UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Expectativas y riesgos en el uso de la red Tor: Un análisis enfocado en la seguridad de la información

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Virgilio Castro Rendón

DIRECTOR DE TESIS

M.C. Paulo Santiago de Jesús Contreras Flores

Ciudad Universitaria, Cd. Mx., 2019

Resumen

Tor es una red superpuesta a Internet que funciona gracias a aportaciones económicas y no económicas de miles de personas y organizaciones de todo el mundo y cuyo objetivo es brindar anonimato a quienes la utilizan. Esta red basa su funcionamiento en un conjunto de nodos que crean circuitos entre sí y que son usados para ocultar el origen de las comunicaciones, dificultando el rastreo directo de las mismas. De igual forma, la red implementa medidas para mantener íntegros y confidenciales los datos de sus usuarios, así como disponibles sus conexiones. Estas características de la red Tor pueden crear altas expectativas entre quienes la usan y, de igual manera, le han creado una mala reputación, pues muchos creen que la red, dada sus características, es aprovechada principalmente por delincuentes cibernéticos.

Para probar hasta qué punto estas ideas se cumplen, en este trabajo se presenta la configuración de un nodo productivo perteneciente a la red Tor y la aplicación de diferentes técnicas de análisis para revisar detenidamente la actividad de sus usuarios y determinar en qué medida esta es potencial o totalmente maliciosa. Adicionalmente, mediante pruebas de concepto se analizan los posibles riesgos asociados al uso inadecuado de esta red y la relativa facilidad con que administradores maliciosos podrían vulnerar la seguridad de los datos que viajan a través de ella.

Índice General

Resumen iii
Índice General v
Índice de Imágenes vii
Índice de tablas ix
Introducción 1
Objetivo 1
Estructura de la tesis 1
Capítulo 1 1
Capítulo 2 2
Capítulo 3 2
Capítulo 4 2
Capítulo 5 2
Capítulo 6 3
1. Capítulo 1 Marco teórico 5
1.1 Servicios de la seguridad de la información 5
1.2 Modelo TCP/IP 6
1.3 Criptografía y sus aplicaciones 11
1.4 Amenazas y vulnerabilidades de la seguridad de la información 14
1.5 Análisis estructurado de tráfico de red 18
1.6 Privacidad y anonimato en redes 19
1.7 Planteamiento del problema y solución propuesta 20
2. Capítulo 2 La red Tor 23
2.1 Objetivo 23
2.2 Funcionamiento 23
2.3 Tipos de nodos 26
2.4 Servicios ocultos 27
2.5 Anonimato en redes 28
3. Capítulo 3 Puesta en marcha de un nodo de la red Tor 31
3.1 Búsqueda y renta de un servidor público 31
3.2 Instalación de un nodo Tor 34
3.3 Configuraciones para tener un nodo de salida 42
3.4 Dificultades encontradas al poner en marcha la infraestructura 44
4. Capítulo 4 Análisis de la actividad en la red Tor 47

4.1 Visualización geográfica basada en las conexiones de un nodo de entrada 47
4.2 Captura de tráfico de red 49
4.3 Nombres de dominio consultados 53
4.4 Servicios de la capa de aplicación 60
4.5 Dispositivos usados para acceder a la red 66
4.6 Búsqueda de actividad maliciosa 69
4.7 Automatización del análisis 74
5. Capítulo 5 Seguridad en la red Tor 81
5.1 Análisis de la confidencialidad 81
5.2 Análisis de la integridad 86
5.3 Análisis de la disponibilidad 93
6. Capítulo 6 Conclusiones 101
6.1 Instalación y configuración de un nodo en la red Tor 101
6.2 Actividad de los usuarios de la red Tor en la red abierta 101
6.3 Seguridad de la información en la red Tor 102
6.4 Trabajo Futuro 103
Bibliografía 105
Anexo A. Glosario 109
Anexo B. Programas 113
Creación de nuevas capturas de tráfico sin HTTPS 113
Extracción de agentes de usuario 113
Extracción de peticiones DNS 114
Creación de tabla SQLite 114
Visualización de datos (TLDs y SLDs) 115
HTTP scrapping 119
Procesamiento de PCAP y visualización de flujos 122
Visualización de conexiones hacia la red (World map) 126
Listado de nodos de entrada usados en cliente 128
Interacción con VirusTotal para dominios maliciosos 129
Análisis de origen de dominio (TLD y Whois) 133

