TRABAJO FIN DE GRADO

Grado en Ingeniería Informática


UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA


# Técnicas contra el ciberdelito: Servicios ocultos controlados en la red TOR


Curso 2020/2021


**Alumna:**

Ana María Vargas Tobías

**Director:**

José Antonio Álvarez Bermejo

# Índice

## Agradecimientos

Quiero aprovechar este apartado para agradecer a mi director de proyecto por haberme ayudado y guiado a lo largo del desarrollo de este trabajo. Igualmente, también me gustaría a los miembros del curso ADWI de ECTEG por permitirme participar en su proyecto.

## Resumen

Es una verdad ampliamente conocida que *dark nets* como TOR ofrecen a los ciberdelincuentes una gran oportunidad para poseer privacidad y anonimato a la hora de realizar actividades ilegales. La dificultad de demostrar la realización de este tipo de delitos, los cuales han ido incrementando a lo largo de estos últimos años, e identificar a los culpables, sumado a la falta de herramientas para operar en este tipo de redes anónimas, se ha convertido en una amenaza para las fuerzas y cuerpos de seguridad encargados de combatirlos. Por lo tanto, en este proyecto se desarrollarán pruebas de concepto donde se aplicarán técnicas que se pueden utilizar en estos entornos y que permiten adaptar procedimientos y estrategias utilizadas con frecuencia en la *surface web.*

Palabras clave: *The Onion Routing* (TOR), servicios ocultos (*hidden services*), clonación.

## *Abstract*

*It is a truth universally acknowledged that dark nets such as TOR offer a great opportunity to obtain privacy and anonymity for cybercriminals while committing illegal activities. The difficulties presented at proving these types of crimes, which have been increasing over the past years, and identifying the culprits, all added to the lack of specialized tools for these types of anonymous networks, have become a threat for law enforcements agents in charge of fighting them. Therefore, in this project some proofs of concept will be developed where different techniques could be used in these environments which allow adapting procedures and strategies frequently used in the surface web.*


*Key words: The Onion Router (TOR), hidden services, cloning.*

## Abreviaturas

Las abreviaturas que se van a utilizar a lo largo de todo el proyecto se pueden apreciar en la tabla 1.

| Abreviatura | Denominación original |
|---|---|
| **TOR** | *The Onion Router* |
| **OS** | *Operating System* |
| **DNS** | *Domain Name System* |
| **HTTP** | *Hypertext Transfer Protocol* |
| **HTTPS** | *Hypertext Transfer Protocol Secure* |
| **FTP** | *File Transfer Protocol* |
| **BEEF** | *The Browser Exploitation Framework* |
| **JS** | *JavaScript* |
| **v3** | Versión 3 |
| **v2** | Versión 2 |
| **DVWA** | *Damn Vulnerable Web Application* |
| **SQLi** | *SQL injection* |
| **XSS** | *Cross-Site Scripting* |
| **WWW** | *World Wide Web* |
| **HSDir** | *Hidden Service Directory* |
| **FDM** | *Fallback Directory Mirror* |
| **TCP** | *Transmission Control Protocol* |
| **HSD** | *Hidden Service Descriptor* |
| **IP** | *Introduction Point* |
| **RP** | *Rendezvous Point* |
| **DHT** | *Distributed Hash Table* |
| **SSH** | *Secure Shell* |
| **SMTP** | *Simple Mail Transfer Protocol* |

*Tabla 1. Abreviaturas utilizadas en este documento.*

# Listado de figuras

# Capítulo 1 – Introducción

## Aclaración previa

Antes de comenzar con el desarrollo de este proyecto, se debe resaltar que la finalidad última de este proyecto es didáctica. Las técnicas utilizadas tienen como objetivo distintos tipos de ciberdelitos y de ciberdelincuentes y están enfocadas a ser utilizadas por individuos con autoridad para ello. En ningún caso fueron pensadas para infringir la ley y de su uso depende la responsabilidad de cada uno.

## Motivación del trabajo de fin de grado

Hoy en día la falta de privacidad en la *surface web* es el motivo fundamental por el que muchas personas opten por una alternativa anónima como las *dark nets* (Saleh, Qadir, & Ilyas, 2018), entre ellas la red TOR (*The Onion Router*) (Filiol, Delong, & Nicolas, 2019), sobre la que se va a centrar este proyecto ya que es la más ampliamente utilizada, para conseguir ocultar su verdadera identidad. Los perfiles de estas personas pueden ser muy variados, desde particulares que desean una mayor privacidad para manifestar sus opiniones hasta periodistas que buscan actuar de incógnito para publicar noticias comprometidas desde países donde la censura es la norma.

Por tanto, redes anónimas como TOR han contribuido positivamente a potenciar la privacidad y el anonimato de los que no se disponen en la *surface web.* No obstante, ello no ha impedido que, aunque no fuera el uso para el fueron concebidas, estas *dark nets* se utilicen para realizar actividades ilegales aprovechando las características mencionadas anteriormente. Debido a las características particulares de las *dark nets,* la forma de identificar a aquellos que cometen el delito amparándose en el anonimato difiere de los modos empleados para la *Surface web,* lo cual dificulta la labor de las fuerzas y cuerpos de seguridad al enfrentarse a este tipo de amenazas (Al-Nabki, Fidalgo, Alegre, & Fernández-Robles, 2019).

Por consiguiente, no existen demasiadas estrategias para combatir los ciberdelitos que se amparan en TOR comparado con las existentes en la *surface web*. No obstante, sí que se pueden aprovechar dichas técnicas de ataque contra el ciberdelito utilizadas frecuentemente en la *surface web* para su posterior aplicación en TOR, entre otras, para poder identificar a tanto a quien pretende cometer un ciberdelito como a los servicios online (Bernaschi, Celestini, Guarino, & Lombardi, 2017) que le permiten cometerlo. Los procedimientos para realizar estos ataques se pueden adaptar para aplicarlo sobre la red TOR cuyo objetivo sería poder obtener información del criminal e, idealmente, identificarlo y localizarlo.

## Objetivos

El objetivo principal de este proyecto es detectar e identificar actores que realizan actividades ilegales en la red TOR (Alsabah & Goldberg, 2016). Para ello, se proponen dos técnicas aplicables en TOR (Echeverri, 2018). Desde entonces los servicios ocultos han evolucionado y se pretende rediseñar las técnicas para los *Hidden Services 3.0.*

La primera, consiste en un procedimiento para utilizar cualquier herramienta de *pentesting* habitual contra un servicio oculto creando un *proxy* que conecte automáticamente el tráfico enviado por una máquina local con dicho servicio oculto. De esta manera, las herramientas de *pentesting* solo tendrían que dirigir su ataque por el puerto que el *proxy* haya configurado.

Por otro lado, se propone como segunda técnica duplicar servicios ocultos (Steinebach, Zenglein, & Brandl, 2021) para interceptar y actuar sobre quienes realizan actividades sospechosas. Una vez duplicado el servicio y engañado al delincuente mediante la aplicación previa de ingeniería social, se podrían ejecutar rutinas maliciosas en el lado del cliente mediante código JS (*JavaScript)* modificando el contenido de esos servicios con el objeto de poder controlar y, en su caso, lanzar diversos tipos de ataques a los usuarios que acceden a ese servicio ilegal.

Las posibilidades pre-explotación (conseguir que los delincuentes accedan al servicio clonado) y post-explotación (los diferentes ataques que se pueden lanzar contra estos usuarios una vez explotados) son muy variadas, por lo que esta técnica se centrará en la etapa de explotación, es decir, conseguir llegar a monitorizar a los usuarios mientras realizan actividades sospechosas no siendo ellos conscientes de estar siendo atacados.

Por otra parte, es necesario resaltar que, dependiendo del tipo de sitio que se desee suplantar, el artículo 18 de la Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal (Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal. , 1995) puede considerar las técnicas desarrolladas en este proyecto como provocación y, en su caso, inducción al delito. No obstante, la finalidad de este proyecto es puramente didáctica y todo el contenido estará explicado sobre un punto de vista técnico. Por tanto, en la prueba de concepto incluida en este proyecto no se clonará ningún sitio que pueda dar lugar a esta interpretación por el Código Penal.

## Planificación temporal y estructura del proyecto

Para poder aplicar las técnicas mencionadas anteriormente es necesario conocer en profundidad el funcionamiento del protocolo TOR, las características principales de dicha red y tanto sus similitudes como sus diferencias con la *surface web*, especialmente lo relativo a sitios y servicios web.

Por este motivo, en primer lugar, se destinará la primera parte del trabajo a realizar un desarrollo teórico sobre estos conceptos, detallando los puntos más importantes y desarrollando su contenido para poder entender la posterior realización práctica.

Asimismo, también es necesario entender cómo funcionan las herramientas utilizadas durante la realización de este proyecto, por lo que después de la explicación teórica se analizarán brevemente las principales herramientas que hacen posible el desarrollo de las técnicas posteriormente utilizadas.

A continuación, se dará paso a la fase más importante del proyecto, es decir, se procederá a realizar dos pruebas de concepto donde en la primera se procederá a atacar un servicio oculto en TOR vulnerable previamente levantado y en la segunda se clonará el contenido de un servicio (oculto) web donde se modificará el contenido del sitio creado para inyectar código malicioso en el lado del cliente (usuario susceptible de realizar actividades sospechosas). Asimismo, para impedir que está técnica pueda ser contraatacada, se realizará unos procesos de *hardening* del servicio oculto para que sea lo más seguro posible mientras este expuesto en TOR. Con estas pruebas de concepto se enseñará una solución eficaz para monitorear las actividades realizadas en la red TOR.

Finalmente, se resaltarán las conclusiones que se han ido alcanzando a lo largo de las etapas de realización de este proyecto, así como una serie de propuestas con las que poder expandir el contenido explicado y otras alternativas para poder alcanzar los mismos resultados.

Igualmente, se mostrará un diagrama de tareas con la planificación planteada para desarrollar este proyecto en la figura 1.1.

| *Duración del proyecto: 300 horas* | | *Semana* | | | | Desde el 5/10/2020 hasta 2/5/2021 en semanas (Media de horas por semana: 10 horas) | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nº Tarea | Nombre de tarea | Inicio | Fin | Semanas totales | Horas totales | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| **1** | **Investigación** | **1** | **9** | **9** | **90** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1.1 | Búsqueda de referencias | 1 | 3 | 3 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1.2 | Lectura de información | 4 | 9 | 6 | 60 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **2** | **Desarrollo anteproyecto** | **10** | **12** | **3** | **30** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **3** | **Prueba de concepto** | **13** | **25** | **13** | **130** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3.1 | Preparación laboratorio | 13 | 17 | 5 | 50 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3.2 | Fase de explotación | 18 | 25 | 8 | 80 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **4** | **Desarrollo teórico** | **26** | **28** | **3** | **30** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4.1 | Principales conceptos | 26 | 27 | 2 | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4.2 | Herramientas utilizadas | 28 | 28 | 1 | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **5** | **Revisión final** | **29** | **30** | **2** | **20** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

*Figura 1.1. Diagrama de Gantt con la planificación planteada para el desarrollo de este proyecto*

# Capítulo 2 - Desarrollo teórico

Para poder comprender el desarrollo de este proyecto es necesario entender los conceptos que se explicarán a continuación.

## *Dark nets*

En primer lugar, es necesario comprender las diferencias entre conceptos como *surface web, deep web, dark nets* o *dark web* debido a que algunos de ellos, en ocasiones, se utilizan sin distinción, o bien se tratan como independientes.

