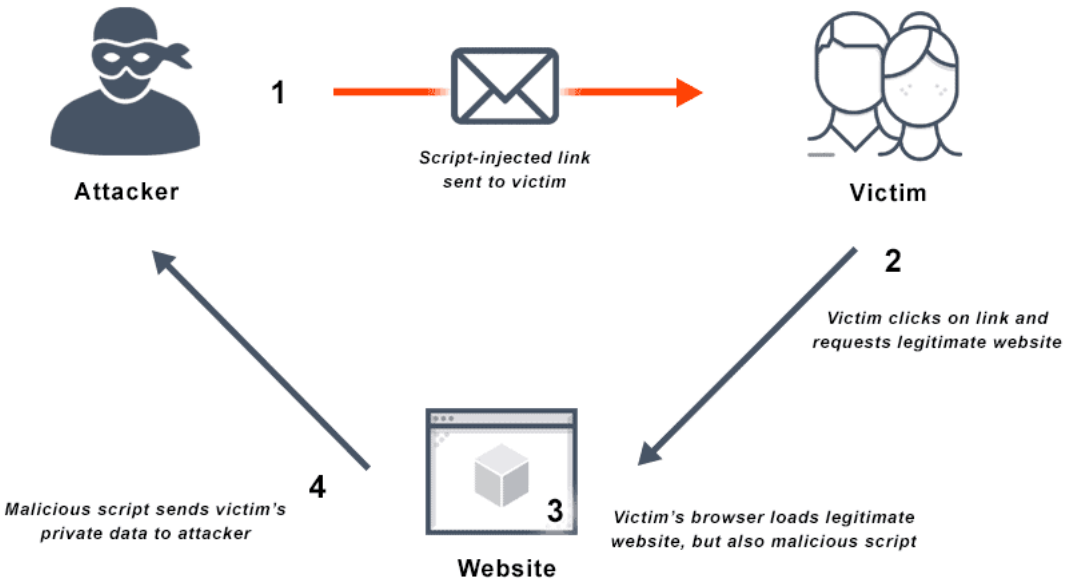

Ilustración 21. XSS.

La organización OWASP (la misma creadora de la herramienta ZAP) tiene un gran listado de artículos en los cuales explica y detalla técnicas de desarrollo web que previenen este tipo de ataques.

DNS spoofing

Es un método peculiar de hackeo en el que los atacantes se aprovechan de que un servidor DNS permite la resolución de nombres de direcciones IP por lo que el ser humano no necesita recordar estas direcciones para cada sitio a visitar, es por ello que, aprovechando esta dependencia que existe, los hackers cambian o alteran esas direcciones IP de los servidores DNS de la víctima para que apunten a otros servidores maliciosos. Prácticamente se trataría de una suplantación de identidad falsa para que la víctima consulte y obtenga respuesta de un servidor maligno en lugar del que debería de consultar.

El método de modificación de las IP de los servidores más común es aprovecharse de las debilidades de configuración del router, ya que algunos traen la opción de gestión remota, lo que facilita mediante una dirección URL el acceso a ese router. Si a ello le sumamos que gran

parte de la gente deja la contraseña por defecto de acceso al router, todo ello facilita el ataque de este tipo.

Los daños más comúnmente causados por este método son:

- Montaje de sitios falsos/maliciosos que sean réplica exacta o sin apenas diferencias para el usuario de algún sitio común del que se quiera obtener información, de este modo, por ejemplo, con un inicio de sesión fraudulento, el usuario será redirigido a otro sitio web ajeno y el atacante obtendrá las credenciales de la víctima. Este método resulta dañino incluso con sitios cifrados con https.
- Explotación de alguna vulnerabilidad como, por ejemplo, solicitándole al usuario un permiso de ejecución de un falso proceso con el applet de Java CVE-2011-3544, donde la victima creerá que se va a ejecutar de un sitio de confianza y el atacante aprovecha este permiso para ejecutar su código malicioso directamente en la máquina de la víctima.

Para protegerse de este tipo de ataques, la mejor forma es deshabilitar la opción de gestión remota de los routers, mejorar las contraseñas eliminando las de fábrica y cambiándolas por unas más seguras y fuertes, además de actualizar java y navegadores para no conceder permisos desconocidos de ejecución.

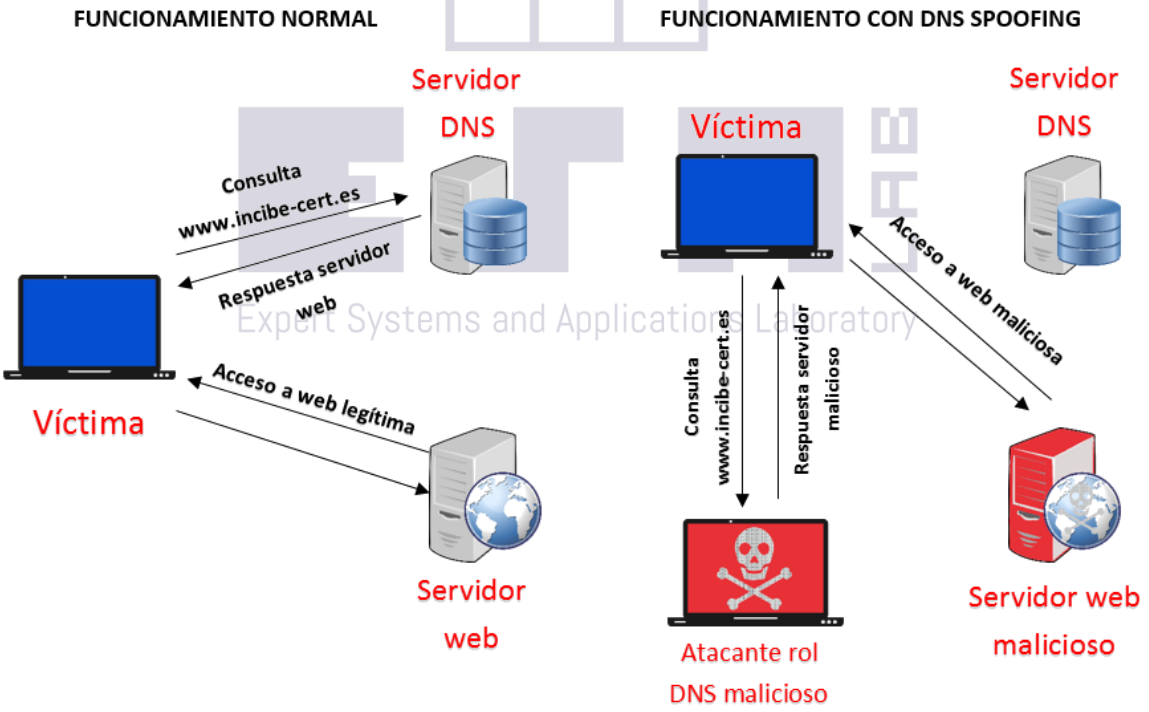

Ilustración 22: DNS Spoofing

Key logger

Se trata de un pequeño programa aparentemente inocente, que tiene como objetivo espiar al usuario, grabar y almacenar todas las pulsaciones de las teclas que realice, para poder, así, obtener detalles sobre la información que introduce en el ordenador.

Los atacantes usan estos programas para obtener detalles valiosos y delicados como pueden ser cuentas bancarias, claves, usuarios y contraseñas, o en el sector empresarial, datos claves estratégicos, o información confidencial.

Tienen dos modos de funcionamiento generalmente, uno para almacenar la información de forma local, en algún dispositivo de almacenamiento que después se pueda retirar para obtenerla, o en línea, enviando la información en tiempo real a un servidor del atacante, como puede ser FTP o bases de datos.

A menudo se usan dispositivos USB que se conectan en la parte de atrás del ordenador, de modo que cuesta mucho darse cuenta de su presencia, pero se han visto casos de hasta teclados y ratones modificados, que escondían un dispositivo malicioso en su interior.

DoS/DDoS

Los ataques de denegación de servicios son un tipo de ataques que en si no dañan la máquina, sino que provocan que un servicio o recurso sea inaccesible para un usuario teniendo la consecuencia de la perdida de conectividad red por el alto consumo de ancho de banda debido a la sobrecarga de peticiones o recursos.

Estos ataques se generan saturando los puertos con demasiada información o peticiones de información consiguiendo así una sobrecarga del servidor y que deje de responder, de ahí el término de *denegación* ya que el servidor denegará más peticiones al usuario por la gran cantidad de ellas recibidas.

Una ampliación de los ataques de denegación de servicios DoS son los ataques DDos, llamados ataques de denegación de servicios distribuidos, el funcionamiento es similar al ataque DoS pero se lleva a cabo desde varios puntos de conexión hacia un único destino, este ataque es el más comúnmente usado y se realiza con una red de bots que se encargar de autogenerar peticiones.

También sirve como método de seguridad realizado para comprobar la capacidad de tráfico que un ordenador o servidor soporta sin volverse inestable o dejar de responder.

Los síntomas comunes para saber si se está sufriendo un ataque DoS o DDoS son:

- Red lenta o demasiado lenta a la hora de navegar por internet o abrir archivos.
- Lentitud a la hora de ingresar en una web o indisponibilidad de la misma.

Los métodos de ataque de este tipo más comunes usando TCP son:

- Consumo de recursos computacionales (ancho de banda, espacio en disco)

- Alteración de la información de configuración como las rutas de encaminamiento
- Alteración de la información de estado como el reseteo o interrupción de sesiones TCP
- Interrupción de componentes físicos de la red, como los routers
- Obstrucción de medios de comunicación entre usuarios de un servicio y la victima rompiendo su comunicación.

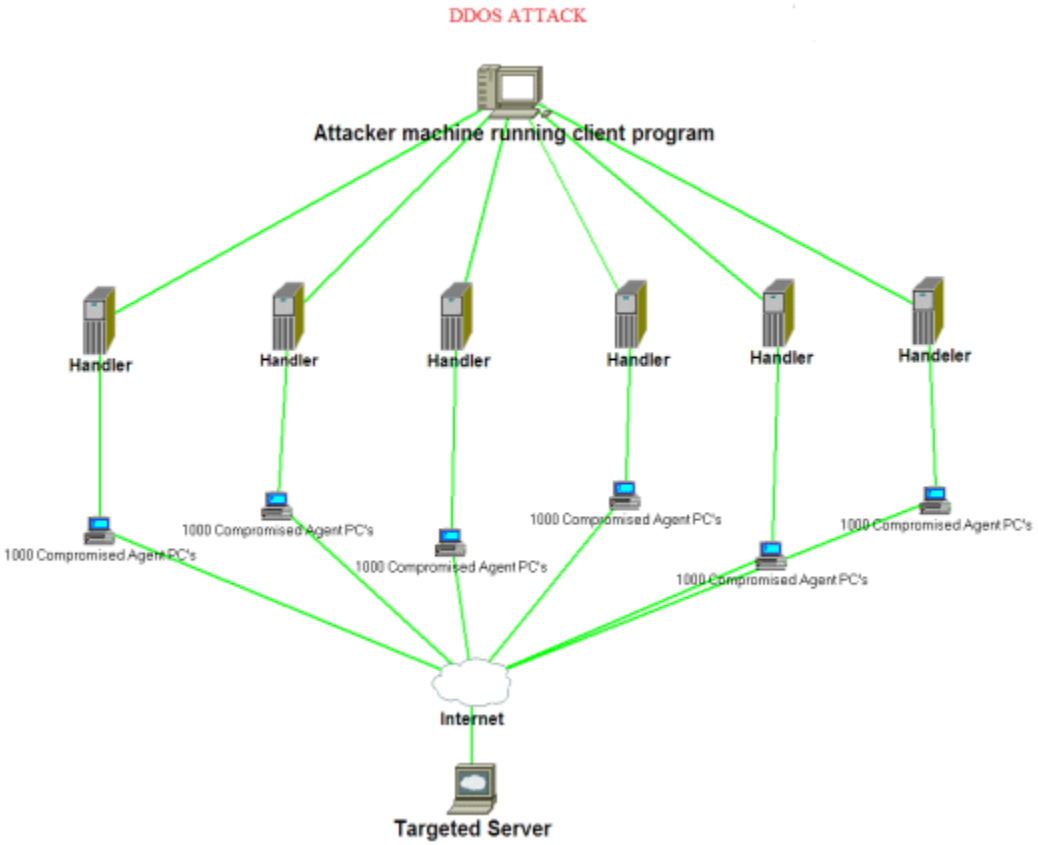

Ilustración 23: DDoS Attack

Ataques de secuestro (hijacks)

Un ataque *hijack* se trata de un tipo de malware usado para modificar la configuración de un navegador web con el fin de secuestrarlo, también extiende su ámbito más allá de navegadores web como por ejemplo el secuestro de páginas web, dominios DNS o bases de datos.