Índice de Imágenes

Imagen 2.1. Diagrama de conexiones en la red Tor. 24
Imagen 2.2. Capas de cifrado removidas por cada nodo en el circuito. 25
Imagen 3.1. Espacio disponible para almacenamiento en el servidor. 33
Imagen 3.2. Versión del sistema operativo instalado en el servidor. 34
Imagen 3.3. Instalación de tor mediante paquete. 36
Imagen 3.4. Archivo de configuración de paquetes a actualizar automáticamente. 36
Imagen 3.5. Archivo de configuración para habilitar actualizaciones automáticas. 37
Imagen 3.6. Validación de configuración de actualizaciones automáticas. 37
Imagen 3.7. Archivo de configuración de tor. 38
Imagen 3.8. Pruebas automáticas por parte de tor para validar configuración. 39
Imagen 3.9. Nodo mostrado en el sistema de búsquedas de la red Tor. 40
Imagen 3.10. Pantalla de consola de monitoreo del nodo. 41
Imagen 3.11. Probabilidad de ser nodo de entrada, intermedio o de salida. 42
Imagen 3.12. Petición DNS para hacer resolución inversa de la dirección IP del nodo. 43
Imagen 3.13. Página web mostrada al visitar el servicio HTTP del nodo. 43
Imagen 3.14. Características del nodo mostradas por el sistema de búsquedas de la red. . 44
Imagen 4.1. Cantidad de conexiones por país hacia un nodo de entrada. 48
Imagen 4.2. Uso general de Internet en todo el mundo. 49
Imagen 4.3. Ejecución de tcpdump para mostrar su versión. 50
Imagen 4.4. Script usado para capturar el tráfico de red. 50
Imagen 4.5. Regla de crontab para ejecutar cada hora el script de captura de tráfico. 51
Imagen 4.6. Listado de archivos pcap generados. 52
Imagen 4.7. Relación de tráfico del puerto 443 en comparación con el resto de puertos. ... 52
Imagen 4.8. Listado de capturas de tráfico al aplicarles el nuevo filtro. 53
Imagen 4.9. Filtro para obtener los nombres de dominio consultados por el nodo. 54
Imagen 4.10. Conteo de dominios consultados totales y únicos. 55
Imagen 4.11. Top 10 de dominios de nivel superior consultados. 56
Imagen 4.12. Top 15 de dominios de segundo nivel más consultados. 57
Imagen 4.13. Actividad por país basada en los nombres de dominio consultados. 58
Imagen 4.14. Top 10 de dominios mexicanos consultados. 59
Imagen 4.15. Relación 1 de cantidad de tráfico por puerto en un archivo pcap. 60
Imagen 4.16. Relación 2 de cantidad de tráfico por puerto en un archivo pcap. 61
Imagen 4.17. Porcentaje 1 de tráfico en los principales puertos. 62
Imagen 4.18. Porcentaje 2 de tráfico en los principales puertos. 62
Imagen 4.19. Ejecución de herramienta en Python que analiza y grafica los flujos de una captura. 63
Imagen 4.20. Gráfica de flujos por puerto obtenida de un único archivo de captura. 63
Imagen 4.21. Gráfica de flujos por puerto obtenida de todos los archivos de captura. 64
Imagen 4.22. Salida de tcpdstat con análisis estadístico preliminar. 65
Imagen 4.23. Salida de tcpdstat con clasificación de puertos más usados basado en número de paquetes. 66
Imagen 4.24. Filtro a partir de cadenas para obtener los agentes de usuarios de dispositivos móviles. 68
Imagen 4.25. Relación de dispositivos móviles y estáticos. 69
Imagen 4.26. Consola principal de Xplico con el conteo de correos enviados. 70