El término *deep web* se refiere a todo el contenido web de la WWW que no se encuentra indexado por motores de búsqueda convencionales (ej. Google, Bing, etc.), en cambio, la *surface web* es todo aquello que sí se puede encontrar haciendo uso de estos navegadores. En numerosas ocasiones se hace uso del término *deep web* para hacer referencia exclusivamente al conjunto de *dark nets,* no obstante, si se utiliza de base la definición anterior, todo el contenido que no se pueda encontrar a través de navegadores convencionales se consideraría parte de la *deep web*. Esto podría incluir, por ejemplo, los contenidos enviados a un correo electrónico o los archivos guardados en *Dropbox* o *Drive.*

Por otro lado, las *dark nets* son redes anónimas donde los usuarios pueden conectarse unos con otros sin tener que revelar su identidad o su localización (Erdin, Zachor, & Hunes, 2015). Estas redes pueden dividirse en dos grupos: Sistemas de alta latencia y sistemas de baja latencia. Los sistemas de alta latencia en general se basan en el uso de múltiples *proxies* entre el emisor y el receptor para principalmente evitar ataques de análisis de tráfico donde un individuo intenta relacionar al emisor con el receptor. Por esta razón, los sistemas de alta latencia ofrecen un gran nivel de anonimato, pero con un tiempo de espera muy alto por lo que no es muy adecuado para actividades que requieran altos niveles de interacción. Debido a esto, los sistemas de baja latencia son más utilizados ya que utilizan protocolos donde el tiempo de espera es mucho menor. Actualmente, las *dark nets* de baja latencia más famosas son TOR e I2P. Asimismo, también es interesante señalar que las *dark nets* no solo incluyen contenido *web* (*dark* web), es decir, que no solo trabajan con el protocolo *http,* sino que también pueden incluir otros protocolos como SSH, FTP, SMTP, etc.

## TOR

Como se ha resaltado previamente, en este proyecto se trabajará con TOR por lo que es conveniente conocer su estructura y funcionamiento. El funcionamiento de TOR se basa en un número de usuarios que utilizan un protocolo específico. La red de TOR, que trabaja exclusivamente por el protocolo TCP, está formada por miles de *onion routers* (o *relays*) operados por voluntarios alrededor del mundo. La gran mayoría de estos *relays* son conocidos de forma pública, pero hay un pequeño porcentaje de ellos que no están expuestos al público (*bridges*), los cuales sirven para evitar que una autoridad censure la entrada a TOR bloqueando el acceso a estos *relays.*

Nueve de estos *routers* son autoridades de directorio que controlan y configuran la red TOR. En ellos están clasificados el resto de *routers,* tanto *relays* como *bridges,* de la red y asignan a cada uno un peso en función de su ancho de banda, su antigüedad o su estabilidad y, además, mantienen y firman cada hora documento consensuado para que todos los clientes que estén utilizando la red TOR tengan la misma información sobre los *relays* que la forman (How Does Tor Really Work? The Definitive Visual Guide (2020), 2020).

Otro grupo de *routers* muy importante son los HSDir (*Hidden Service Directory*) que funcionan como una especie de servidor DNS en TOR ya que contienen la información para comunicarse con un servicio oculto a partir de su dirección *onion* en una estructura DHT (*Distributed Hash Table*) (Owenson, Cortes, & Lewman, 2018). Junto con los dos grupos anteriores, los directorios de caché, que descargan los datos de las autoridades de directorio para que los clientes no tengan que acudir a ellos cada vez que quieran consultar las listas de *routers,* y los *FDM (Fallback Directory Mirror),* que permiten a los clientes descargarse el documento consensuado durante la primera conexión, forman las autoridades que gestionan la red TOR completa.

El resto de *relays* se pueden dividir en las siguientes categorías:

- Nodo de entrada: Es primer *relay* con el que se comunica un cliente para acceder a la red TOR.
- Nodo de salida: Es el último *relay* por el que pasan el tráfico de un cliente antes de salir de la red TOR.
- Nodo medio: Es un *relay* que solo se comunica con otros y, por lo tanto, solo opera con el tráfico dentro de TOR.

Para mantener las comunicaciones anónimas, TOR construye circuitos compuestos de tres nodos que se sitúan entre el origen y el destino en el que cada uno solo conoce las direcciones de aquellos con los que mantiene comunicación directa. Dichos nodos se eligen pseudo (porque no todos los nodos tienen la misma probabilidad de ser elegidos) aleatoriamente. Cada nodo le da al emisor una clave para cifrar el mensaje (para este traspaso, se utiliza encriptación asimétrica y el intercambio de claves *Diffie-Hellman*). Así el emisor cifra primero el mensaje con la clave del tercer nodo, después con la del segundo nodo y, por último, con la del tercer nodo. Por tanto, cuando el mensaje pasa por el primer nodo, este utiliza su clave para descifrarlo, por lo que el mensaje sigue cifrado con la clave del segundo y del tercer nodo. Una vez que el mensaje llega al segundo nodo, este también utiliza su clave para descifrarlo y el mensaje continúa cifrado con la clave del tercer nodo. Posteriormente, cuando llega al último nodo, este termina de descifrar completamente el mensaje con su clave. De esta manera el mensaje en claro llega al receptor. En la figura 2.1 se puede apreciar gráficamente esta explicación.

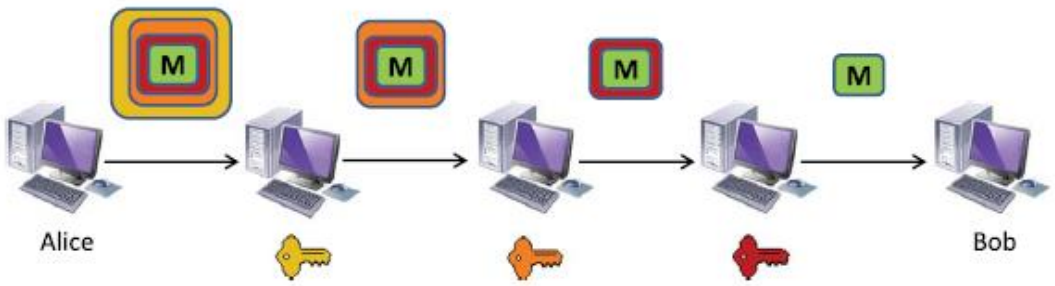


*Figura 2.1. Creación de un circuito en TOR.*

Dependiendo de su uso, TOR puede funcionar en modo *out proxy* y en modo *in proxy.* El modo *out proxy* sirve para utilizar la infraestructura de la red TOR como un *proxy* para navegar por la *surface web* de forma anónima. En cambio, el modo *in proxy* usa para acceder a servicios internamente dentro de la propia red TOR (los servicios ocultos).

Asimismo, el funcionamiento de TOR en modo *out proxy* es bastante sencillo ya que sólo se basa en la creación de un circuito para enrutar el tráfico por él antes de salir a Internet tal y como se puede observar en la figura 2.2.

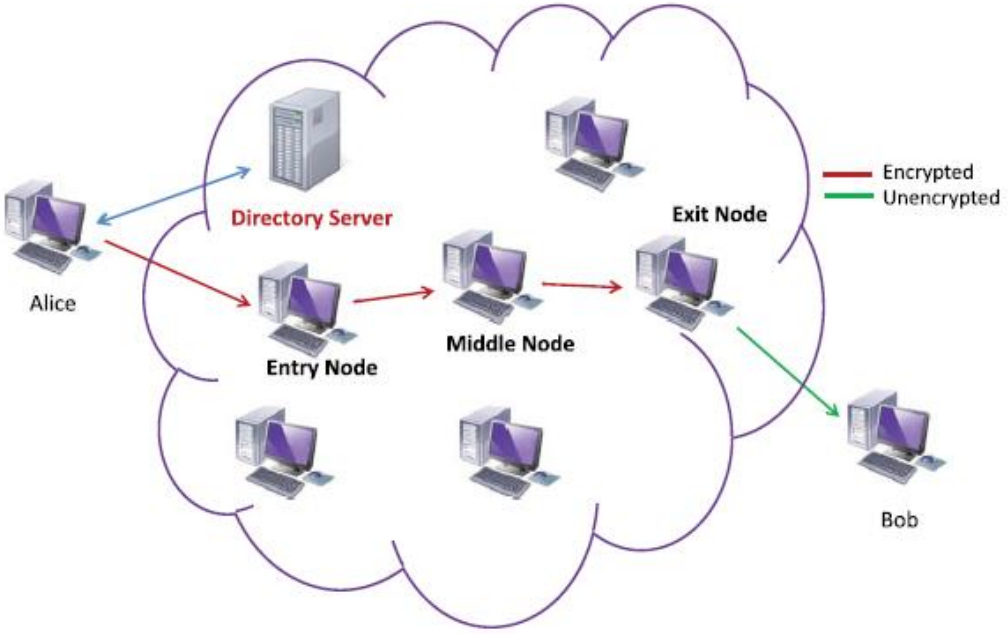


*Figura 2.2. Funcionamiento de TOR en modo out proxy.*

El funcionamiento del modo *in proxy* es bastante más complejo ya que su objetivo es que el cliente y el servicio oculto sean anónimos entre ellos.

En primer lugar, cuando se crea un servicio oculto, TOR genera un HSD (*Hidden Service Descriptor*) que contiene la información necesaria para acceder a dicho servicio a través de su dirección *onion* y lo registra en un HSDir utilizando un circuito de tres nodos.

El cliente que quiere acceder a ese servicio oculto y conoce su dirección *onion,* solicita (a través de un circuito de tres nodos) al HSDir, el HSD del servicio oculto que posea dicha dirección *onion*. Con ese HSD, el cliente conoce unos nodos llamados IP (*Introduction Point*) que han sido seleccionados por el servicio oculto y que están conectados con él a través de un circuito de tres nodos. Además, el cliente se conectará aleatoriamente con un circuito de tres nodos a un *relay* de la red TOR que actuará como RP (*Rendezvous Point*).

El RP generará una clave que transmitirá al cliente. Posteriormente, el cliente se comunica con un IP (utilizando también un circuito de tres nodos) y a través de ese IP le envía la clave generada por el RP y la dirección de dicho RP. Finalmente, el servicio oculto se conecta con el RP (también con un circuito de tres nodos) y le envía la clave recibida del cliente (que a su vez ha recibido del RP) a través del IP.

El RP permite a través de él, al recibir la misma clave que él emitió al cliente, la comunicación entre cliente y el servicio oculto. En la figura 2.3 se puede observar la comunicación descrita de manera gráfica.