Su misión es modificar lo que reciben los usuarios redirigiéndoles a un servidor maligno, por ejemplo, controlado por los atacantes.

Hijacks puede instalar complementos maliciosos que bloquean los antivirus y cortafuegos del sistema poniendo aún más en peligro a la víctima.

Los tipos de ataque por secuestro o *hijacks* son:

- Secuestro del navegador, se trata del más usado y podría dañar gravemente la seguridad del navegador con el que prácticamente la victima accede a todo.
- Secuestro de páginas web, se trata del secuestro de una página web y la adicción o instalación de complementos maliciosos para el robo de credenciales o información de la víctima.
- Secuestro de DNS, se trata de modificar los servidores DNS para que la víctima sea redirigida a otra web maliciosa.
- Secuestros de sesión, se trata de robar los datos de inicio de sesión o credenciales correctos de la víctima.

Las mejores formas de protegerse ante estos ataques, de primeras es fijarse en las url de cada sitio que se acceda o se vaya a acceder en el navegador, ya que muchas veces incluyen un mínimo cambio para redirigir a web fraudulentas; también se debe de mantener el equipo actualizado, proteger los dominios web propios, o hacer uso de programas y herramientas de seguridad contra ataques hijack.

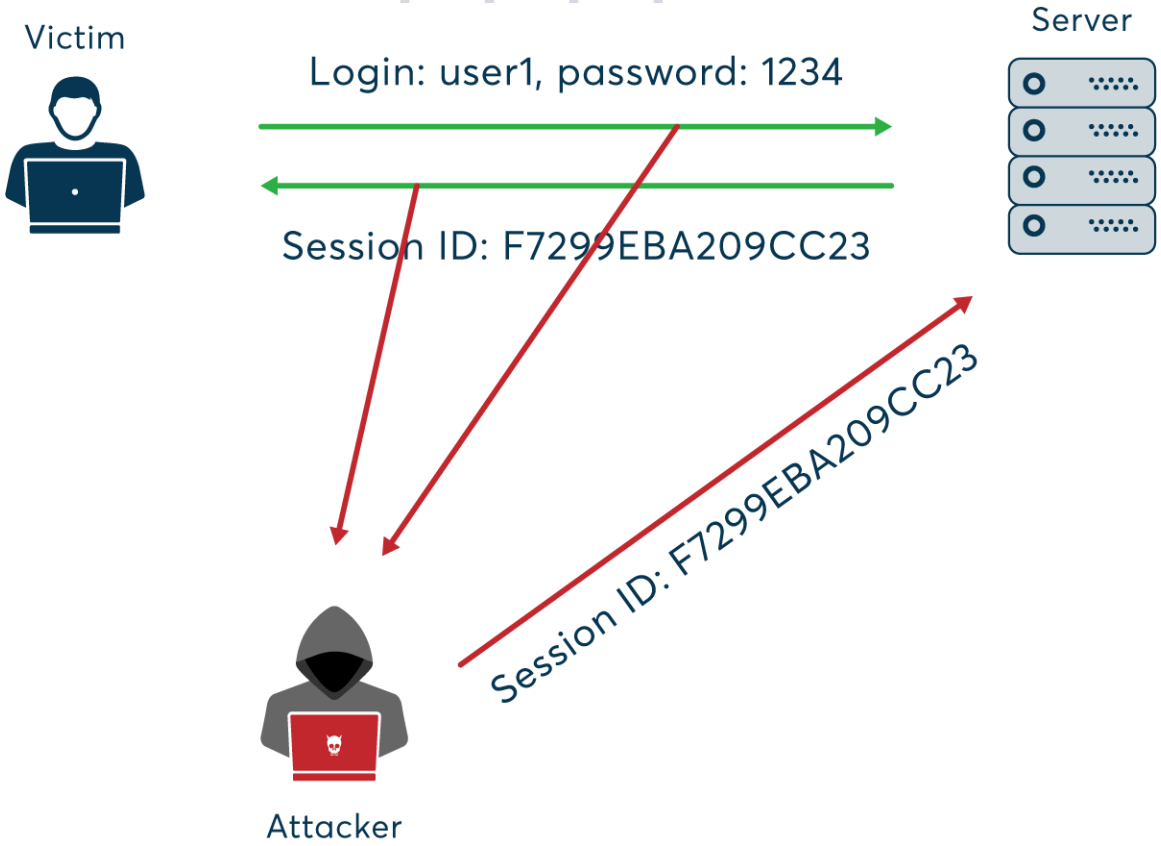

Ilustración 24: Hijack Attack

Punto de acceso falso (Evil Twin AP)

Consiste en la creación de un punto de acceso (red Wi-Fi) que tenga como nombre el mismo que la que el atacante tiene como objetivo, y en hacer que el cliente se desconecte de su punto de acceso mediante el uso de una técnica llamada ataque de desautenticación, que consiste en engañar al cliente haciéndole pensar que el router le ha echado de la red. Es el equivalente al phishing en términos de redes locales. Este ataque suele ser usado para robar la contraseña del punto de acceso, pero también puede usarse para robar todo tipo de datos haciendo un ataque tipo MITM, ya que el cliente se piensa que está usando su red personal, y tiene la seguridad para introducir sus detalles personales, pero no sabe que está siendo espiado y su información robada.

Junto con este tipo de ataques, suele usarse, además, un llamado portal cautivo. Seguro que cuando hemos ido a un aeropuerto o centro comercial nos hemos conectado al Wi-Fi gratis que hay, y al intentar navegar, se redirige al usuario a un portal donde deberá, desde crearse una cuenta o iniciar sesión, hasta aceptar un acuerdo de no usar la red para fines ilegales. En estos ataques, lo más común es que el atacante obligue a la víctima a introducir sus contraseñas.

Hay un gran listado de herramientas para realizar este tipo de ataques, pero quizá, la más famosa es *wifite*, una herramienta para realizar todo tipo de ataques Wi-Fi, con un gran rendimiento y que permite la automatización de dichos ataques. A continuación, se puede observar la interfaz de dicha herramienta.

```
root@kali:~/wifite2# ./Wifite.py --crack

   .;'                     `;,
  .;'  ,;'             `;,  `;,
 .;'  ,;'  ,;'     `;,  `;,  `;,      WiFite v2.00
 ::   ::   :   ( )   :   ::   ::      Automated Wireless Auditor
 ':.  ':.  '. /_\ ,:'  ,:'  ,:'
  '.   '.    /___\    ,:'  ,:'       https://github.com/derv82/wifite2
   '.      /_____\      ,:'
          /       \

[+] Listing captured handshakes...

 NUM  ESSID        BSSID              DATE CAPTURED
 ---  -----------  -----------------  -------------------
   1  ShittyGuest  A6:2B:8C:16:6B:3A  2017-05-14T09:39:19

[+] Select handshake num to crack (1-1): 1
[+] Different ways to crack /root/wifite2/hs/handshake_ShittyGuest_A6-2B-8C-16-6B-3A_2017-05-14T09-39-16.cap:

 # AIRCRACK: CPU-based cracking. Slow.
 aircrack-ng -a 2 -w /usr/share/wordlists/fern-wifi/common.txt /root/wifite2/hs/handshake_ShittyGuest_A6-2B-8C-16-6B-3A_2017-05-14T09-39-16.cap

 # PYRIT: GPU-based cracking. Fast.
 pyrit -i /usr/share/wordlists/fern-wifi/common.txt -r /root/wifite2/hs/handshake_ShittyGuest_A6-2B-8C-16-6B-3A_2017-05-14T09-39-16.cap attack_passthrough

 # JOHN: CPU or GPU-based cracking. Fast.
 # Use --format=wpapsk-cuda (or wpapsk-opengl) to enable GPU acceleration
 # See http://openwall.info/wiki/john/WPA-PSK for more info on this process
 aircrack-ng -J hccap /root/wifite2/hs/handshake_ShittyGuest_A6-2B-8C-16-6B-3A_2017-05-14T09-39-16.cap
 hccap2john hccap.hccap > hccap.john
 john --wordlist "/usr/share/wordlists/fern-wifi/common.txt" --format wpapsk "hccap.john"

 # OCLHASHCAT: GPU-based cracking. Fast.
 # Visit https://hashcat.net/cap2hccapx to generate a .hccapx file
 hashcat -m 2500 /usr/share/wordlists/fern-wifi/common.txt generated.hccapx

root@kali:~/wifite2#
```

Ilustración 25. Herramienta Wifite.

Robo de cookies

También llamado raspado de cookies o Cookie Scraping se trata de un método de secuestro de sesión o secuestro de cookies donde el atacante se apodera de la sesión de la víctima. Esta sesión comienza cuando el usuario introduce sus credenciales correctamente y finaliza cuando cierra sesión.

Cuando un internauta inicia sesión en una aplicación web, el servidor establece una cookie de sesión temporal en el navegador web. Gracias a esta cookie de sesión temporal, sabemos que ese usuario concreto está conectado una sesión en particular. Hay que señalar que un secuestro de sesión exitoso sólo se va a producir cuando el ciberdelincuente conozca la clave de sesión de la víctima o el ID de sesión. Así, en el caso de que pueda robar las cookies de sesión, puede hacerse cargo de la sesión del usuario.

Los procedimientos típicos para el robo de cookies son:

- Ataque **Session Fixation** o **fijación de sesión** es un tipo de intento de Phishing. En este procedimiento el atacante envía un enlace malicioso al usuario objetivo por correo electrónico. Luego, en el momento en que el usuario inicia sesión en su cuenta haciendo clic en ese enlace, el pirata informático conocerá el ID de sesión del usuario. A continuación, cuando la víctima inicia sesión con éxito, el pirata informático se hace cargo de la sesión y ya tiene acceso a la cuenta y a toda la información de la víctima.

- **Ataque de secuencias de comandos entre sitios (XSS)** donde el ciberdelincuente engaña al sistema informático de la víctima con un código malicioso de forma segura que parece provenir de un servidor confiable. A continuación, el cibercriminal ejecuta el script y obtiene acceso para robar las cookies. Esto sucede en el momento en que un servidor o página web carecen de parámetros de seguridad esenciales, los piratas informáticos pueden inyectar fácilmente scripts del lado del cliente.