Imagen 4.27. Correo con archivo malicioso. 71
Imagen 4.28. Correo no deseado suplantando la identidad de un dominio mexicano. 71
Imagen 4.29. Correo electrónico enviado desde un servidor con Open Relay. 72
Imagen 4.30. Resultado 1 del análisis de un archivo en packettotal..... 76
Imagen 4.31. Resultado 2 del análisis de un archivo en packettotal..... 76
Imagen 4.32. Detalles de alertas generadas por el tráfico de Tor..... 77
Imagen 4.33. Alertas generadas por el tráfico de Tor sobre malware y servidores CNC. 78
Imagen 4.34. Detección de una URL maliciosa por varios motores de VirusTotal. 79
Imagen 4.35. Salida del programa donde se muestra el total de dominios relacionados con actividad maliciosa. 80
Imagen 4.36. Total de URLs relacionadas y no relacionadas con actividad maliciosa..... 80
Imagen 5.1. Datos de autenticación falsos en la página Altoro Mutual. 82
Imagen 5.2. Credenciales de acceso de prueba capturadas en nodo de salida. 83
Imagen 5.3 Filtro en tcpdump para obtener las peticiones HTTP POST..... 83
Imagen 5.4. Petición HTTP POST con credenciales para autenticarse en aplicación web. . 85
Imagen 5.5. Conteo de peticiones HTTP POST encontradas en una captura de tráfico..... 85
Imagen 5.6. Número de peticiones POST por cada hora de captura..... 86
Imagen 5.11. Infraestructura de red para acceder a un sitio de forma común. 87
Imagen 5.12. Infraestructura de red modificada por atacante para acceder a recursos en internet. 88
Imagen 5.13. Validación de la inexistencia del dominio de prueba con WHOIS. 89
Imagen 5.14. Validación de la inexistencia del dominio de prueba mediante una resolución DNS. 89
Imagen 5.15. Contenido del archivo hosts para resolver dominio de prueba. 89
Imagen 5.16. Ejecución del programa dnsmasq..... 90
Imagen 5.17. Configuración del servidor DNS en la interfaz local. 91
Imagen 5.18. Configuración del cliente para utilizar preferentemente el nodo de salida..... 91
Imagen 5.19. Extracto de dominios consultados en el servidor DNS..... 91
Imagen 5.20. Ejecución y validación de la prueba de concepto. 92
Imagen 5.21. Registro del dominio de prueba en la bitácora del servicio DNS. 93
Imagen 5.22. Histórico de número de nodos en últimos meses. 94
Imagen 5.23. Histórico de número de nodos en los últimos años. 95
Imagen 5.24. Cantidad de tráfico visto en el nodo reportado por la red. 96
Imagen 5.25. Conexiones por minuto durante un día entero. 97
Imagen 5.26. Circuitos creados en un cliente de la red. 98
Imagen 5.27. Detención de servicio tor y su hora. 98
Imagen 5.28. Creación de nuevos circuitos al fallar los existentes. 99

Índice de tablas

Tabla 3.1. Requisitos técnicos para montar un nodo en la red. 32
Tabla 4.1. Agentes de usuario más comunes 68
Tabla 4.2. Cadenas de Agentes de usuario potencialmente maliciosas. 74
Tabla 5.1. Valores utilizados en la prueba de concepto. 82

Introducción

Objetivo

La red Tor es utilizada por miles de usuarios que buscan garantizar la seguridad de sus datos que viajan por internet. Ellos creen que los datos se encuentran totalmente seguros, es decir, que se mantienen íntegros, confidenciales y disponibles; además buscan el anonimato en sus conexiones de red. Hay usuarios que emplean la red Tor para acceder de manera libre en lugares donde existen bloqueos de acceso a internet. Por otro lado, se ha extendido la creencia de que su uso principal es realizar actividades maliciosas e ilegales.

Con motivo de lo anterior, el objetivo de esta tesis es el analizar de primera mano y con herramientas de terceros y propias, el funcionamiento de la red Tor, mediante un nodo que forme parte de la red para conocer las actividades de los usuarios y determinar si estas son principalmente maliciosas. Asimismo se busca conocer el nivel de seguridad que es capaz de proveer a los datos de quienes la utilizan, considerando la integridad, confidencialidad y disponibilidad como los servicios principales de la seguridad de la información.

La relevancia de esta investigación también radica en que sus resultados permitirán a los especialistas en seguridad, y a prácticamente cualquier persona que se interese en el tema, contar con información veraz y precisa sobre la eficacia de la red Tor para mantener seguras sus comunicaciones y sobre qué prácticas adicionales se deben seguir para garantizar tanto el anonimato como la seguridad de los datos.

Se decidió implementar una infraestructura que forme parte de la red Tor, mediante la renta de un servidor expuesto a internet, para investigar a fondo y de primera mano las características de esta red. Por la naturaleza de la Tor y debido a que la mayor parte de los análisis planteados no son posibles sin datos verídicos, se decidió no montar una infraestructura de prueba sino utilizar los datos directamente de la red, es decir, estos serán reales.

A continuación, de forma sucinta describo cada uno de los capítulos que componen esta tesis.