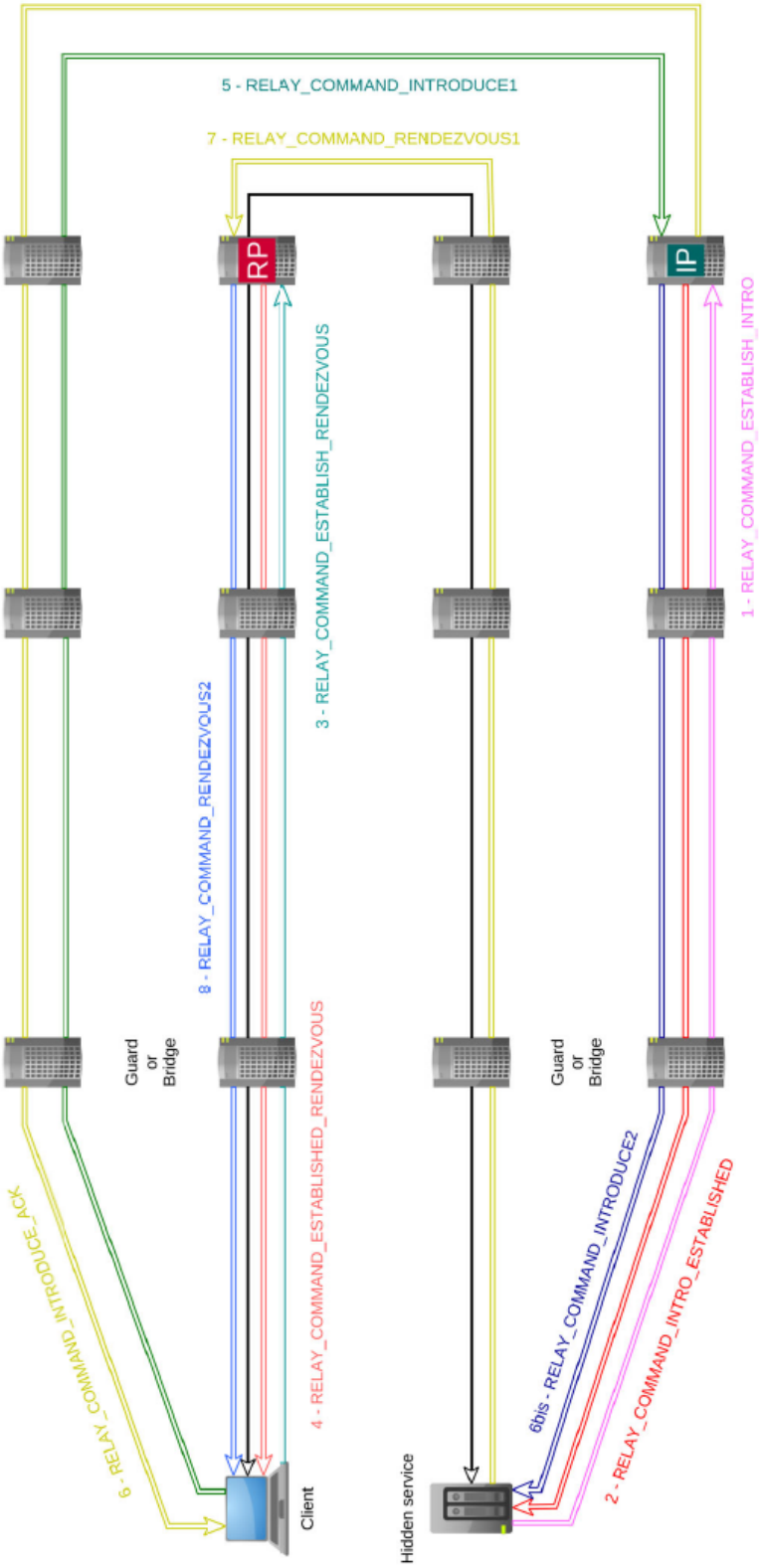


*Figura 2.3. Funcionamiento de TOR en modo in proxy.*

# Capítulo 3 - Materiales utilizados

A continuación, en la siguiente sección se describirán las distintas herramientas utilizadas para poder llevar a cabo este proyecto y conseguir los resultados obtenidos.

## *Parrot OS*

Para poder levantar el laboratorio necesario para aplicar las propuestas citadas previamente, se utilizará la distribución *Parrot OS* (Parrot OS, 2020), el cual es una distribución GNU/Linux basada en Debian enfocada la seguridad y a la privacidad, por lo que incorpora numerosas herramientas para poder realizar acciones relacionas con la ciberseguridad, entre ellas, las que van a ser utilizadas a lo largo de este proyecto. Además, este sistema operativo simplifica en enrutamiento a través de TOR del tráfico que sale del dispositivo local.

## TOR *browser*

Este popular navegador basado en *Firefox* permite navegar de modo anónimo a través de Internet y con el que se puede conseguir el acceso a dominios *.onion.* A la hora de iniciar este navegador, este configura una conexión a la red TOR donde se oculta tanto la dirección ip como otros datos que podrían identificar el dispositivo. Este navegador se utilizará para navegar a las direcciones *onion* de los servicios levantados.

## *Anonsurf*

Esta es una de las muchas herramientas integradas en *Parrot OS*, la cual permite enrutar todo el tráfico del dispositivo a través de TOR y no solo el tráfico web como TOR *Browser*.

## *Torsocks*

*Torsocks* funciona de manera similar a *anonsurf* pero dirigido a una aplicación específica en vez de todo el tráfico del dispositivo. Se asegura de que las peticiones DNS se procesen correctamente y deniega el resto de tráfico que no sea el TCP de la aplicación que esté haciendo uso de él. Esta herramienta se utilizará principalmente para enrutar el tráfico de la consola de comandos a través de TOR.

## *Wget*

*Wget* es una herramienta de *software* libre que permite descargar archivos de los protocolos de Internet más utilizados como HTTP, HTTPS, FTP, etc. (GNU Wget, 2020). Esta herramienta presenta una gran variedad de funcionalidades, sin embargo, en este proyecto se utilizará para descargar el contenido de un sitio web en TOR para posteriormente clonar su contenido y obtener así una copia idéntica.

## *Metasploit*

*Metasploit* es un proyecto de código abierto que proporciona un *framework* para realizar actividades de *pentesting* y auditorías de seguridad, en particular, permite encontrar y explotar vulnerabilidades en otros sistemas. El funcionamiento de *metasploit* se basa principalmente en módulos. Un módulo es un código que permite automatizar el proceso de explotación. En este proyecto se integrará *metasploit* con la herramienta BEEF (que se mencionará posteriormente) para poder atacar al objetivo una vez ha sido vulnerado.

### BEEF

Al igual que *metasploit,* BEEF también es una herramienta de *pentesting* enfocada a navegadores web y ataques en lado del cliente. A través de un código malicioso en JS (*JavaScript*) inyectado en un sitio web, BEEF puede comprometer los navegadores de los clientes que acceden a dicho sitio.

### *Nginx*

*Nginx* es un servidor web que también sirve como *proxy* inverso y como un servidor *proxy* de correo electrónico (aunque en este proyecto se usará como servidor web). Es un *software* libre, ligero, de alto rendimiento y multiplataforma (Welcome to NGINX Wiki!, n.d.). Sin embargo, el motivo principal por el que se ha optado por hacer uso de este servidor es porque es uno de los recomendados para levantar un servidor web como servicio oculto en TOR (Configuring Onion Services for Tor, 2019).

### *Onionscan*

*Onionscan* es una herramienta de auditoría automática de código abierto enfocado a servicios en TOR. Con esta herramienta se puede comprobar de manera sencilla y rápida la seguridad de un servicio *onion* (Discovering the Dark Web, n.d.). Los pasos para instalar y configurar *onionscan* están recogidos en el anexo I de este informe.

### *mkp224o*

Esta herramienta sirve para crear direcciones *onion* v3 personalizadas, lo cual resulta bastante útil a la hora de clonar sitios web ya que se puede crear una dirección que se parezca al del sitio original. Es interesante tener en cuenta que los dominios *onion* v3 son de 57 caracteres por lo que, en general, solo se recuerdan los primeros caracteres. El funcionamiento de esta herramienta se basa en crear a base de fuerza bruta pares de claves hasta encontrar uno que genere una dirección *onion* que contenga el patrón establecido. Asimismo, se debe tener en consideración que, al utilizar fuerza bruta, el tiempo de generación de claves aumenta exponencialmente con respecto al número de caracteres especificados. Los pasos para instalar y configurar *mkp224o* están recogidos en el anexo II de este informe.

### DVWA

DVWA es una aplicación web que presenta multitud de vulnerabilidades que tiene un objetivo didáctico ya que ayuda a los auditores de seguridad a poner en práctica sus conocimientos en un entorno controlado. Las vulnerabilidades de esta aplicación varían en función de la dificultad especificada en la configuración, la cual, se utiliza para simular un servidor web real más o menos seguro. Con esta herramienta se demostrará que se pueden atacar servicios ocultos de la misma manera que los servicios encontrados en la *surface web.* Los pasos para instalar y configurar DVWA están recogidos en el anexo III de este informe.

### *Socat*

*Socat* es una utilidad que permite crear un canal bidireccional de transferencia de datos entre dos canales independientes (Amoany, 2020). En este proyecto, se utilizará *socat* para levantar un puerto en la máquina local y enrutar las comunicaciones entrantes por dicho puerto hacia la dirección *onion* especificada.

## Sqlmap

*Sqlmap* es una herramienta de código abierto cuya principal característica es la automatización del proceso de detectar vulnerabilidades del tipo *SQLi* y explotarlas para tomar el control de una base de datos. Esta herramienta se utilizará como ejemplo a la hora de atacar servicios en la red TOR.

# Capítulo 4 - Técnicas de *pentesting* sobre servicios en TOR

En este capítulo, se procederá a realizar el desarrollo práctico de dos técnicas que pueden utilizarse contra el ciberdelito en la red TOR. Estos ataques están ideados para aplicarse en servicios en la *surface web,* sin embargo, a lo largo de este capítulo, se explicará cómo adaptar los diferentes procedimientos para su aplicación en TOR. Con el éxito de estas técnicas, se podría obtener información valiosa del criminal e incluso su identificación.

## Primera técnica

Como se ha mencionado anteriormente, no existen demasiadas herramientas enfocadas específicamente a la red TOR. No obstante, es posible utilizar las mismas herramientas que se utilizarían para realizar *pentesting* en la *surface web*. Por tanto, si un servicio en TOR presenta vulnerabilidades, es posible explotar dichas vulnerabilidades de la misma forma que si ese servicio estuviera en la *surface web.* La clave reside en levantar un *proxy* que envíe el tráfico automáticamente en un puerto de la máquina local hacia un servicio oculto en la red TOR. De esta manera, se pueden utilizar las herramientas habituales contra ese puerto local y esas peticiones se redirigirán a través del *proxy* hacia el servicio oculto objetivo.

A continuación, se desarrollará una prueba de concepto de esta técnica donde se levantará un servicio oculto con un servidor DVWA con vulnerabilidades y se utilizará la herramienta *sqlmap* para explotar algunas de ellas.

En primer lugar, se configura la aplicación DVWA para levantar un servidor web y *mysql* en la máquina local (explicado con detalle en el anexo III). Una vez funcionando DVWA de forma local, se procede a crear los servicios ocultos en el archivo de configuración de TOR (*torrc*) (Tor Hidden Services) (Set up Your Onion Service, n.d.). Para ello, primero se deben crear los directorios donde se ubicará la dirección *onion* y el par de claves del servicio oculto que se generarán posteriormente. Asimismo, en el archivo */etc/tor/torrc* se introducen las líneas que se pueden observar en la figura 4.1. En dicha figura se pueden apreciar las líneas *HiddenServiceDir*, la cual especifica el directorio que contendrá el par de claves correspondiente dicho servicio junto con la dirección *onion* generada a partir de dichas claves, y las líneas *HiddenServicePort,* las cuales mapean los puertos virtuales (los que utilizarán los clientes de este servicio oculto) hacia los puertos sobre el que está corriendo el servicio en la máquina local. Por tanto, las directivas configuradas en la figura 4.1 establecen que las peticiones dirigidas al puerto 80 y al puerto 3306 del servicio oculto, se redirijan al puerto 80 y 3306 de la máquina local, respectivamente. Antes de poner en funcionamiento el servicio oculto, se debe cambiar los permisos del directorio especificado en *HiddenServiceDir* para que pueda ser accedido por TOR con el siguiente comando:

- *sudo chmod 700 /home/tfg/hidden_service_dvwa*

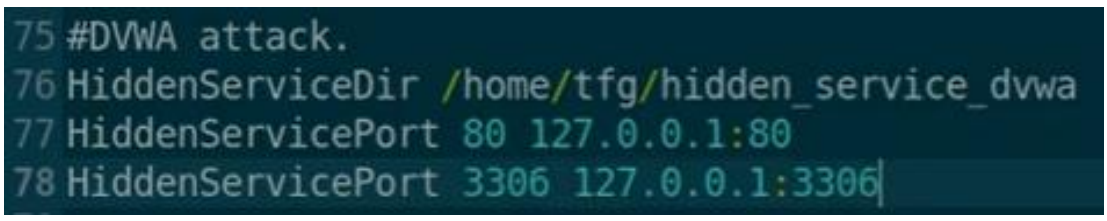


*Figura 4.1. Configurando el fichero de configuración de TOR para crear servicios ocultos.*

Para que el servicio oculto entre en funcionamiento y se cree el par de claves de dicho servicio se debe ejecutar TOR con el comando:

- *tor*

En la figura 4.2 se puede observar el inicio de la ejecución del comando especificado.