- **Ataques de malware** que se crean para realizar un rastreo de paquetes, lo cual les facilita el robo de las cookies de sesión. Este malware accede al sistema del usuario cuando visita páginas web no seguras o pulsa sobre enlaces maliciosos.

Para prevenir estos ataques la mejor forma y la que más recomiendan es borrar periódicamente las cookies del navegador web y cerrar sesión en todos los sitios web una vez dejemos de usarlos para que las cookies caduquen y no puedan ser usadas.

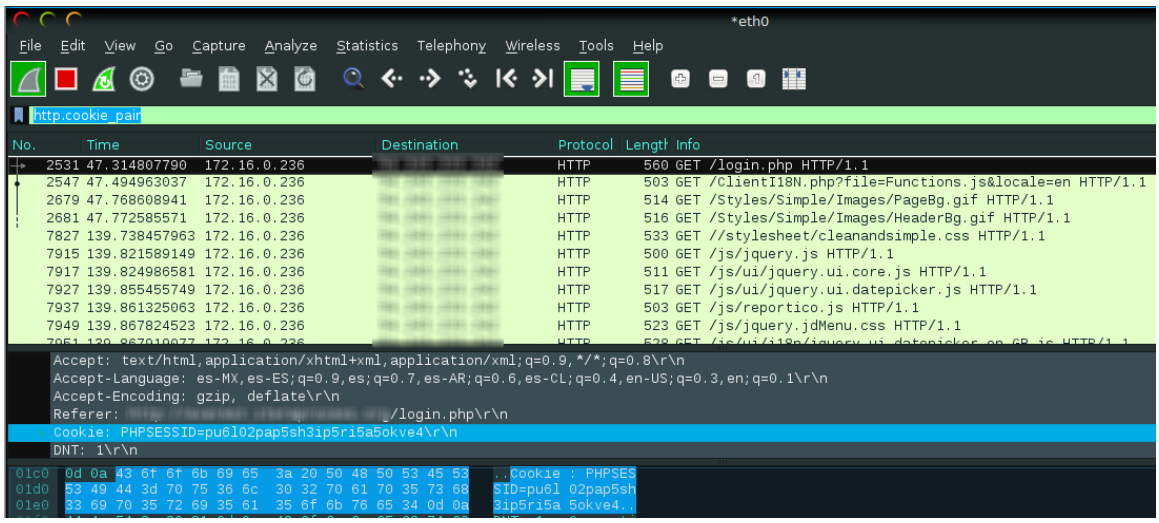

Ilustración 26. Robo de cookies con WireShark

Virus y troyanos

Son programas software maliciosos que una vez instalados en el sistema son capaces de enviar los datos al atacante, corromper el estado de la máquina, bloquear archivos y gran parte del contenido de la máquina, propagarse a todas las maquinas conectadas en red o tener peores consecuencias.

Los virus son malware cuyo objetivo es alterar el comportamiento de la computadora, dejarla inservible mediante la infección de archivos, su destrucción o la creación automática de un gran número de archivos de importante tamaño de ello colapsando la capacidad de la maquina hasta su total paralización. Lo más inusual de un virus es que necesita la intervención del propio usuario para su ejecución, dado que no se ejecutan automáticamente si el usuario no los ejecuta, es por ello que, en la mayor parte de los virus, los archivos son copias o plagios de archivos comunes que un usuario, por ejemplo, descarga de internet como *ejecutable.exe* donde en el momento en que el usuario ejecute ese fichero estará infectado el sistema.

Los troyanos son diferentes a los virus, puesto que su nombre proviene del famoso Caballo de Troya, donde según la historia se introdujo un caballo de madera llena de atacantes dentro del poblado enemigo, en este caso el funcionamiento es parecido, se trata de introducir un malware en la sombra sin que el usuario se dé cuenta de ello y sin alterar el estado de la maquina pero que resulte dañino para la víctima, por ejemplo, robando sus datos o ejecutando acciones en la sombra dado que su principal objetivo es pasar totalmente desapercibido.

Son dos malware similares pero diferentes, dado que ambos necesitan la intervención humana para ser ejecutados, pero uno es destructivo y el otro pasa desapercibido actuando en la sombra y sin desplegarse en otros dispositivos.

Algunos ejemplos conocidos en cuanto a *top* virus informáticos son:

- Gusano Morris.
- CIH/Chernobyl.
- Melissa.
- ILoveYou.
- Mydoom.
- Conficker.
- WannaCry.

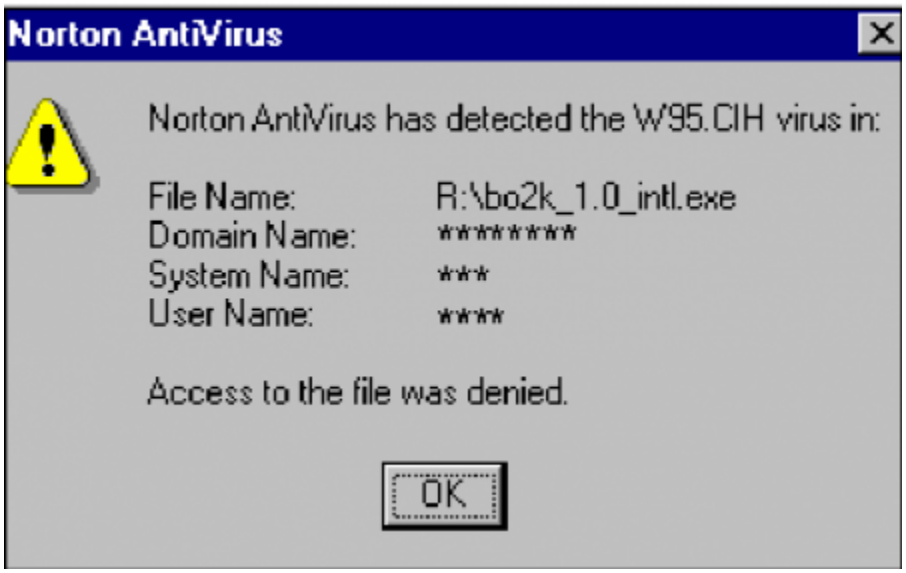

Ilustración 27: Chernobyl Virus

Man in the Middle (MITM)

En un ataque de tipo MITM, el atacante se infiltra de intermediario entre la víctima y el servidor con el que se está comunicando, pudiendo de esta manera leer el intercambio de mensajes entre ambos, monitorizar y controlar el tráfico de mensajes o incluso modificarlos, todo ello sin ser detectado por ninguna de las partes.

La forma más comúnmente usada para realizar un ataque MITM es explotar una conexión WiFi no segura.

Los tipos de ataque más común de MITM son:

- Servidores DHCP, se crean solicitudes DHCP falsificadas para agotar todas las direcciones IP disponibles que pueda asignar dicho servidor, consiguiendo así un denegado del servicio de red a la víctima. En estos ataques, se configura un servidor DHCP falso en la red para emitir direcciones DHCP a los clientes, obligando a la víctima a usar un servidor DNS falso siendo este establecido como su puerta de enlace predeterminada controlada por el atacante.
- Servidores DNS, se crean servidores DNS falsos al que se enrutan nombres de sitios web comúnmente visitados por la víctima con direcciones IP diferentes, creando a

su vez un sitio web falso teniendo así acceso a las credenciales y toda la información que necesite.

- Puntos de acceso WiFi, el funcionamiento es similar, el atacante crea puntos de acceso WiFi falsos con nombres conocidos como el de una empresa o redes abiertas a las que la víctima se conecta, teniendo el atacante acceso a toda la comunicación e información que circula por la red.
- MITW(Man in the Browser), se trata de un tipo de ataque MITM que aprovecha las vulnerabilidades de un navegador web, el funcionamiento es similar al anteriormente explicado donde el atacante espía el uso del navegador, accediendo a toda la información que se suministra a través de él, parecido a un troyano dado que la víctima no se dará cuenta.

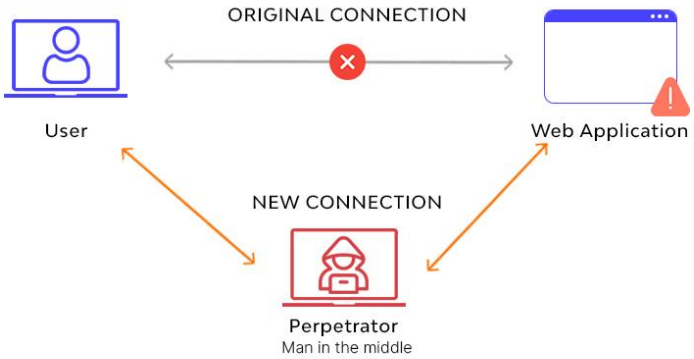

Ilustración 28: Man in the Middle Atack

VULHAB

Vulhab es una página donde podemos encontrar diferentes recursos de aprendizaje y practica sobre seguridad, administración y redes.

Ofrecen retos con un objetivo, que en materias de seguridad suele ser ganar acceso de root (o administrador) de sistema dentro de una máquina virtual, aunque puede ser otro objetivo, como el robo de información conseguir un *flag* o bandera, por poner ejemplos.

VulnHub crea un catálogo de 'cosas' que son (legalmente) 'rompibles, pirateadas y explotables', lo que le permite aprender en un entorno seguro y practicar.

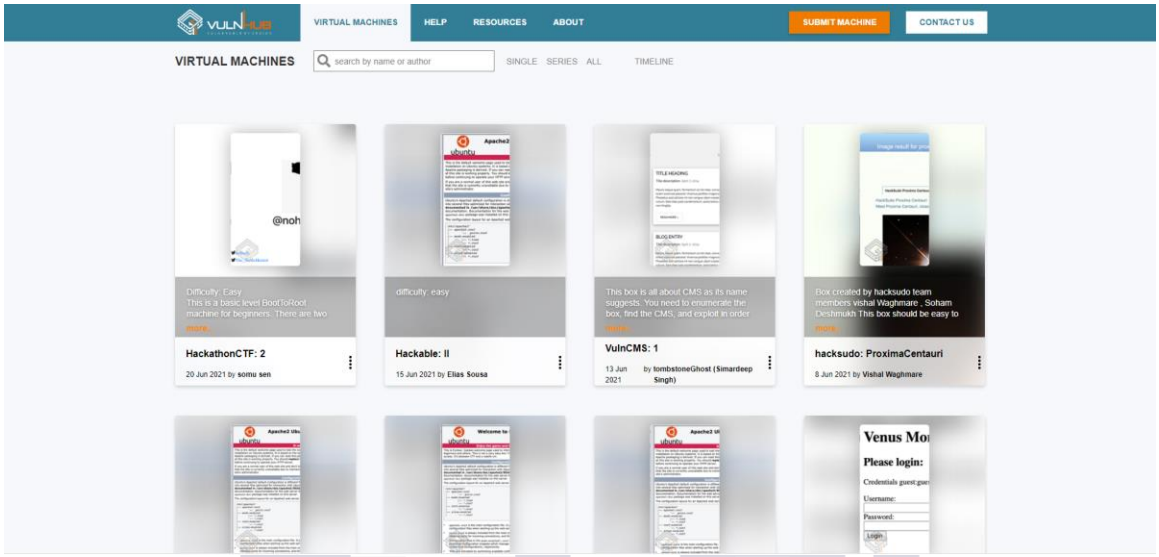

Ilustración 29. Vulhab

Para tener acceso a esta página, accederemos a través del siguiente repositorio creado para el efecto:

- https://github.com/ExpertSystemsApplicationsLab/vulhub/

Instrucciones de clonado:

1. Instalar PIP:
   curl -s https://bootstrap.pypa.io/get-pip.py | python3
2. Instalar Docker:
   curl -s https://get.docker.com/ | sh
3. Ejecutar el servicio docker:
   systemctl start docker
4. Instalar docker compose:
   pip install docker-compose
5. Descargar el Proyecto del respositorio anterior:

wget https://github.com/ExpertSystemsApplicationsLab/vulhub/archive/master.zip -O vulhub-master.zip

unzip vulhub-master.zip

cd vulhub-master

6. Entrar al directorio:
   cd flask/ssti
7. Compilar:
   docker-compose build
8. Ejecutar:
   docker-compose up -d

Después de la prueba, eliminar el entorno con el siguiente comando:

docker-compose down -v

Tipos de Dificultad:

Pueden distinguirse 3 tipos de dificultad en los retos y en las máquinas:

1. Fácil (Easy) para retos sencillos o capturas de banderas.
2. Media (Medium) para retos más avanzados.
3. Dificil (Hard) para retos con mayor grado de dificultad que los anteriores.