Estructura de la tesis

Capítulo 1

En el primer capítulo se establecen diversas definiciones y términos necesarios para comprender el contenido de la investigación y el análisis del tráfico de red y pruebas de concepto realizadas en capítulos posteriores. Se enuncian los pilares de la seguridad de la información, el modelo TCP/IP, así como algunos términos sobre criptografía, pruebas de penetración y análisis estructurado de tráfico de red.

Además, se define la problemática por la cual nace el trabajo actual; la que se explica en función del uso desinformado de la red Tor y en la fama creada alrededor de este tipo de tecnologías. De igual manera, se propone una solución al problema en forma de una investigación, cuyo objetivo es responder una serie de preguntas base.

Capítulo 2

En el segundo capítulo se presenta el objetivo y funcionamiento de la red Tor, explicando los puntos necesarios para comprender los análisis realizados durante este trabajo, como los circuitos que esta crea, los tipos de nodos que la conforman, la existencia de los servicios ocultos y la importancia del anonimato en las redes de computadoras.

Capítulo 3

En el tercer capítulo se muestra a detalle los requisitos necesarios y el procedimiento utilizado para poner en marcha la infraestructura necesaria para formar parte de la red Tor, considerando las configuraciones de seguridad necesarias para garantizar su buen funcionamiento. De igual manera, se explican las principales dificultades encontradas en este procedimiento.

Capítulo 4

En el cuarto capítulo se explica el proceso que se siguió durante la captura de tráfico de red para, posteriormente, poder profundizar el análisis en el tipo de actividad que desarrollan los usuarios dentro de la red Tor. Este análisis tiene por objetivo determinar qué tanta actividad maliciosa existe a través de esta red específica, y se utilizan técnicas basadas en los nombres de dominio consultados por los usuarios, además de los servicios de capa de aplicación utilizados.

Adicionalmente, se realiza una búsqueda profunda de actividad maliciosa mediante diversas técnicas de análisis estructurado de tráfico de red, en la que se utilizan herramientas de terceros y otras de desarrollo propio.

Capítulo 5

El quinto capítulo se basa en la explicación y ejecución de diversas pruebas de concepto para analizar la seguridad de la información de los datos que viajan a través de la red Tor. Las pruebas que se muestran en este capítulo se realizaron considerando de forma independiente la integridad, confidencialidad y disponibilidad de los datos y conexiones.

Capítulo 6

En el sexto capítulo se muestran las conclusiones obtenidas a partir de los resultados de los diferentes análisis y pruebas de concepto realizados, así como un listado de oportunidades de trabajo futuro relativas a la red Tor.

Capítulo 1 Marco teórico

1.1 Servicios de la seguridad de la información

Generalmente la confidencialidad, integridad y disponibilidad son términos aceptados como los grandes pilares o servicios de la seguridad de la información. Sin embargo, es común encontrar otros términos como parte de estos servicios, entre los que se encuentran la autenticación y el no repudio (Stallings, Cryptography and Network Security, 2006). A continuación se abordan los pilares de la seguridad de la información que servirán de referencia para hablar sobre el tema de la privacidad.

1.1.1 Confidencialidad

Este primer gran pilar consiste en mantener la información accesible únicamente a los usuarios o sistemas autorizados. Es decir, que aquellos que no tengan permitido acceder a la información no puedan hacerlo (Stallings, Cryptography and Network Security, 2006).

1.1.2 Integridad

El segundo gran pilar se refiere a que la información debe poder ser modificada únicamente por los usuarios o sistemas autorizados para hacerlo y, de esta forma, poder asegurar que la información es la que debería ser. Asimismo, el control de la integridad se refiere a que debe ser posible detectar cuando la información ha sido modificada sin autorización en su tránsito o almacenamiento (Stallings, Cryptography and Network Security, 2006).

1.1.3 Disponibilidad

El tercer gran pilar indica que la información debe estar accesible para los usuarios o sistemas autorizados en el momento en que estos requieran acceder a ella (Stallings, Cryptography and Network Security, 2006).

1.1.4 Autenticación

La autenticación valida y asegura que el usuario que interactúa con un sistema realmente es quien dice ser, es decir, que sea auténtico. Sin embargo, no se encarga de validar o proveer protección contra la modificación de los datos (Stallings, Cryptography and Network Security, 2006). Existen al menos tres formas de autenticar a alguien: a través de algo que se sabe (como una contraseña), a través de algo que se tiene (como un token físico o digital) o a través de lo que se es (como

una huella dactilar). De igual manera, es común que se utilicen combinaciones de estos mecanismos para hacer más confiable la autenticación en un método que se conoce como un doble factor de autenticación.