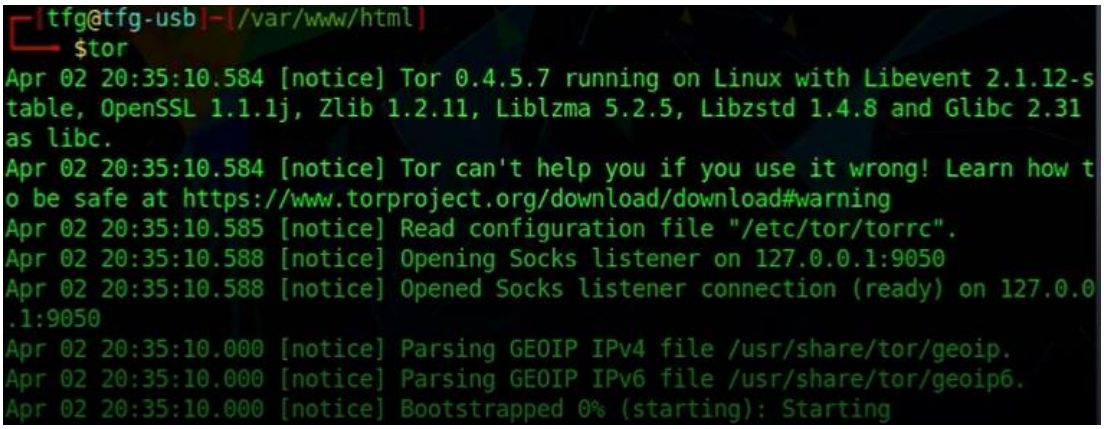


*Figura 4.2. Iniciando TOR.*

Una vez ejecutado TOR, utilizando TOR *Browser* se navega a la dirección *onion* localizada en el archivo *hostname* de la carpeta especificada en *HiddenServiceDir*, en este caso, la carpeta */home/tfg/hidden_service_dvwa.* Como se puede observar en la figura 4.3, en dicha dirección *onion,* en el subdirectorio */DVWA* se encuentra la aplicación *DVWA* configurada previamente. Asimismo, se puede iniciar sesión con el usuario *admin* y la contraseña *password.*

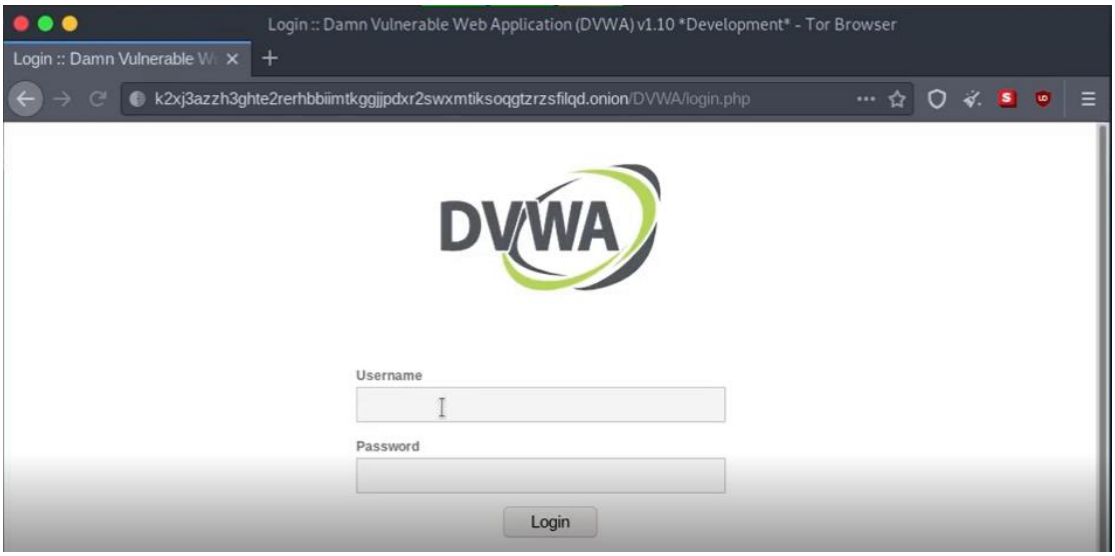


*Figura 4.3. DVWA como servicio oculto en TOR.*

A continuación, con el servicio oculto levantado y en funcionamiento, se procede a configurar un *proxy* en el puerto 9051 (el puerto que utiliza TOR) para crear un canal entre el puerto 7000 en la máquina local y la dirección *onion* en el puerto 80. Para ello, se utiliza *socat* de la manera mostrada en la figura 4.4.


*Figura 4.4. Creación del proxy entre la máquina local y el servicio oculto.*

Finalmente, ya se dispone de todo lo necesario para atacar al servicio oculto. Para esta prueba de concepto se utilizará la herramienta *sqlmap* para explotar una vulnerabilidad *SQLi* de la base de datos *MySQL* del servicio oculto. Para ello, en la sección *SQL Injection (Blind)* del sitio web del servicio oculto se introduce un número para obtener el enlace con el que va a trabajar *sqlmap*. En la figura 4.5, se obtiene el resultado de dicho paso.


*Figura 4.5. Obtención del enlace vulnerable.*

Si este servicio estuviera en la *surface web* para ejecutar *sqlmap* se tendría que utilizar el enlace a dicho sitio web por el puerto 80 (o el que se haya configurado en cada caso). No obstante, como ya se ha explicado previamente, en este caso *sqlmap* debe utilizar la dirección 127.0.0.1:7000, que es donde el *proxy* configurado anteriormente enrutará el tráfico hacia el servicio oculto. Por otro lado, las *cookies* del servicio oculto sí que se obtienen de la misma forma que cualquier otro servicio en la *surface web*, es decir, en la pestaña *Network* del inspector de TOR *Browser* se puede encontrar las *cookies* utilizadas en la cabecera de los paquetes. Por tanto, ya se dispone de la URL objetivo (URL compuesta por la dirección 127.0.0.1:7000 y el enlace resaltado en la figura 4.5) y de las *cookies* que *sqlmap* necesita para poder realizar el ataque. En la figura 4.6 se muestra su ejecución.


*Figura 4.6. Ejecución de sqlmap.*

Al terminar la auditoría *sqlmap* consigue explotar la vulnerabilidad *SQLi* y ya se puede tomar el control de la base de datos. En la figura 4.7, se puede comprobar el resultado de esta auditoría.

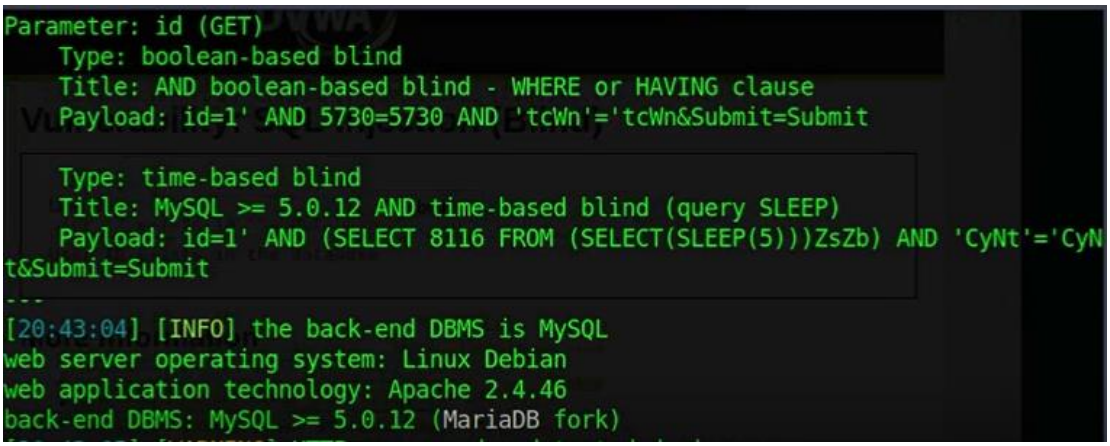


*Figura 4.7. Resultado de la ejecución de sqlmap.*

## Segunda técnica

A continuación, se desarrollará una prueba de concepto donde se clonará un sitio web de un servicio oculto y se inyectará un código malicioso JS en el sitio clonado para poder tomar el control de los usuarios visitantes habituales de dicho tipo de sitio web. Esta técnica está enfocada principalmente a la aplicación sobre sitios frecuentados por ciberdelincuentes, los cuales son los usuarios objetivos a los que se desea comprometer e identificar. No obstante, como se ha mencionado anteriormente, estas prácticas pueden vulnerar el Código Penal, por lo que en esta prueba de concepto se duplicará un sitio web legítimo. Este sitio web se puede apreciar en la figura 4.8. Además, para que esta técnica funcione es un requisito imprescindible que el nivel de seguridad de TOR *Browser* del cliente objetivo este en *Standard,* el cual es el que viene por defecto seleccionado, puesto que el resto de los niveles desactivan JS por lo que el código malicioso no podría ejecutarse.

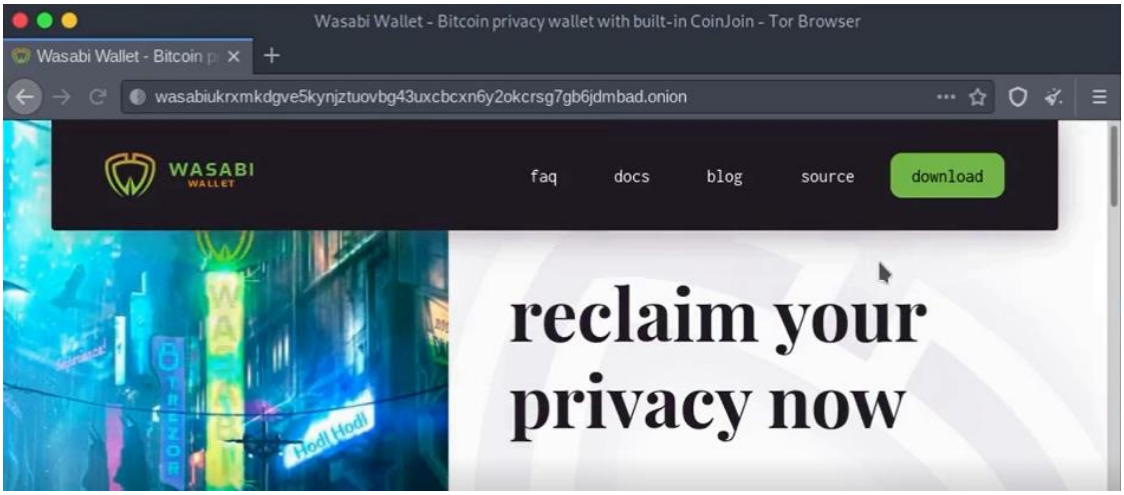


*Figura 4.8. Sitio web real que se duplicará.*

Para poder duplicar este sitio web, es necesario descargarse todos sus contenidos. Para poder realizar este paso, se debe enrutar el tráfico de la consola de comandos a través de TOR. Para ello, se procede a iniciar *anonsurf* y *torsocks.*

Una vez con dichas herramientas activadas, se puede descargar el contenido del sitio web anterior usando *wget* de la forma especificada en la figura 4.9 para obtener exactamente todos los archivos del sitio web y conseguir una copia idéntica.