Tipos de Maquinas:

Dependiendo de cada creador, dispone de una serie de máquinas con retos categorizados por dificultad, que van desde conseguir acceso root hasta capturar banderas.

Por ejemplo, para una maquina observamos lo siguiente:

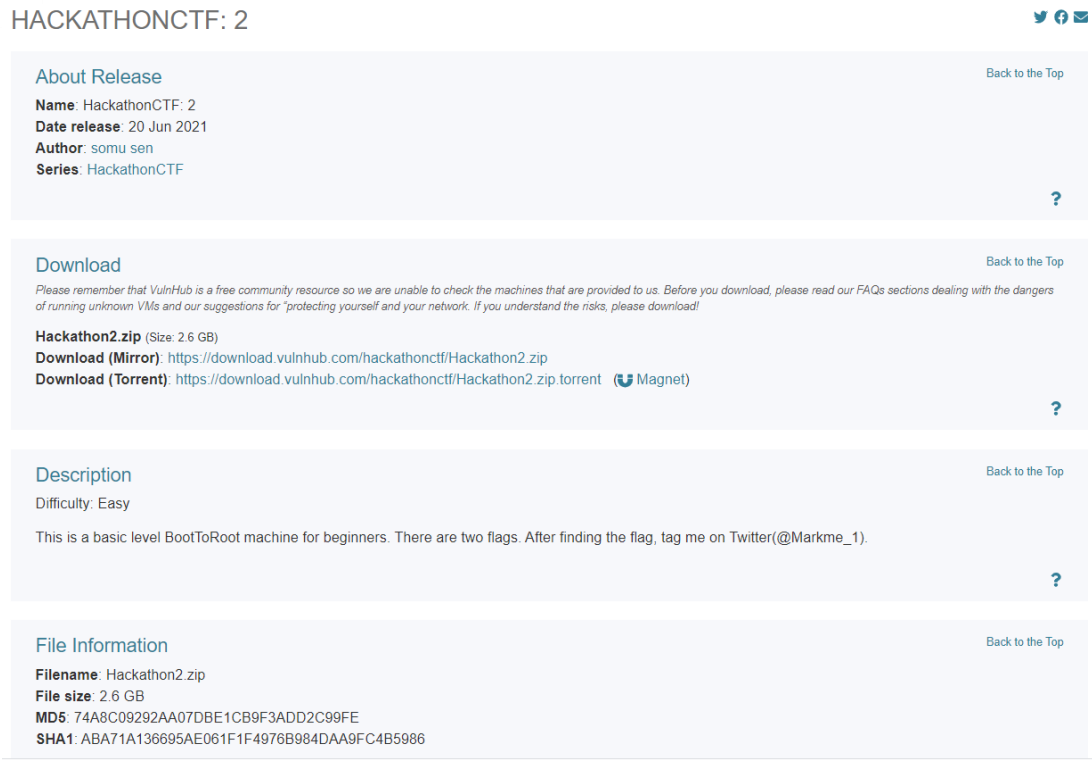

Ilustración 30. Máquina Vulhub I

Ilustración 31. Máquina Vulhub II

Observamos toda la información necesaria para lograr el objetivo de esta máquina, que trata de capturar dos banderas, una vez hecho avisaremos al creador mediante su twitter.
Para ello nos proporciona la máquina virtual en Linux con sus claves de encriptación SHA1 y MD5.

Además de proporcionarnos, en caso de ser necesario, las direcciones del servidor DHCP y la IP.

EJEMPLO DE RESOLUCIÓN DE RETOS

En este apartado se comentan dos ejemplos de resolución de retos de captura de bandera, más conocidos por el término inglés, *Capture The Flag* (CTF). En esta clase de retos el objetivo es encontrar una serie de cadenas de texto que se conocen como banderas. Para ello, hay que usar diversas técnicas de *pentesting* para encontrar vulnerabilidades en la máquina que permitan acceder a información con acceso restringido. Además, en muchas ocasiones, alguna de las banderas va asociada a conseguir tener acceso de administrador (*root*) por medio de una escalada de privilegios.

En concreto, en este apartado se resuelven dos retos CTF propuestos por VulnHub: Cibersploit-1 y Pwned-1. VulHub es una iniciativa que proporciona diversas máquinas virtuales vulnerables con el objetivo de que cualquiera pueda poner en práctica técnicas de *hacking* con fines educativos. Los dos retos resueltos son de nivel principiante y consisten en recuperar tres banderas, de las cuales una requiere el uso de una escalada de privilegios.

Cibersploit-1

1. Se busca la IP de la máquina utilizando el comando netdiscover. En este caso, la máquina vulnerable es el equipo 192.168.56.101.

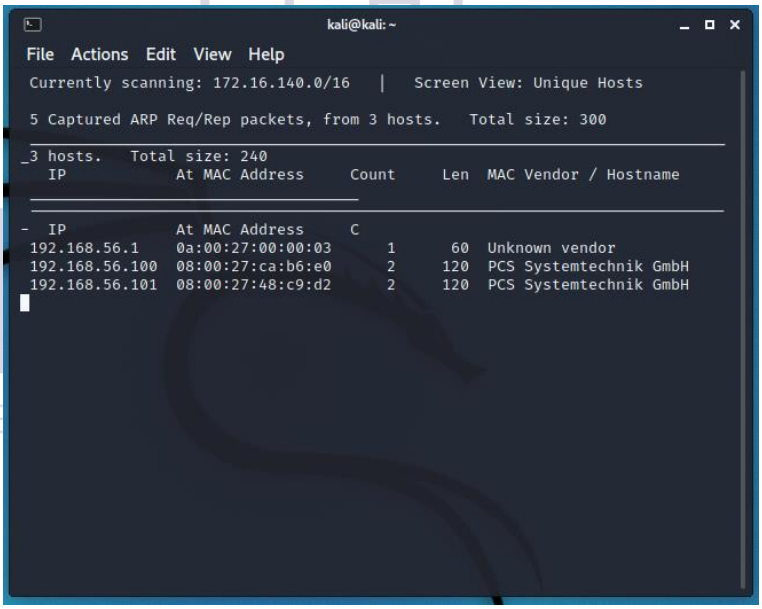

Ilustración 32: Cibersploit-1 buscar IP.

2. Se utiliza el comando nmap para buscar los puertos abiertos que tiene la máquina vulnerable. En este caso, tiene abiertos los puertos 80 (servidores HTTP) y 22 (servidores SSH).

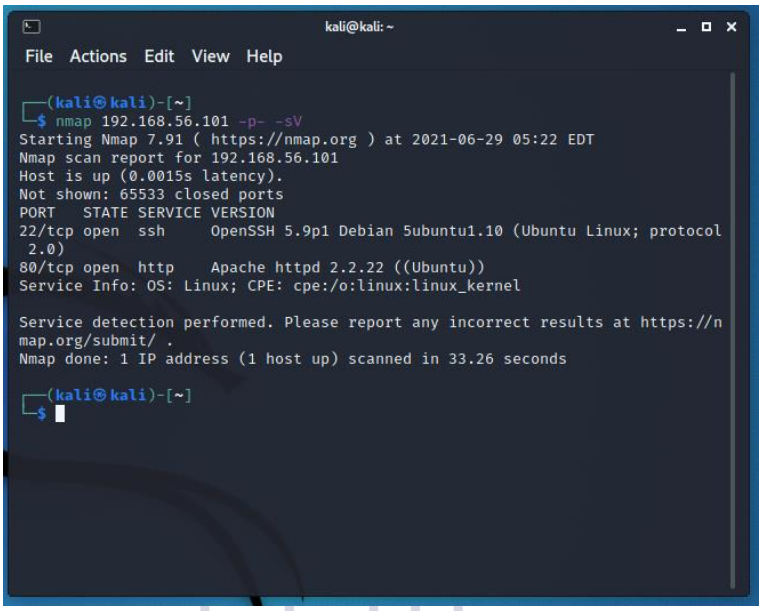

Ilustración 33: Cibersploit-1 nmap.

3. Como tiene el puerto 80 abierto, que es el asociado con los servidores HTTP, se intenta entrar en la IP desde un navegador. A simple vista no parece haber nada útil en esta página. Más adelante se comprobará si hay información valiosa en el código de fuente.

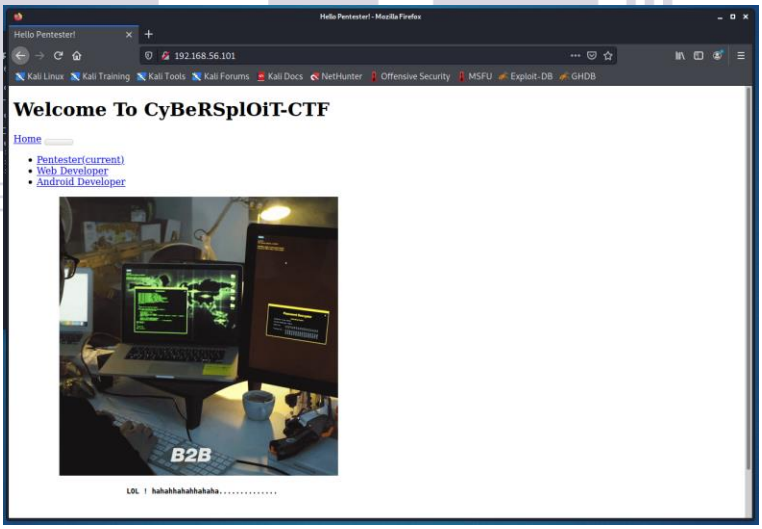

Ilustración 34: Cibersploit-1 exploracion de pagina web.