1.1.5 No repudio

El no repudio ayuda a asociar una acción o comunicación a un usuario, sin que este pueda negar su participación en ella (Stallings, Cryptography and Network Security, 2006).

1.2 Modelo TCP/IP

TCP/IP es un modelo de red de computadoras formado por cuatro capas y es utilizado para crear la mayoría de las redes que conocemos hoy en día. Está compuesto por la capa de enlace, la capa de internet, la capa de transporte y la capa de aplicación. Se conoce por ese nombre debido a sus dos protocolos más importantes: IP (*Internet Protocol*) de la capa de internet y TCP (*Transport Control Protocol*) de la capa de transporte (Tanenbaum & Wetherall, 2012), sin embargo, en cada una de las capas funcionan diversos protocolos con diferentes objetivos, en donde los protocolos de las capas más altas se encapsulan en los protocolos de las capas más bajas. De esta forma, los protocolos de capas más altas no tienen que preocuparse por los problemas de las capas inferiores, por lo que podría decirse que las capas inferiores proveen servicios a las superiores.

Puesto que la red Tor es una red superpuesta a internet, depende directamente del funcionamiento de las diferentes capas del modelo TCP/IP. Por lo anterior, explicaré brevemente las características principales de cada capa.

1.2.1 Capa de enlace

Esta es la capa más baja del modelo y en esta capa se observa una combinación entre hardware (p. ej., un adaptador de red) y software (p. ej., un controlador para el adaptador de red) para cumplir su objetivo, el cual es la creación de enlaces físicos entre dispositivos y el acceso a dichos enlaces (Tanenbaum & Wetherall, 2012). Es aquí donde funcionan estándares como Ethernet (IEEE 802.3) para redes alámbricas y Wi-Fi (IEEE 802.11) para redes inalámbricas, pues es en estos donde se especifica la estructura de las tramas que encapsularán a los protocolos de capas superiores.

En esta capa, de acuerdo al estándar IEEE 802.3, es donde existen las direcciones MAC (Media Access Control) o direcciones físicas, que son identificadores únicos a nivel mundial de los adaptadores de red compuestas por seis bytes, en donde la primer mitad se identifica con el fabricante y en la segunda mitad, al dispositivo. Cabe

destacar que, aunque esta dirección está diseñada para ser única en el mundo, puede ser modificada por software.

Cada nodo de la red Tor debe tener al menos una interfaz de red y, por lo tanto, una dirección física para poder comunicarse a través de internet.

1.2.2 Capa de internet

Su objetivo es entregar los paquetes adonde fueron enviados de manera independiente entre sí, pudiendo tomar rutas diferentes y llegar en desorden (Tanenbaum & Wetherall, 2012). El protocolo principal de esta capa es IP (*Internet Protocol*), el cual define la estructura de los paquetes que encapsulan a los protocolos de capas superiores y el direccionamiento IP asignado a los equipos dentro de una red.
A pesar de que IP es el protocolo principal de esta capa, existen otros protocolos complementarios con diferentes objetivos. Por ejemplo, el protocolo ICMP *(Internet Control Message Protoco*l) el cual sirve para el envío de mensajes informativos o de error dentro de una red.

1.2.2.1 Direccionamiento IP

Una dirección IP es un número que identifica a un equipo dentro de una red (Stallings, Comunicaciones y Redes de computadores, 2000). Este identificador, a diferencia de las direcciones MAC, puede no ser único en el mundo. Lo anterior se debe a que un dispositivo puede estar conectado a diferentes redes, públicas o privadas, en diferentes instantes de tiempo y tener una configuración diferente en cada una de ellas. En la red Tor, cada nodo es una computadora que tiene asociada una dirección IP; la cual es utilizada para comunicarse entre nodos, con clientes y con otros servidores conectados a internet.

Existen dos grandes versiones en funcionamiento del protocolo IP: IPv4 e IPv6. La segunda nace producto de una necesidad de poder otorgar direccionamiento a más y más equipos dentro de Internet, pues los números IPv4 actualmente están agotados, es decir, ya todos han sido asignados a diferentes organizaciones. Esto se debe a que las direcciones del protocolo IPv4 están conformadas por 32 bits pudiendo generar 2^{32} direcciones únicas, mientras que las de IPv6 constan de 128 bits pudiendo tener hasta 2^{128} direcciones IP únicas, eliminando el problema de agotamiento de direcciones.