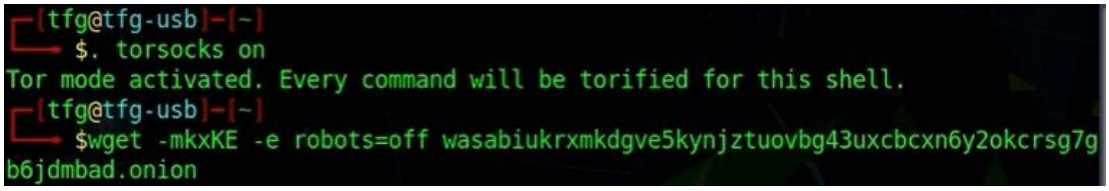


*Figura 4.9. Descarga de los contenidos del sitio web.*

De esta manera, los contenidos del sitio web se guardan en un directorio que tiene como nombre la dirección *onion* del sitio web. Con el sitio web ya descargado, se pueden desactivar las herramientas *anonsurf* y *torsocks.*

Por otra parte, se levantará un servidor web con *nginx* (Gefroh, 2017) (Create Tor Hidden Onion Service Using Nginx, 2019) para alojar el sitio web duplicado. Se debe tener en cuenta que este servidor estará expuesto en TOR donde, al igual que el resto de los servicios, puede ser atacado y vulnerado. Para evitar que convertir este ataque en una vulnerabilidad, se levantarán un servidor web y un servicio oculto siguiendo una serie de prácticas de *hardening* (Best Practices for Hosting Onion Services, n.d.) para que sean lo más seguros posibles.

En primer lugar, se deben copiar el directorio, y los contenidos de este de forma recursiva, del sitio web clonado al directorio */var/www/.* En la figura 4.10 se muestra el comando utilizado para conseguirlo.


*Figura 4.10. Copia de los contenidos del sitio web clonado al directorio /var/www/.*

A continuación, se debe editar el archivo de configuración de *nginx*, llamado */etc/nginx/nginx.conf* e introducir en el bloque *http* las líneas de configuración básicas mostradas en la figura 4.11.

Posteriormente, para alojar el sitio web duplicado en el servidor *nginx* se debe crear un archivo de configuración en el directorio */etc/nginx/sites-available/.* En este caso el archivo tiene el mismo nombre que la dirección *onion* del sitio web real. En este archivo de configuración, aparte de realizar las configuraciones básicas para manejar las peticiones al servidor, se especificará que el sitio web esté escuchando por un *unix socket* en vez de por un puerto de la máquina local lo cual proporciona cierto grado de aislamiento.

Asimismo, también se debe determinar el directorio con los contenidos del sitio web que, como se ha visto anteriormente, se localizan en el directorio copiado a */var/www/.* Igualmente, se establece también el nombre del servidor y el archivo *index.html.* En la figura 4.12 se puede observar el resultado de todas estas configuraciones.

```
http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    types_hash_max_size 2048;
    server_tokens off;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";

    client_body_buffer_size 1k;
    client_header_buffer_size 1k;
    client_max_body_size 1k;
    large_client_header_buffers 2 1k;
```

*Figura 4.11. Configuración del servidor nginx.*

wasabiukrxmkdgve5ky...crsg7gb6jdmbad.onion

```
server {
    listen unix:/home/tfg/unix.sock;
    root /var/www/wasabiukrxmkdgve5kynjztuovbg43uxcbcxn6y2okcrsg7gb6jdmbad.onion;
    index index.html;
    server_name wasabiukrxmkdgve5kynjztuovbg43uxcbcxn6y2okcrsg7gb6jdmbad.onion;
    location / {
        try_files $uri $uri/ = 404;
    }
    if ($request_method !~ ^(GET|HEAD|POST)$ ) {
        return 405;
    }
}
```

*Figura 4.12. Archivo de configuración del sitio web.*

A continuación, para que el sitio web esté disponible, se debe crear un puntero desde el archivo de configuración anterior al directorio */etc/nginx/sites-enabled/.* En la figura 4.13 se puede comprobar el comando para crear dicho puntero.

```
┌[tfg@tfg-usb]-[~]
└─$sudo ln -s /etc/nginx/sites-available/wasabiukrxmkdgve5kynjztuovbg43uxcbcx
n6y2okcrsg7gb6jdmbad.onion  /etc/nginx/sites-enabled/
```

*Figura 4.13. Creación del puntero en /etc/nginx/sites-enabled/.*

Para concluir con las configuraciones del servidor se accede al archivo */lib/systemd/system/nginx.service* y se introduce la sentencia *PrivateNetwork=yes* en el bloque *[Service].* Para hacer efectiva estas configuraciones se puede ejecutar el siguiente comando:

- *sudo systemctl daemon-reload*

Por otro lado, se procederá a configurar BEEF (Walker, 2020). Para que esta herramienta sea más efectiva una vez se haya producido la explotación, se integrará con *metasploit* haciendo uso del módulo *MSGRPC.* En primer lugar, se debe editar el archivo de configuración de BEEF (*/usr/share/beef-xss/config.yaml*) para cambiar la contraseña por defecto del servidor en el bloque *beef: credentials: passwd*, cambiar el puerto 3000 por el puerto 80 en el bloque *beef: http: port* y habilitar el módulo de *metasploit* en el bloque *beef: extension: metasploit.* Igualmente, también se debe habilitar la extensión de metasploit en su archivo de configuración propio (*/usr/share/beef-xss/extensions/metasploit/config.yaml*) y cambiar el bloque *beef: extension: metasploit: enable* a true y cambiar la contraseña por defecto en el bloque *beef: extension: metasploit: pass.*

Una vez con el sitio clonado y BEEF configurados, se procederá a crear los servicios ocultos. Estos pasos se realizan de manera similar a los explicados en la técnica anterior. En primer lugar, se crean los directorios donde se almacenarán los pares de claves junto con las direcciones *onion* de cada servicio oculto. Después, se le cambian los permisos a dichos directorios para que puedan ser accedidos por TOR. Los comandos para realizar estas configuraciones se pueden apreciar en la figura 4.14.


*Figura 4.14. Creación de los directorios de los servicios ocultos.*

Por último, se deben configurar las respectivas líneas *HiddenServiceDir* y *HiddenServicePort* en el fichero de configuración de TOR en */etc/tor/torrc.* Como el servidor de sitio clonado está funcionando con un *unix socket*, este se debe especificar en la línea *HiddenServicePort* tal y como se muestra en la figura 4.15.

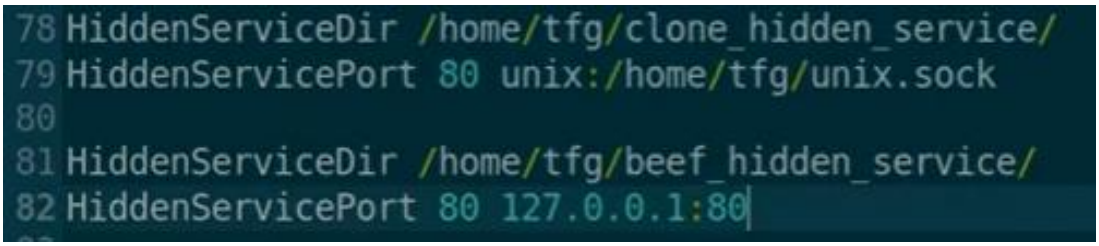


*Figura 4.15. Configurando el fichero de configuración de TOR para levantar los servicios ocultos con BEEF y el sitio clonado.*

Antes de iniciar TOR, se utilizará la herramienta *mkp224o* para personalizar una dirección *onion* para el sitio duplicado que se parezca al sitio web original. En esta prueba de concepto la dirección *onion* del sitio original es:

- http://wasabiukrxmkdgve5kynjztuovbg43uxcbcxn6y2okcrsg7gb6jdmbad.onion

Por tanto, se creará una dirección *onion* cuyas primeras sean *wasabi.* De esta forma es muy poco probable que un cliente se dé cuenta de la diferencia entre ambas direcciones ya que lo más seguro es que solo recuerde los primeros caracteres. Para ello, primero se debe crear un directorio donde generar los pares de claves que se vayan generando (aunque con uno solo de ellos ya es suficiente). Después se inicia *mkp224o* especificando el directorio anteriormente creado y el patrón a utilizar (*wasabi* para esta prueba de concepto). En la figura 4.16, se pueden apreciar las direcciones creadas usando fuerza bruta.

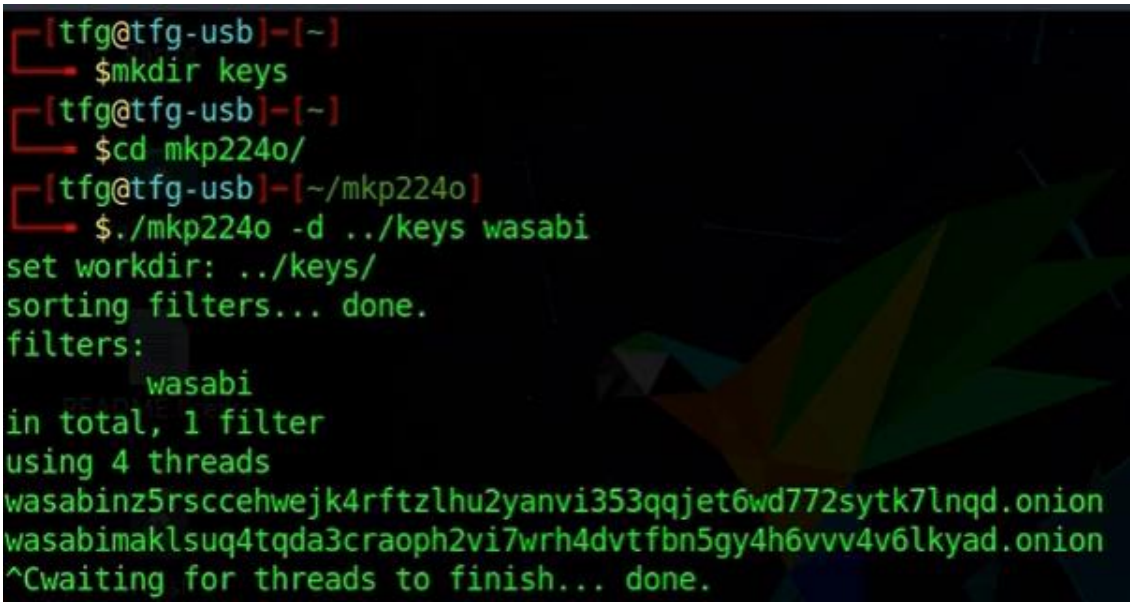


*Figura 4.16. Creación de direcciones onion personalizadas.*

Para poder usar una de las direcciones creadas, se debe navegar a su directorio en el directorio *keys* y copiar el archivo *hostname* y el par de claves, es decir, todo el contenido de ese directorio en el directorio del servicio oculto del sitio web clonado previamente especificado en */etc/tor/torrc,* en este caso dicho directorio sería */home/tfg/clone_hidden_service.* Así, cuando se inicie, TOR utilizará ese par de claves para levantar el servicio oculto en vez de crear un desde 0.