4. Se buscan las páginas que se sirven con HTTP desde la IP de la máquina vulnerable utilizando el comando dirb.

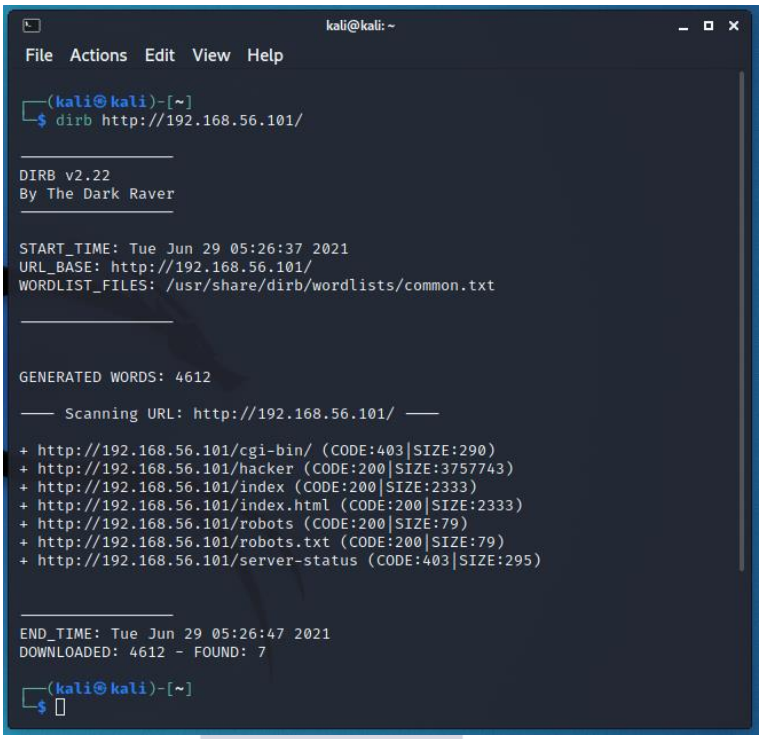

Ilustración 35: Cibersploit-1 comando dirb.

Tras probar con varios, se puede ver que el fichero robots.txt tiene un contenido sospechoso.

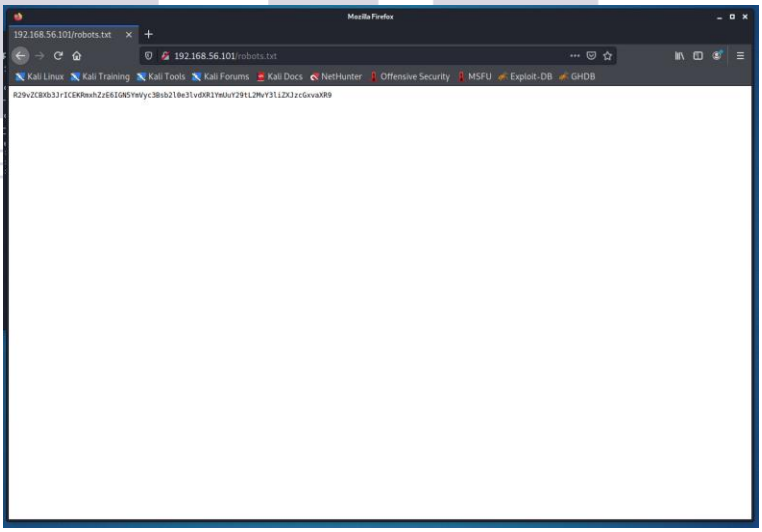

Ilustración 36: contenido robots.txt.

5. Utilizando la Burp Suite se prueban varios métodos de decodificación hasta que con el método Base64 se obtiene un resultado coherente: la primera bandera.

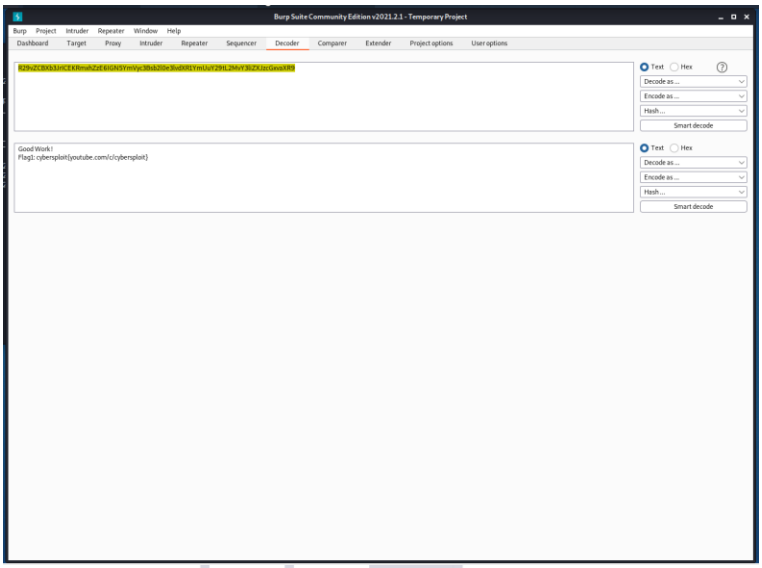

Ilustración 37: Cibersploit-1 Burp Suite.

6. Se vuelve a la página inicial y se analiza el código HTML. Salta a la vista una de las cadenas comentario que contiene un nombre de usuario.

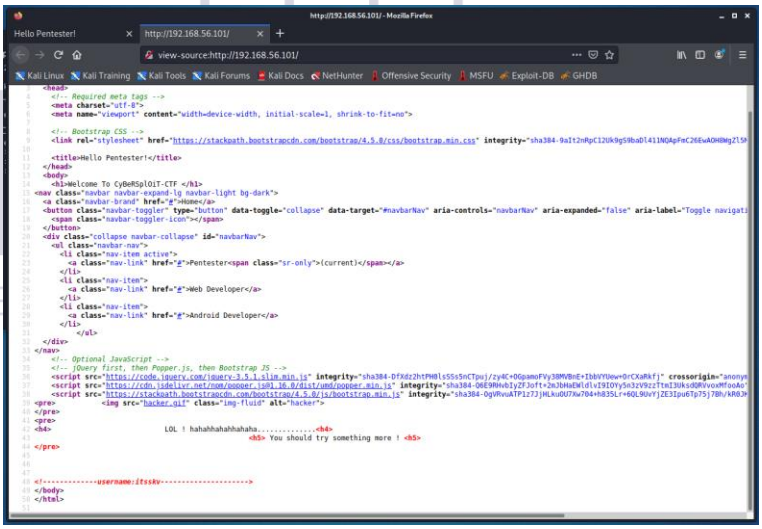

Ilustración 38: Cibersploit-1 exploración de código html.

7. Se accede a la máquina con SSH utilizando el usuario encontrado y la primera bandera como contraseña.

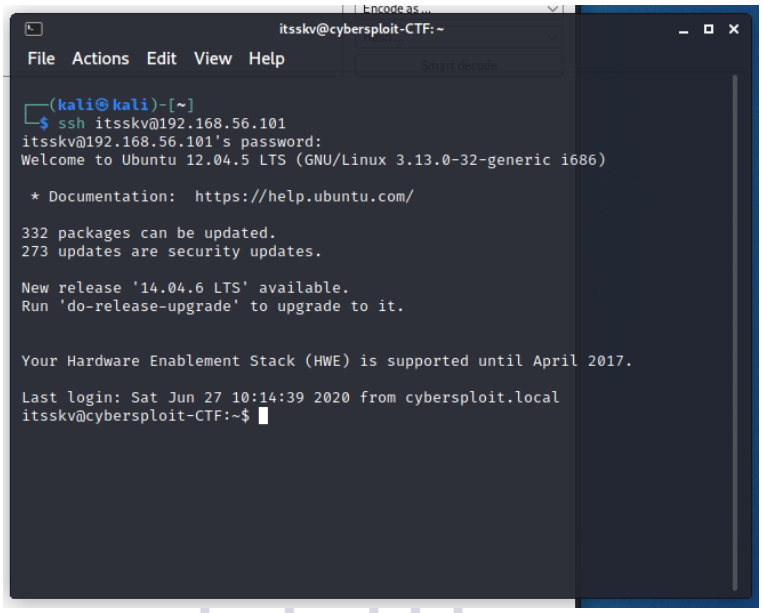

Ilustración 39: Accesso ssh.

8. Si se hace un ls, se puede ver un fichero con nombre sospechoso (flag2.txt) que tiene como contenido una serie de dígitos binarios. Utilizando un conversor web se convierten a texto y se obtiene la segunda bandera. Sin embargo, aún queda el reto más importante: conseguir acceso root.

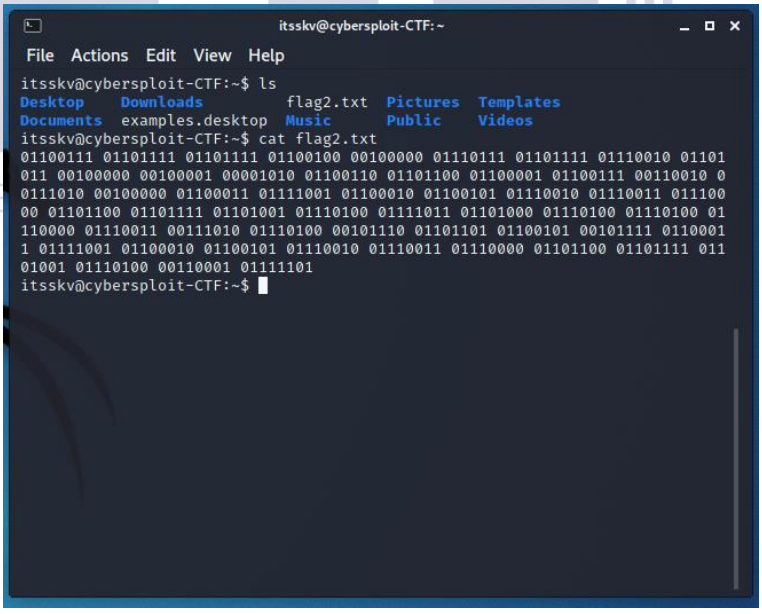

Ilustración 40: Contendio flag.

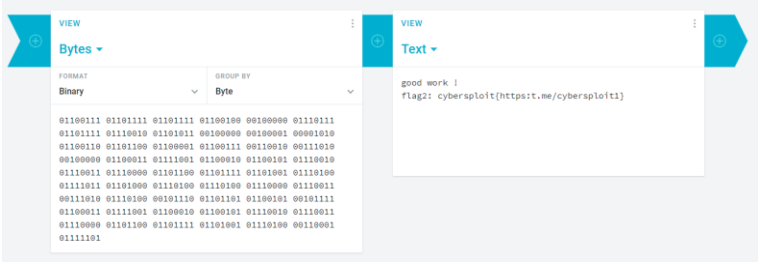

Ilustración 41: Descodificación del flag.

9. Tras consultar la versión del núcleo con uname -a, se buscan exploits aplicables a nivel usuario en la máquina vulnerable. Se encuentra uno de escalada local de privilegios llamado overlayfs.

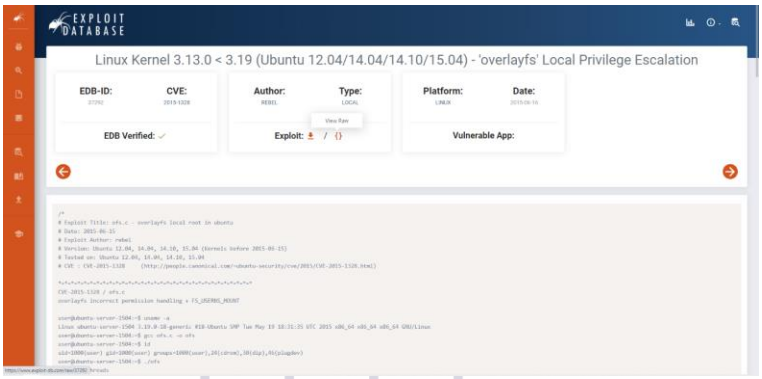

Ilustración 42: Consulta de exploit.