A continuación, antes de iniciar BEEF, se debe cargar el módulo MSGRPC en *metasploit* con el usuario y la contraseña especificada en el archivo */usr/share/beef-xss/extensions/metasploit/config.yaml* para integrar este posteriormente con BEEF (Mike, 2018). *Metasploit* se puede iniciar con el siguiente comando:

- *sudo msfconsole.*

En la figura 4.17 se puede apreciar la forma de cargar el módulo MSGRPC en *metasploit.*

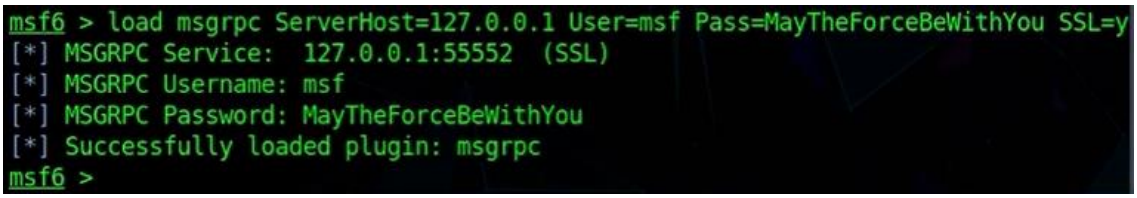


*Figura 4.17. Carga del módulo MSGRPC en metasploit.*

Una vez cargado dicho módulo, ya se puede iniciar BEEF. En la figura 4.18, se puede verificar que los módulos provenientes de *metasploit* se han cargado de forma correcta.

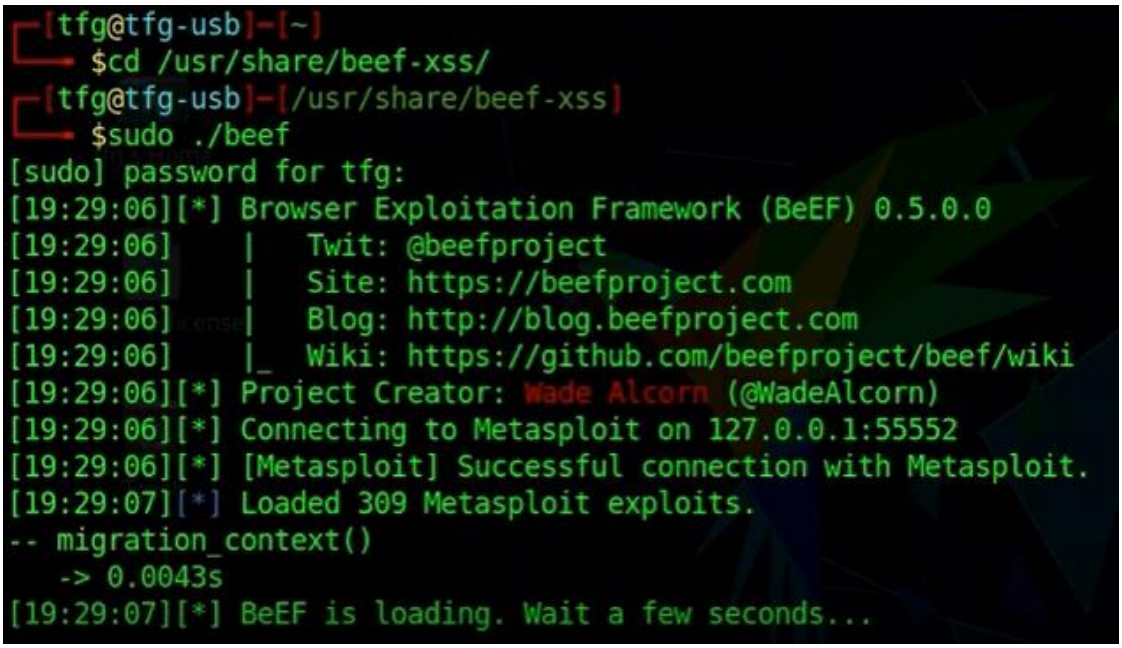


*Figura 4.18. Carga de BEEF.*

Además, se inyectará código malicioso JS en el sitio clonado para tomar el control de los clientes que accedan a dicho servicio oculto usando BEEF. No obstante, para ello, es necesario que el servicio oculto de BEEF tenga una dirección *onion* donde se conecten los clientes vulnerados, por lo que es necesario iniciar TOR durante unos momentos para genere automáticamente dicha dirección.

Obtenida la dirección *onion*, en el archivo *index.html* del sitio clonado situado en el directorio */var/www/* se insertará una etiqueta <*script*> para que los clientes se descarguen el código del servicio oculto de BEEF que permitirá que este tome el control de aquellos. Dicha etiqueta se muestra en la figura 4.19. La URL que se puede apreciar en dicha figura es la del servicio oculto de *BEEF.*

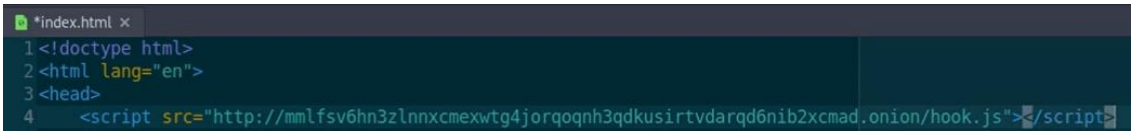


*Figura 4.19. Inyección de código malicioso JS en el sitio clonado.*

Finalmente, se inicia *nginx* previa recarga de su configuración para levantar el servidor web y se inicia TOR para levantar los servicios ocultos de dicho servidor y de BEEF usando los comandos siguientes:

- *sudo systemctl daemon-reload*
- *sudo service nginx start*
- *tor*

Después de realizar todas estas configuraciones, ya se tiene preparado el laboratorio para realizar el ataque. No obstante, antes de realizar la explotación, se realizará una auditoría rápida con la herramienta *onionscan* del servicio oculto del sitio clonado para comprobar su seguridad.

Para realizar esta auditoría se ejecuta el comando mostrado en la figura 4.20. La dirección *onion* de la figura es la dirección personalizada del servicio clonado.

```
┌─[tfg@tfg-usb]─[~]
└──╼ $sudo ./go/bin/onionscan --verbose http://wasabimaklsuq4tqda3craoph2vi7wrh4
dvtfbn5gy4h6vvv4v6lkyad.onion/
```

*Figura 4.20. Auditoría sobre el servicio oculto del sitio web clonado.*

Tras unos segundos, *onionscan* termina todas las comprobaciones y muestra el resultado final. Como se puede verificar en la figura 4.21 no se ha detectado ninguna vulnerabilidad.

```
--------------- OnionScan Report ---------------
Generating Report for: wasabimaklsuq4tqda3craoph2vi7wrh4dvtfbn5gy4h6vvv4v6lkyad.
onion

No risks were found.
```

*Figura 4.21. Resultado de la auditoría de onionscan.*

Una vez comprobada la seguridad del servicio oculto, se procede a realizar el ataque para tomar el control de un cliente que visite el sitio clonado. En primer lugar, con TOR *Browser,* se navega a la dirección *onion* del servicio oculto de BEEF (se recuerda que esta dirección se encuentra en el archivo *hostname* del directorio del servicio oculto) para acceder a su interfaz de usuario ubicada en el subdirectorio */ui/panel*. Esta interfaz se puede observar en la figura 4.22. Para acceder al panel de control, se debe introducir el usuario y la contraseña especificados en el archivo */usr/share/beef-xss/config.yaml.*

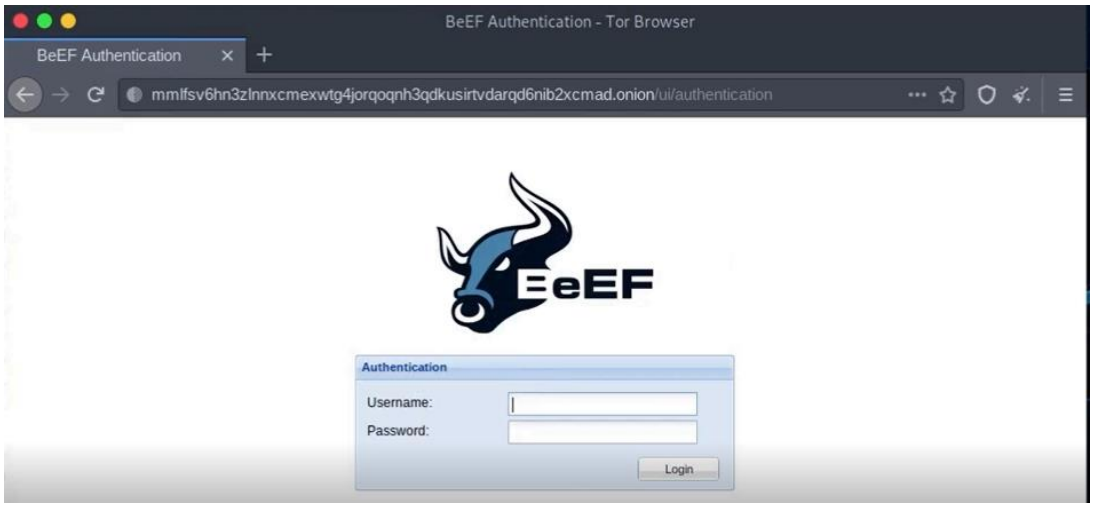


*Figura 4.22. Interfaz web de BEEF.*

Posteriormente, se accede al servicio oculto clonado con la dirección *onion* personalizada creada anteriormente. Este servicio oculto se muestra en la figura 4.23. Si se comparan la figura anterior con la figura 4.8, se puede comprobar claramente como la primera es una copia con los mismos contenidos que la segunda.

Desde que se carga este sitio web, ya se está ejecutando la rutina JS para conectar el cliente con el servicio oculto de BEEF. Esto se puede comprobar en la pestaña *Network* del inspector de TOR mostrado en la figura 4.24.

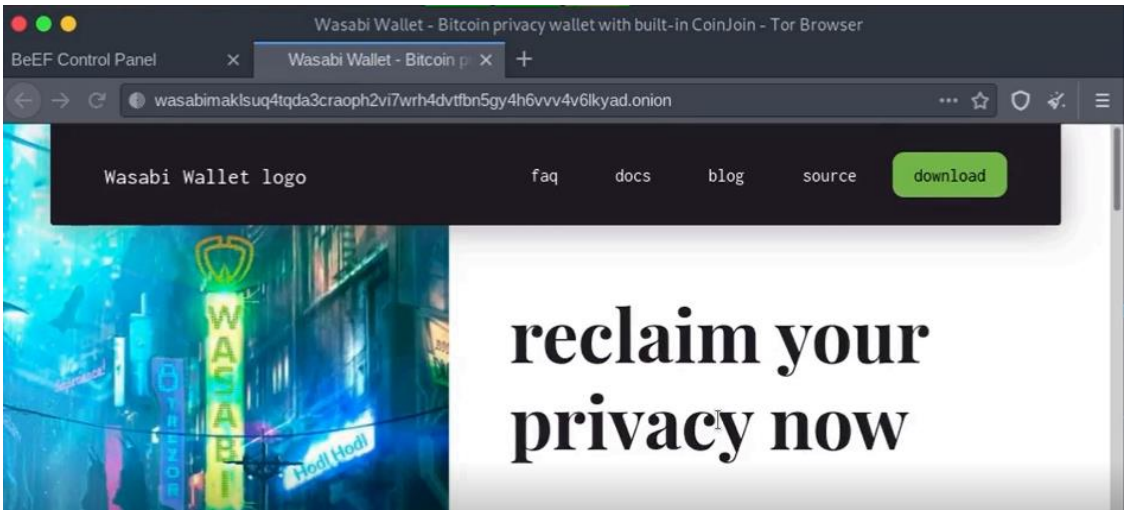


*Figura 1.23. Sitio web clonado.*

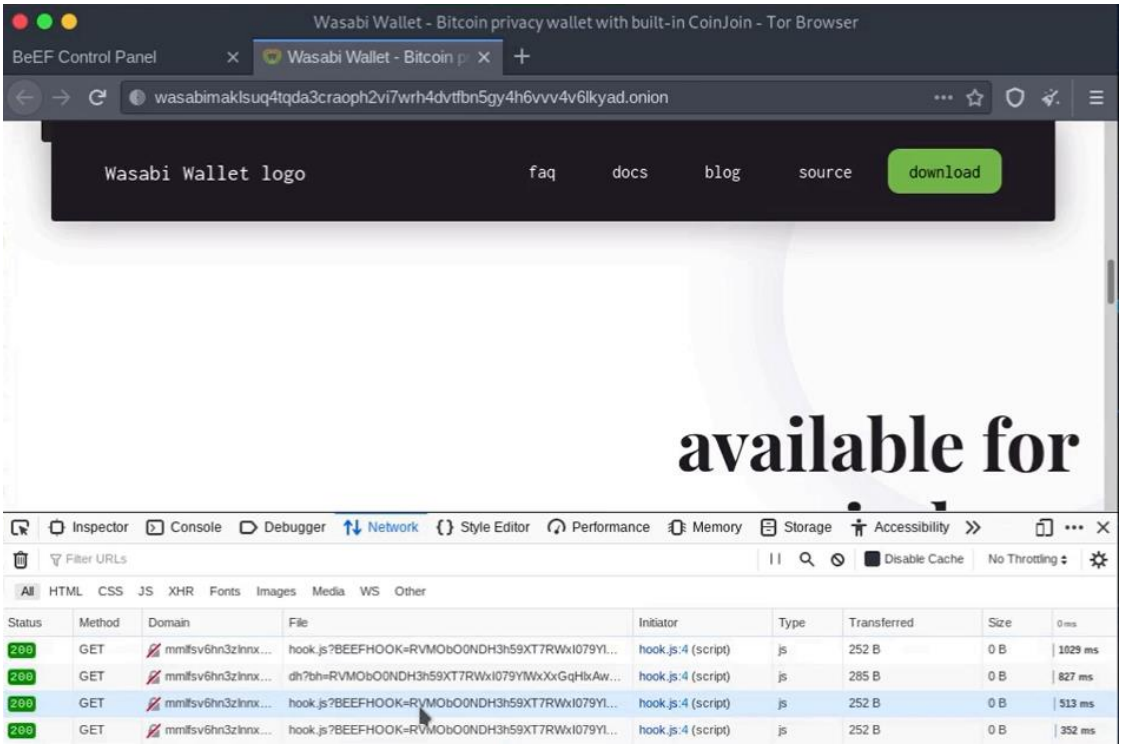


*Figura 4.24. Ejecución del código malicioso.*

Gracias a este código, BEEF ha conseguido capturar el navegador del cliente y ya tiene acceso a gran cantidad de información de ese navegador como los detalles más importantes sobre este o los *logs.* Sin duda, la funcionalidad más interesante se encuentra en la pestaña *Commands* donde se tienen disponibles gran cantidad de módulos, incluidos aquellos integrados con *metasploit* para poder realizar numerosos ataques sobre ese cliente. Esto se puede observar en la figura 4.25.

Para demostrar la eficacia de estos módulos se ejecutará uno de ellos para mostrar al cliente una falsa barra de notificación. Para ello, en la carpeta *Social Engineering* se selecciona la opción *Fake Notification Bar.* Después en el panel *Fake Notification Bar,* se introduce el texto que se desee mostrar. Finalmente, para ejecutar el módulo, se selecciona el botón *Execute.* En la figura 4.26 se muestran los pasos descritos.

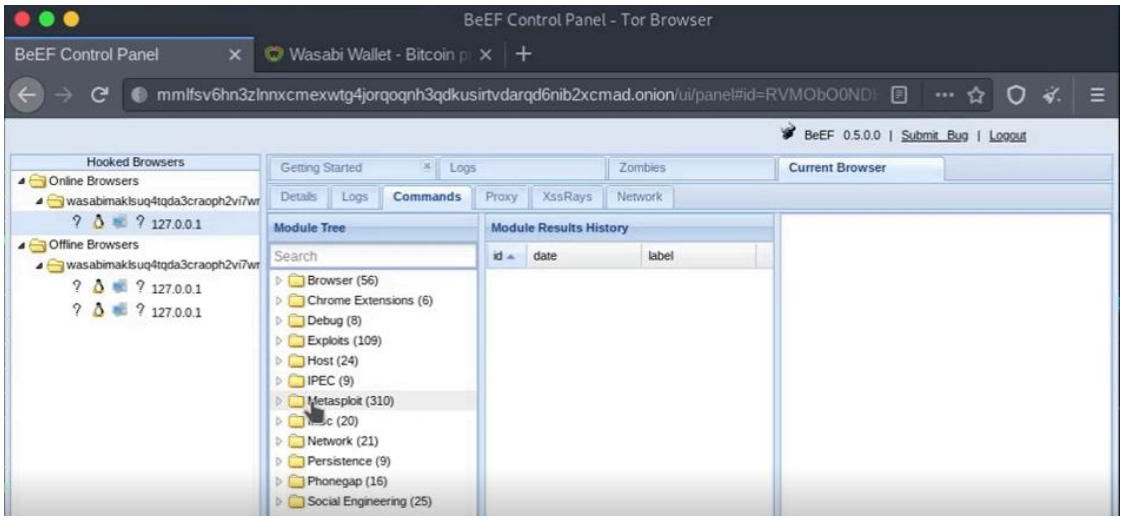


*Figura 4.25. Módulos de ataque en BEEF.*

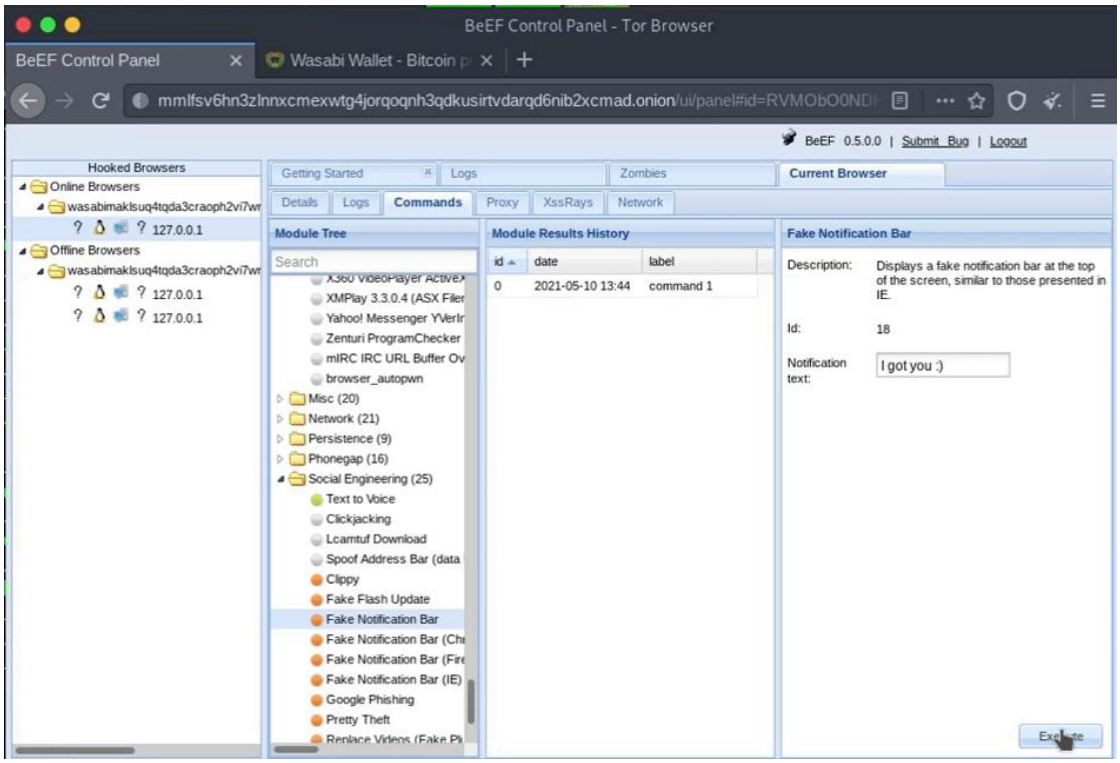


*Figura 4.26. Ejecución del módulo de BEEF Fake Notification Bar.*

Para concluir, se muestra el resultado de la ejecución de este módulo en el sitio clonado en la figura 4.27.

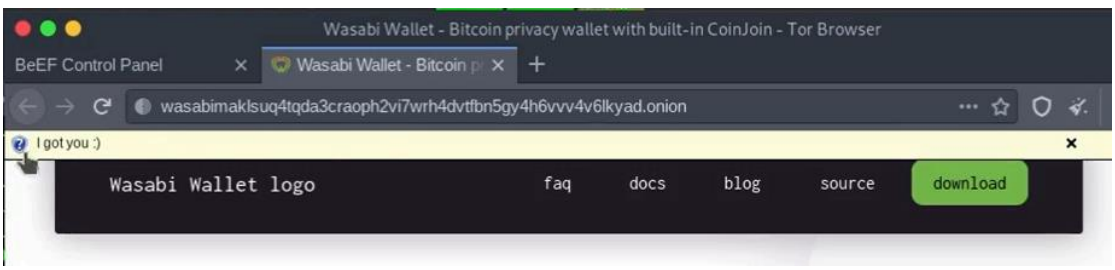


*Figura 4.27. Módulo ejecutado.*

# Capítulo 5 - Conclusiones y propuestas

En lo referente a la primera técnica, aunque la prueba de concepto se ha desarrollado con *sqlmap* el procedimiento se puede aplicar para cualquier herramienta de *pentesting* que se utilizase si el objetivo del ataque estuviera localizado en la *surface web*. Igualmente, aunque se ha levantado un servidor propio con vulnerabilidades, está técnica se puede utilizar contra cualquier servicio oculto disponible en TOR, los cuales, pueden presentar vulnerabilidades al igual que resto de servicios en el *surface web.* Así, se resalta el hecho de que, aunque TOR proporcione privacidad y anonimato no quiere decir que ofrezca seguridad, por lo que una vez vulnerada esta, las dos anteriores no quedan aseguradas.