10. Se descarga el código del exploit y se compila con GCC.

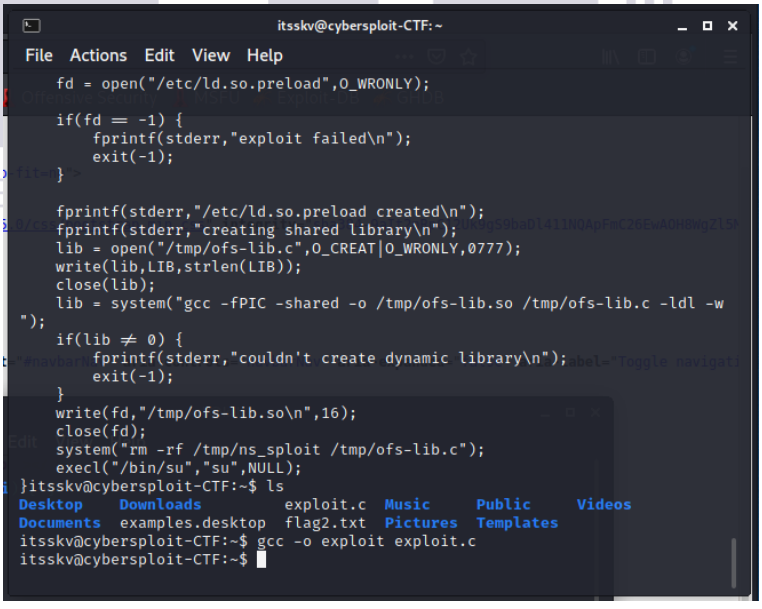

Ilustración 43: Descarga y compilación del exploit.

11. Se ejecuta el exploit y se comprueba que se ha obtenido acceso a root. Además, se obtiene la bandera final.

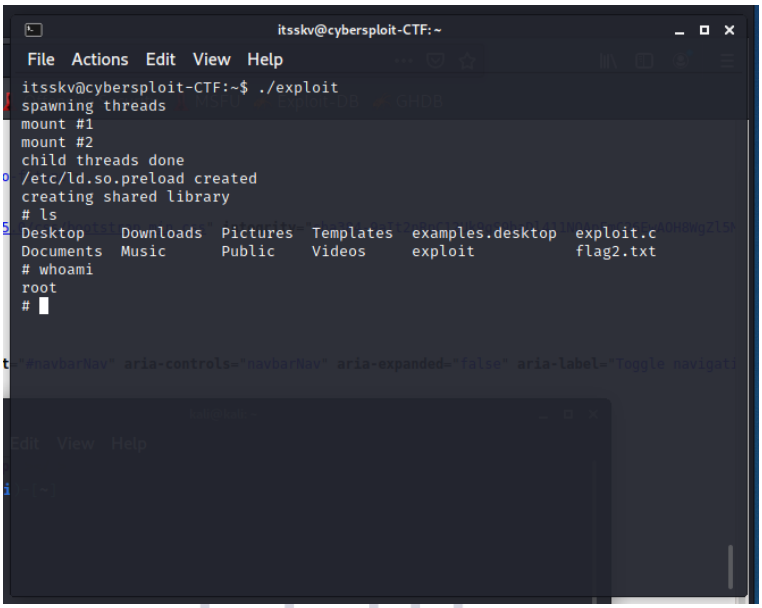

Ilustración 44: Ejecución del exploit.

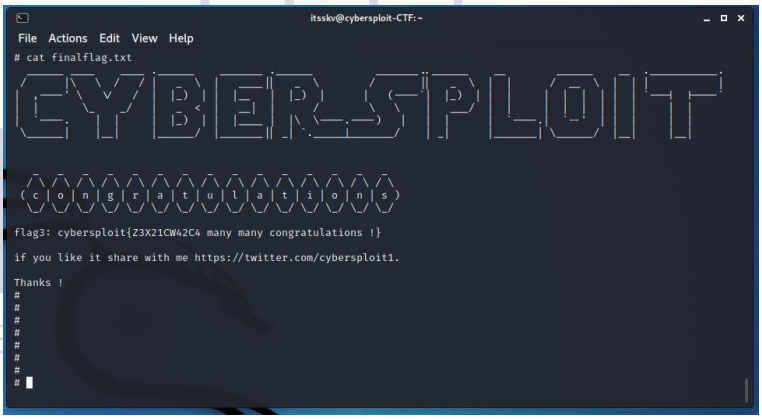

Ilustración 45: Accesso root y Flag de root.

Pwned-1

1. Se busca la IP de la máquina utilizando el comando netdiscover. En este caso, la máquina vulnerable es el equipo 192.168.56.103.

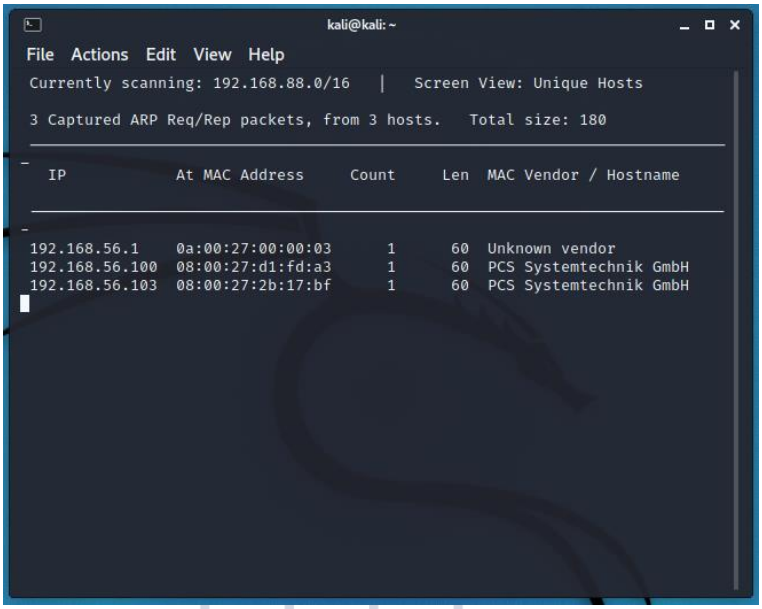

Ilustración 46: Pwned-1 comando netdiscover.

2. Se utiliza el commando nmap para buscar los puertos abiertos que tiene la máquina vulnerable. En este caso, tiene abiertos los puertos 80 (servidores HTTP), 21 (servidores FTP) y 22 (servidores SSH).

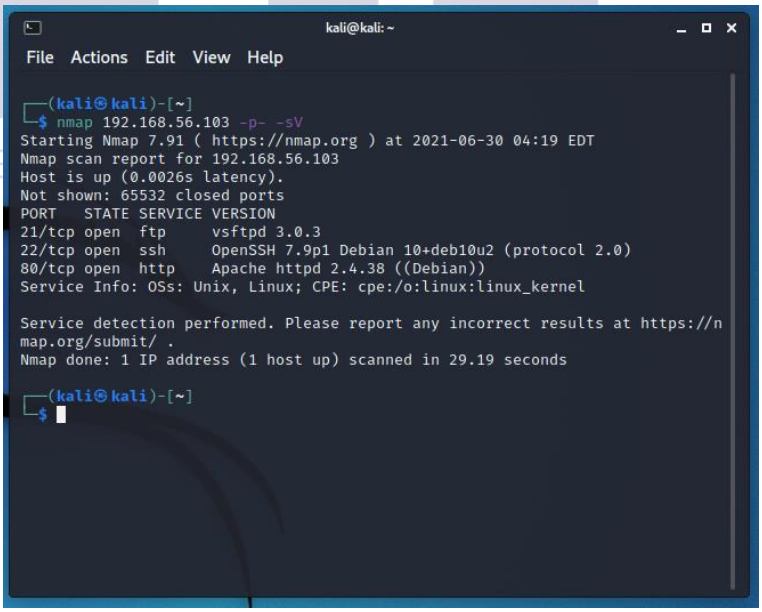

Ilustración 47: Pwned-1 Comando nmap.

3. Como tiene el puerto 80 abierto, que es el asociado con los servidores HTTP, se intenta entrar en la IP desde un navegador. A simple vista no parece haber nada útil en esta página. Más adelante se comprobará si hay información valiosa en el código de fuente.

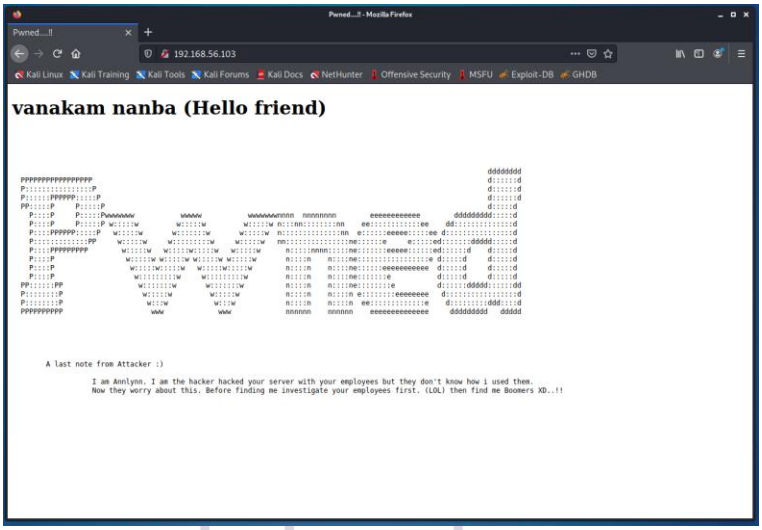

Ilustración 48: Pwned-1 exploración de la web.

4. Se buscan las páginas que se sirven con HTTP desde la IP de la máquina vulnerable utilizando el comando dirb.

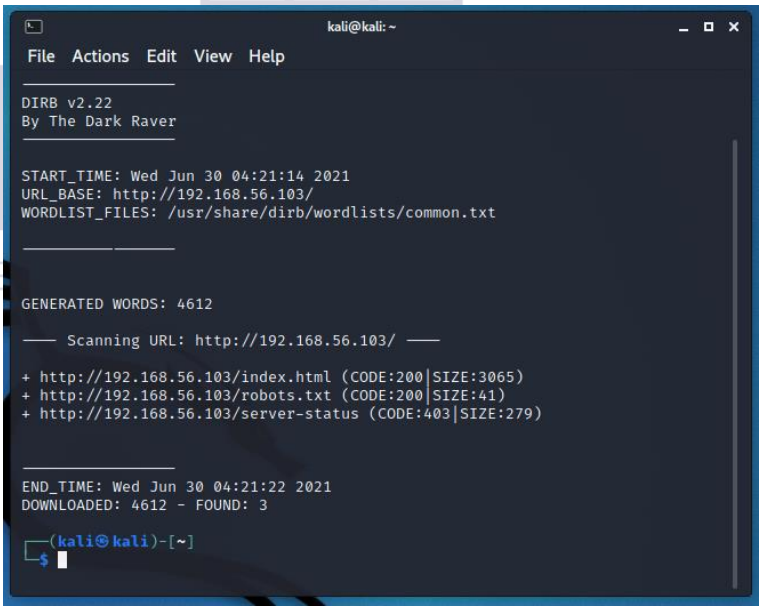

Ilustración 49: Pwned-1 Comando dirb.