Por otra parte, en la segunda técnica solo se ha desarrollado la parte de explotación, es decir, la etapa donde se consigue llegar a monitorizar al usuario objetivo mientras desarrolla actividades sospechosas no siendo él consciente de estar siendo atacado. Sin embargo, para que esta técnica sea efectiva, el cliente objetivo debe obtener y navegar primero a la dirección *onion* del sitio web duplicado. Esto requiere de una fase de pre-explotación, la cual se podría realizar mediante técnicas ingeniería social o, dicho de otro modo, engañar o convencer al usuario de que acceda a ese servicio oculto. No obstante, un buen método para conseguir que el cliente objetivo entre en el servicio oculto clonado podría consistir en publicar la dirección en buscadores habituales de TOR como *NotEvil* ya que estos buscadores no son como los convencionales donde los enlaces se pueden ver afectados por actividades como el número de búsquedas.

Con todo esto, se ha pretendido demostrar que las estrategias de análisis, monitorización y demostración de ciberdelitos es posible en redes anónimas como TOR, aunque esta esté especializada en el anonimato de sus usuarios.

# Bibliografía

Al-Nabki, W., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019). *ToRank: Identifying the most influential suspicious domains in the Tor network.* León: Elsevier. doi:10.1016/j.eswa.2019.01.029

Alsabah, M., & Goldberg, I. (2016). *Performance and Security Improvements for Tor: A Survey.* Qatar. doi:10.1145/2946802

Amoany, E. (25 de Junio de 2020). *Getting started with socat, a multipurpose relay tool for Linux*. Recuperado el 12 de Mayo de 2021, de https://www.redhat.com/sysadmin/getting-started-socat

Bernaschi, M., Celestini, A., Guarino, S., & Lombardi, F. (2017). *Exploring and Analyzing the Tor Hidden Services Graph.* IANCIS, Roma. doi:10.1145/3008662

*Best Practices for Hosting Onion Services*. (s.f.). Recuperado el 21 de Febrero de 2021, de https://riseup.net/en/security/network-security/tor/onionservices-best-practices

*cathugger/mkp224o*. (23 de Marzo de 2021). Recuperado el 25 de Marzo de 2021, de https://github.com/cathugger/mkp224o

*Configuring Onion Services for Tor*. (2019). Recuperado el 17 de Febrero de 2021, de https://2019.www.torproject.org/docs/tor-onion-service.html.en

*Create Tor Hidden Onion Service Using Nginx*. (9 de Septiembre de 2019). Recuperado el 3 de Marzo de 2021, de https://arcdetri.github.io/tor-hidden-onion-nginx.html

*digininja/DVWA*. (3 de Marzo de 2021). Recuperado el 13 de Marzo de 2021, de https://github.com/digininja/DVWA

*Discovering the Dark Web*. (s.f.). Recuperado el 6 de Marzo de 2021, de https://onionscan.org/

Echeverri, D. (12 de Marzo de 2018). Hackeando TOR y Freenet por diversión, lucro y detener a los malos. Valencia, Valencia, España. Recuperado el 10 de Diciembre de 2020, de https://www.youtube.com/watch?v=V1msORieS4I

Erdin, E., Zachor, C., & Hunes, M. H. (2015). *How to Find Hidden Users: A Survey of Attacks on Anonymity Networks.* Universidad de Nevada, Informática e Ingeniería. Reno: IEEE. doi:10.1109/COMST.2015.2453434

Filiol, E., Delong, M., & Nicolas, J. (2019). *Statistical and combinatorial analysis of the TOR routing protocol:.* Departamento de Defensa de Francia. París: Springer. doi:10.1007/s11416-019-00334-x

Gefroh, J. (5 de Septiembre de 2017). *A guide to hosting static websites using NGINX*. Recuperado el 27 de Febrero de 2021, de https://jgefroh.medium.com/a-guide-to-using-nginx-for-static-websites-d96a9d034940

*GNU Wget*. (4 de Agosto de 2020). Recuperado el 12 de Mayo de 2021, de https://www.gnu.org/software/wget/

*How Does Tor Really Work? The Definitive Visual Guide (2020)*. (2019 de Septiembre de 2020). Recuperado el 13 de Mayo de 2021, de https://skerritt.blog/how-does-tor-really-work/

*Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal.* . (23 de Noviembre de 1995). Recuperado el 26 de Enero de 2021, de Agencia Estatal Boletín Oficial del Estado: https://www.boe.es/buscar/act.php?id=BOE-A-1995-25444&p=20201217&tn=1

Mike. (29 de Julio de 2018). *Integrating BeEF and Metasploit*. Recuperado el 6 de Mayo de 2021, de https://nullsec.us/integrating-beef-and-metasploit/

Owenson, G., Cortes, S., & Lewman, A. (2018). *The darknet's smaller than we thought: The life cycle of Tor Hidden.* Portsmounth: Elsevier. doi:10.1016/j.diin.2018.09.005

*Parrot OS*. (2020). Recuperado el 25 de Enero de 2021, de https://parrotsec.org/

Saleh, S., Qadir, J., & Ilyas, M. U. (2018). *Shedding Light on the Dark Corners of the Internet: A Survey of Tor.* Elvesier. doi:10.1016/j.jnca.2018.04.002

*Set up Your Onion Service*. (s.f.). Recuperado el 17 de Febrero de 2021, de https://community.torproject.org/onion-services/setup/

*s-rah/onionscan*. (25 de Febrero de 2017). Recuperado el 6 de Marzo de 2021, de https://github.com/s-rah/onionscan

Steinebach, M., Zenglein, S., & Brandl, K. (2021). Phising detection on tor hidden services, Forensic Science International: Digital Investigation. pág. 3. doi:10.1016/j.fsidi.2021.301117

Tor Hidden Services. (s.f.). Recuperado el 13 de Mayo de 2021

Walker, J. (21 de Julio de 2020). *Configuration*. Recuperado el 6 de Mayo de 2021, de https://github.com/beefproject/beef/wiki/Configuration

*Welcome to NGINX Wiki!* (s.f.). Recuperado el 12 de Mayo de 2021, de https://www.nginx.com/resources/wiki/

# Anexos

## Anexo I. Instalación de *onionscan.*

A continuación, se explicarán los pasos necesarios para instalar y utilizar la herramienta *onionscan* (s-rah/onionscan, 2017). En primer lugar, se deben instalar las dependencias necesarias con los siguientes comandos:

- *go get github.com/HouzuoGuo/tiedot*
- *go get golang.org/x/crypto/openpgp*
- *go get golang.org/x/net/proxy*
- *go get golang.org/x/net/html*
- *go get github.com/rwcarlsen/goexif/exif*
- *go get github.com/rwcarlsen/goexif/tiff*

Posteriormente, se puede instalar la herramienta con los siguientes comandos:

- *go get github.com/s-rah/onionscan*
- *go install github.com/s-rah/onionscan*

No obstante, *onionscan* está configurado por defecto para auditar con dominios *onion* v2 (direcciones de 16 caracteres). Para configurar esta herramienta para que funcione también con dominios *onion* v3 (direcciones de 57 caracteres) se debe editar el archivo *go/src/github.com/s-rah/onionscan/utils/validation.go* y cambiar la expresión "{16}" por "{16,57}" como se muestra en la figura Anexo I.

validation.go ×

```
1  package utils
2
3  import (
4      "regexp"
5      "strings"
6  )
7
8  func IsOnion(identifier string) bool {
9      // TODO: At some point we will want to support i2p
10     if len(identifier) >= 22 && strings.HasSuffix(identifier, ".onion") {
11         matched, _ := regexp.MatchString(`(^|\.)[a-z2-7]{16,57}\.onion$`, identifier)
12         return matched
13     }
14     return false
15 }
```

*Figura Anexo I. Configuración para que onionscan funciones con dominios onion v3.*

Al realizar esta configuración, se debe reinstalar la herramienta con el siguiente comando:

- *go install github.com/s-rah/onionscan*

Finalmente, para ejecutar la herramienta:

- *./go/bin/onionscan –verbose (dirección onion que se desee auditar).*

## Anexo II. Instalación de *mkp224o.*

Para utilizar esta herramienta (cathugger/mkp224o, 2021), en primer lugar, se deben instalar los paquetes necesarios con el comando:

- *sudo apt install gcc libsodium-dev make autoconf*

Posteriormente, se clona el repositorio:

- *git clone https://github.com/cathugger/mkp224o.git*

Finalmente, se navega al directorio *mkp224o* y se instala la herramienta usando los siguientes comandos:

- *./autogen.sh*
- *./configure*
- *make*

Una vez realizadas las configuraciones previas, las direcciones *onion* v3 se pueden crear fácilmente:

- *./mkp224o -d (directorio donde almacenar las claves) (dirección personalizada)*

## Anexo III. Instalación de DVWA.

Para poder instalar DVWA (digininja/DVWA, 2021), se debe instalar previamente los siguientes paquetes:

- *sudo apt install apache2 mariadb-server php php-mysqli php-gd libapache2-mod-php*

Posteriormente, se clona el repositorio en el directorio */var/www/html/*:

- *git clone https://github.com/digininja/DVWA.git*

Una vez con el repositorio clonado, se procede a configurar el servidor. Una manera sencilla de realizar esto es copiando el archivo de configuración por defecto:

- *sudo cp DVWA/config/config.inc.php.dist DVWA/config/config.inc.php.*

Este servidor web viene acompañado de un servicio *mysql,* el cual también debe configurarse para garantizar el buen funcionamiento de la aplicación. Para ello, se ejecutan los siguientes comandos:

- *sudo mysql*
- *create database dvwa;*
- *create user dvwa@localhost identified by 'p@ssw0rd';*
- *grant all on dvwa.* to dvwa@localhost;*
- *flush privileges;*

Igualmente, para el correcto funcionamiento de la aplicación también se deben cambiar los permisos de las siguientes carpetas:

- *sudo chmod 777 /var/www/html/DVWA/hackable/uploads/*
- *sudo chmod 777 /var/www/html/DVWA/config/*
- *sudo chmod 777 /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt*

Finalmente, se debe editar el archivo */etc/php/7.4/apache2/php.ini* estableciendo los campos *allow_url_include* y *allow_url_fopen* a *On* y los campos *safe_mode, magic_quotes_gpc* y *display_errors* a *Off*.

Para iniciar esta aplicación, se ejecutan los siguientes comandos:

- *sudo service apache2 start*
- *sudo service mysql start*

UNIVERSIDAD DE ALMERÍA


## Resumen/Abstract

Es una verdad ampliamente conocida que *dark nets* como TOR ofrecen a los ciberdelincuentes una gran oportunidad para poseer privacidad y anonimato a la hora de realizar actividades ilegales. La dificultad de demostrar la realización de este tipo de delitos, los cuales han ido incrementando a lo largo de estos últimos años, e identificar a los culpables, sumado a la falta de herramientas para operar en este tipo de redes anónimas, se ha convertido en una amenaza para las fuerzas y cuerpos de seguridad encargados de combatirlos. Por lo tanto, en este proyecto se desarrollarán pruebas de concepto donde se aplicarán técnicas que se pueden utilizar en estos entornos y que permiten adaptar procedimientos y estrategias utilizadas con frecuencia en la *surface web*.

Palabras clave: *The Onion Routing* (TOR), servicios ocultos (*hidden services*), clonación.

*It is a truth universally acknowledged that dark nets such as TOR offer a great opportunity to obtain privacy and anonymity for cybercriminals while committing illegal activities. The difficulties presented at proving these types of crimes, which have been increasing over the past years, and identifying the culprits, all added to the lack of specialized tools for these types of anonymous networks, have become a threat for law enforcements agents in charge of fighting them. Therefore, in this project some proofs of concept will be developed where different techniques could be used in these environments which allow adapting procedures and strategies frequently used in the surface web.*

*Key words: The Onion Router (TOR), hidden services, cloning.*