Tras probar las distintas opciones, no se encuentra nada sospechoso y se utiliza un analizador más exhaustivo, dirbuster, que proporciona los siguientes resultados tras dos horas de ejecución:

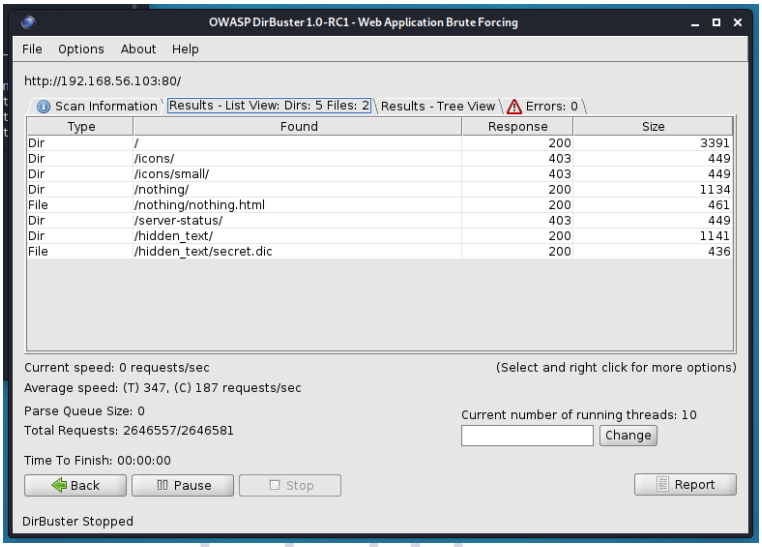

Ilustración 50: Pwned-1 Birbuster.

El fichero /hidden_text/secret.dic tiene un nombre sospechoso, así que, se explora su contenido que resulta ser una lista de directorios:

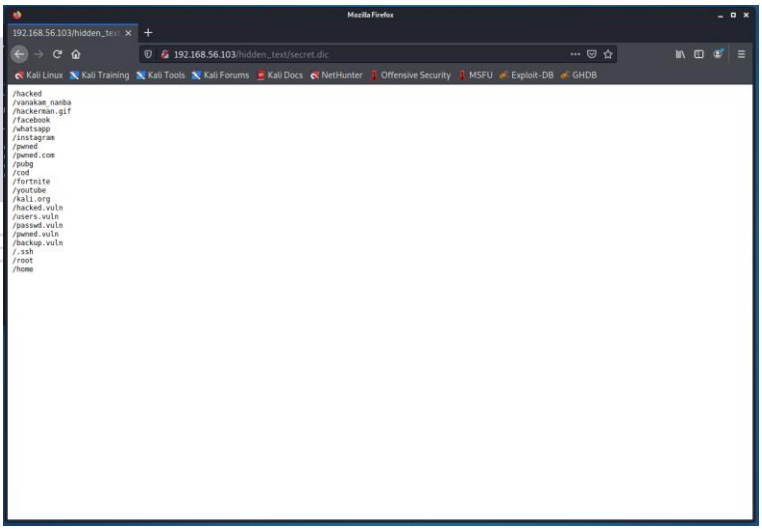

Ilustración 51: Pwned-1 exploración de directorios.

Si se consultan todos estos directorios, se llega a /pwned.vuln:

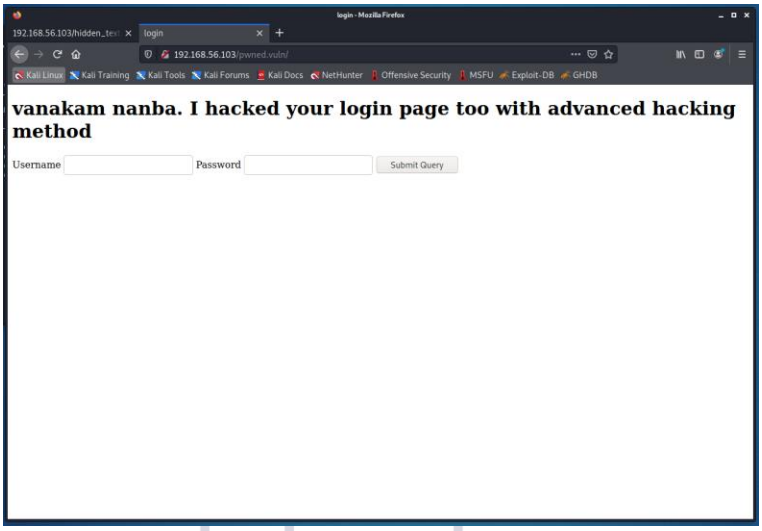

Ilustración 52: Pwned-1 Página de login.

5. Se analiza el código de fuente de la página /pwned.vuln:

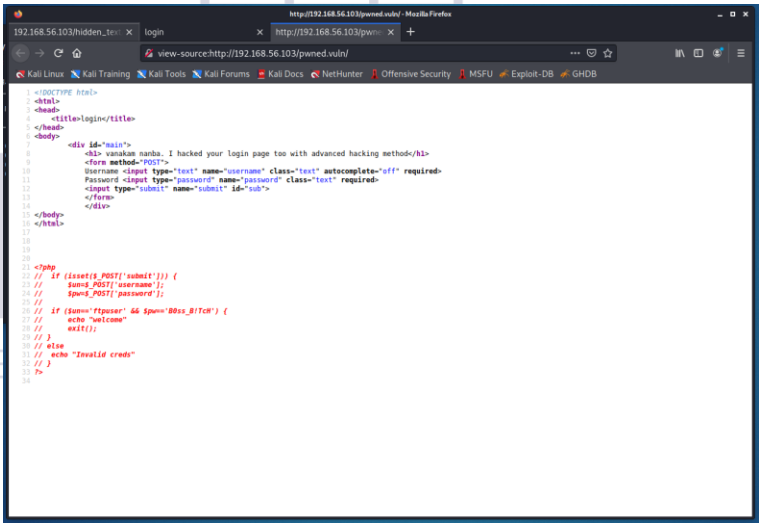

Ilustración 53: Pwned-1 Código fuente.

En esta, se puede encontrar un nombre de usuario “ftpuser” y una contraseña “B0ss_B!TcH”.

6. Se utiliza el comando ftp para acceder al servidor de ficheros utilizando el usuario y la contraseña encontrados:

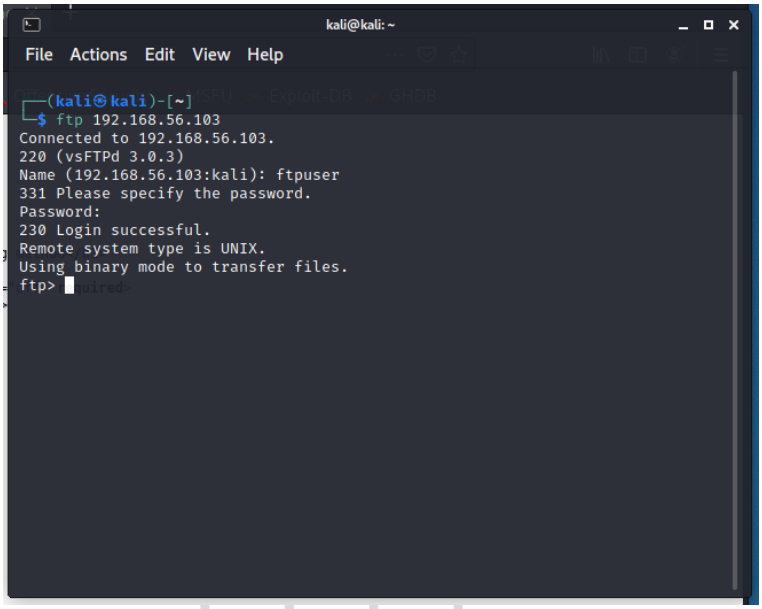

Ilustración 54: Pwned-1 Commando ftp.

Una vez dentro, se comprueban los ficheros existentes y se recuperan los que pueden ser potencialmente útiles.

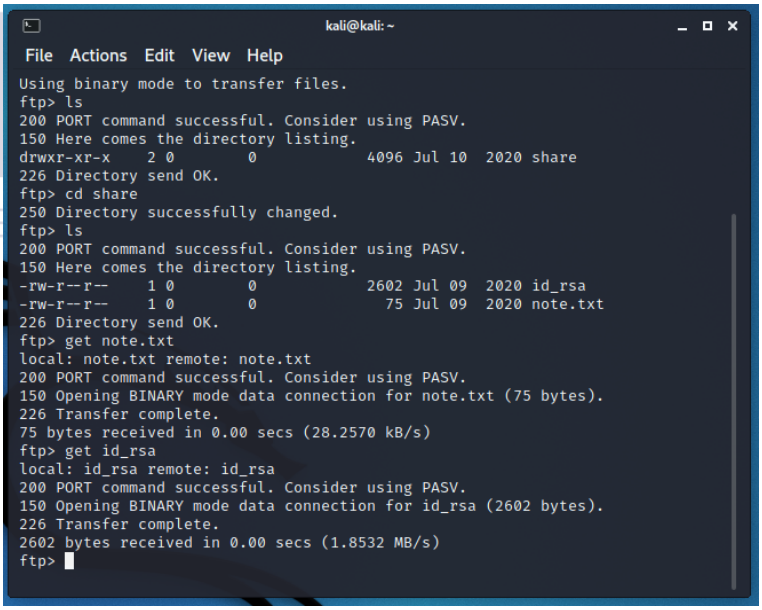

Ilustración 55: Pwned-1 Exploración ftp.

Los dos ficheros recuperados fueron note.txt e id_rsa, cuyo contenido se puede ver a continuación.

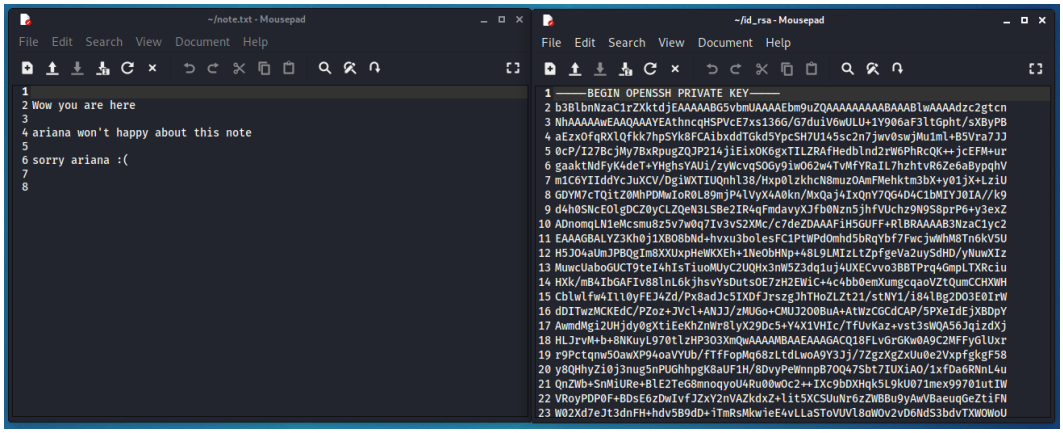

Ilustración 56: Pwned-1 Usuario y clave privada.

De note.txt se puede deducir la existencia de un usuario llamado ariana y id_rsa es un fichero de clave privada.

7. Se accede a la máquina con SSH utilizando el usuario encontrado y el fichero de clave privada.

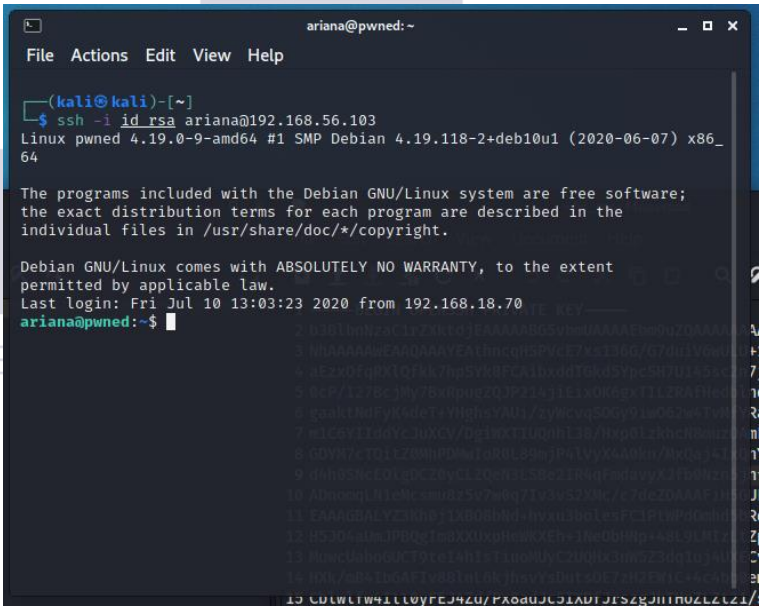

Ilustración 57: Pwned-1 Accesso ssh.

8. Si se hace un ls, se puede ver el fichero “user1.txt” con la primera bandera:

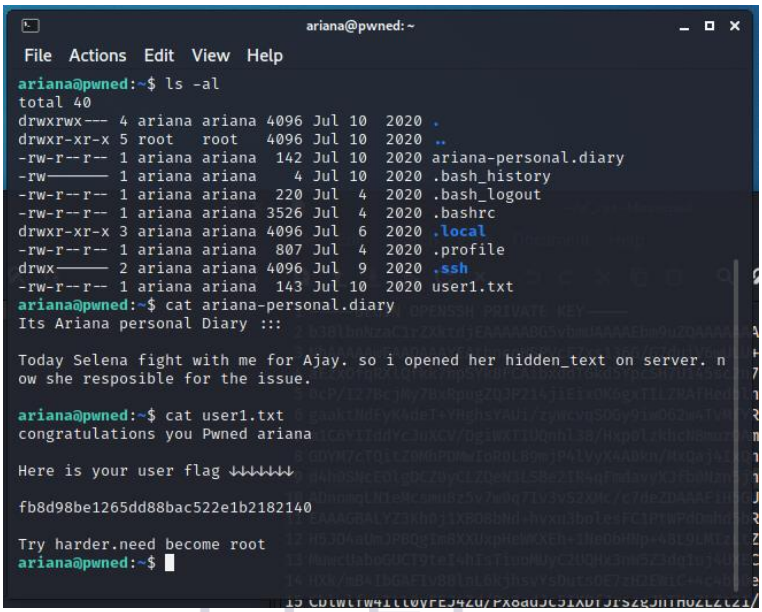

Ilustración 58: Pwned-1 Primera Bandera.

9. Se comprueban comandos que típicamente pueden dar acceso al usuario root. Al probar el comando sudo, se obtiene el siguiente resultado:

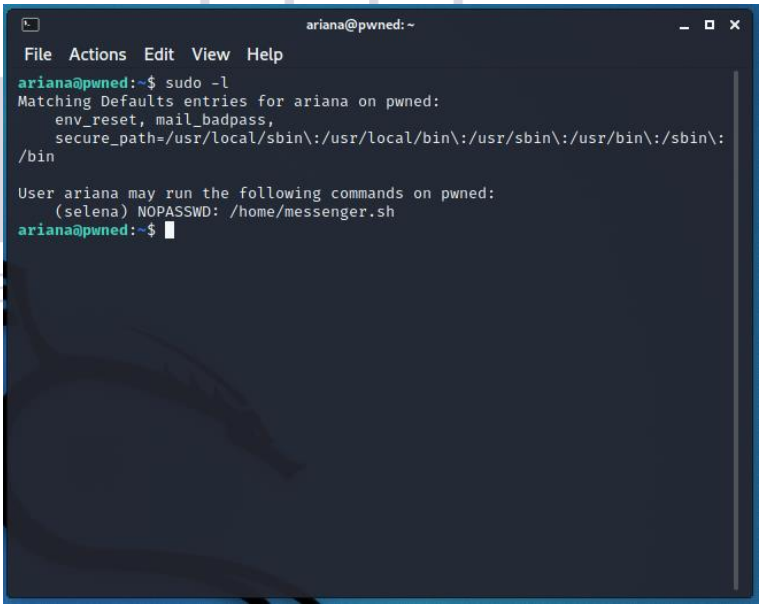

Ilustración 59: Pwned-1 análisis de comandos con permisos de root.

Se tiene acceso a ejecutar el script /home/messenger.sh como el usuario selena.

10. Se analiza el script mencionado:

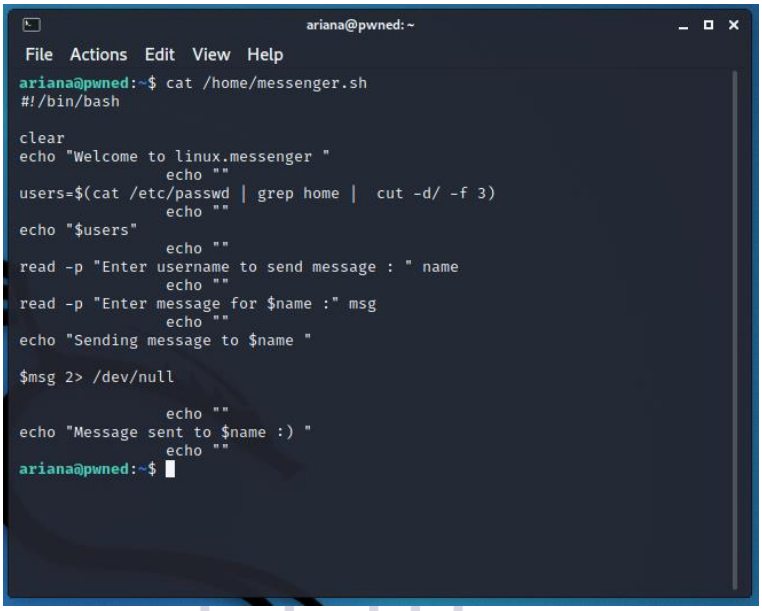

Ilustración 60: Pwned-1 script messenger.sh

Este fichero ejecuta el texto que se le introduce como mensaje, por lo tanto, se puede utilizar para ejecutar un comando arbitrario como selena.

11. Se ejecuta el script /home/messenger.sh como selena usando sudo.

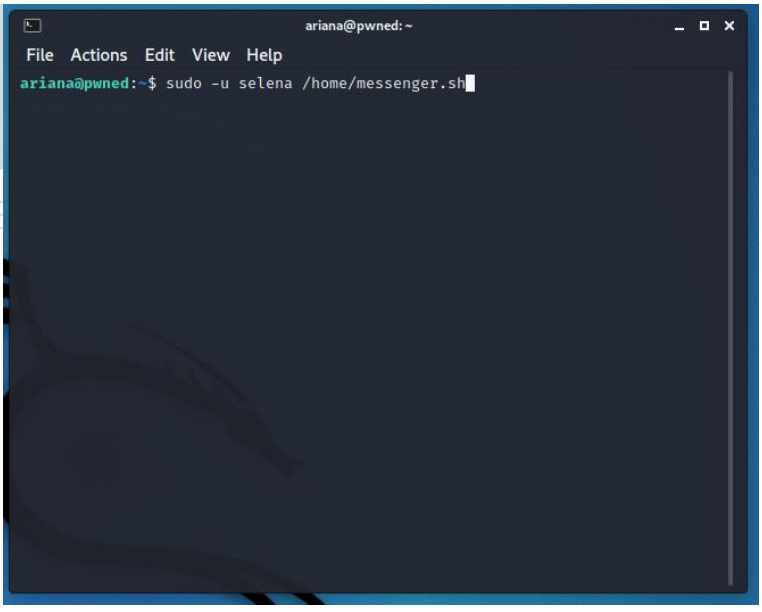

Ilustración 61: Pwned-1 ejecución del script messeger con usuario selena.

Una vez ejecutado, se escribe “bash” como mensaje, lo que permitirá acceso a una terminal como el usuario selena. Dentro de esta, se obtiene la segunda bandera y se comprueban los

grupos a los que pertenece Selena. Curiosamente, uno de ellos es “docker”. Gracias a esto, se podrá obtener una root shell.

12. Se obtiene una root shell por medio de Docker y se obtiene la última bandera:

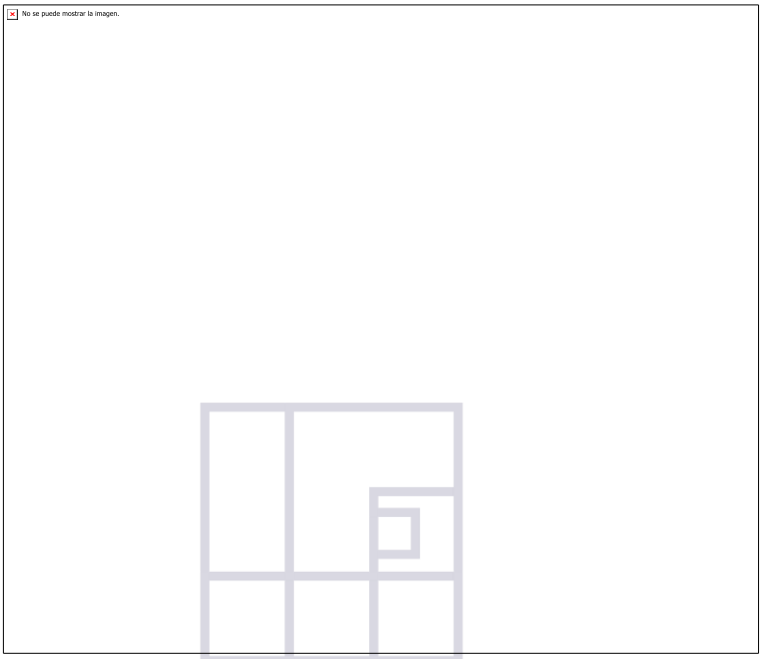

Ilustración 62: : Pwned-1 accesso root.

CONCLUSIONES

Una vez finalizado este proyecto, se puede concluir que se ha puesto en marcha una herramienta que permite utilizar distintas técnicas aplicables al ámbito de la administración de sistemas, más concretamente, la parte relacionada con el Hacking Ético. De esta forma, la herramienta permite poner en práctica los conocimientos teórico-prácticos adquiridos en las asignaturas ya descritas al principio. El sistema se compone de una serie de máquinas con ciertas vulnerabilidades que permitan a cada usuario intentar hacerse con los privilegios de root de la máquina. La posibilidad de interactuar y hacer los ejercicios de una forma más novedosa hace que aprendan sin darse cuenta lo que rebaja la dificultad de aprendizaje. El desarrollo de este proyecto se ha realizado con tecnología open-source de forma que su implantación por parte de la Universidad permita ahorrarse costosas licencias en software. Los profesores pueden utilizar este sistema de una forma ágil y sencilla, permitiendo que sus clases teóricas obtengan un mayor interés al mostrar que el conocimiento adquirido es aplicable en ciertas situaciones. Todo el código utilizado en el desarrollo de este proyecto ha sido publicado en un repositorio para que la comunidad académica pueda hacer uso de esta solución.