Estos números pueden asignarse a los equipos dentro de una red de forma automática apoyándose de protocolos auxiliares como DHCP (*Dynamic Host Configuration Protocol*) o de forma manual en cada uno de los equipos. Las direcciones IP sirven, no solo para identificar al equipo dentro de una red, sino para

identificar a la red ante otras redes para que estas sean capaces de enviarle paquetes. Esta identificación se logra al separar la dirección IP en dos secciones: una que identifica a la red y otra que identifica a los equipos dentro de ella. Esta separación se implementa mediante un segundo número del mismo tamaño conocido como máscara de red que es operado con la dirección IP mediante una operación lógica AND.

1.2.3 Capa de transporte

Entre las funciones de esta capa está la recuperación ante errores, por ejemplo, al ordenar los paquetes recibidos de la capa de internet para garantizar que en el otro extremo se reciban como fueron enviados. Asimismo, sirve para identificar los procesos de los equipos en comunicación que se encargarán de procesar los datos, esta identificación se realiza a través de un número de dos bytes conocidos como puertos lógicos. El protocolo más utilizado de la capa de transporte es TCP (*Transmission Control Protocol*), sin embargo existen otros protocolos correspondientes a esta capa, entre los que destaca el protocolo UDP (*User Datagram Protocol*) (Stallings, Comunicaciones y Redes de computadores, 2000).

Ambos protocolos, tanto TCP como UDP, hacen uso de los puertos lógicos, que son números representados con 16 bits, es decir, van del 0 al 65535. Estos puertos son clasificados en puertos bien conocidos, puertos registrados y puertos dinámicos. Los puertos bien conocidos van del 0 al 1023 y solamente pueden ser usados por usuarios privilegiados en el sistema operativo. Estos puertos han sido asignados como puertos por defecto por la IANA (*Internet Assigned Numbers Authority*) a varios protocolos como SSH, DNS, FTP, HTTP o SMTP. Los puertos registrados van del 1024 al 49151 y pueden ser utilizados por usuarios sin privilegios. Estos puertos han sido asignados a aplicaciones en específico, como por ejemplo, el manejador de bases de datos *PostgreSQL*. Finalmente, los puertos dinámicos o puertos altos suelen ser los utilizados por los equipos al iniciar una comunicación ya que no suelen ser usados por protocolos o aplicaciones comunes. Cabe destacar que esta asignación de puertos a protocolos sirve para una configuración por defecto, pues realmente un protocolo puede usar el puerto que sea, siempre y cuando el usuario que ejecute la aplicación correspondiente cuente con los privilegios necesarios para realizar dicha reasignación.

1.2.3.1 TCP

TCP (*Transmission Control Protocol*) es un protocolo orientado a conexión, es decir, que únicamente se permite la comunicación entre los dos equipos si se establece una sesión previamente. De igual forma, TCP garantiza que todos los datos enviados son recibidos y en el orden en que fueron enviados gracias a su mecanismo de recuperación de errores en el que el equipo receptor le confirma que recibió el

segmento correspondiente y en caso de no existir una confirmación, este es reenviado. Por lo anterior, se dice que TCP es un protocolo confiable ya que resuelve el problema generado por el protocolo IP en la capa inferior, ya que éste último no es orientado a conexión y no garantiza que los paquetes lleguen en orden o que lleguen al destino. Esto vuelve a TCP un protocolo ideal para aplicaciones que necesitan garantizar la recepción correcta de datos como el envío de correos electrónicos (Tanenbaum & Wetherall, 2012).

Dado que este protocolo implementa mecanismos para garantizar la recepción de lo que ha sido enviado, las cabeceras de los segmentos TCP cuentan con varios campos que lo ayudan a cumplir este objetivo, pero que implican el envío de más datos. Entre estos campos están el puerto origen, puertos destino, un número de secuencia, un número de confirmación, banderas, suma de verificación, etcétera.

1.2.3.2 UDP

UDP *(User Datagram Protocol*) es un protocolo no orientado a conexión, es decir, que no es necesario establecer una sesión previa para el envío y recepción de datos. Este protocolo no establece ningún mecanismo para confirmar la recepción de datos, por lo que no garantiza que estos lleguen o que lleguen en el orden en que fueron enviados, dejando esta tarea a la capa de aplicación en caso de que esta requiera esta característica. Por esta razón, UDP no se considera un protocolo confiable (Tanenbaum & Wetherall, 2012), sin embargo, tiene la ventaja de generar menos tráfico para funcionar, lo que permite un menor uso de ancho de banda. Esto lo vuelve un protocolo ideal para aplicaciones que no requieran la confirmación de recepción de todos los datos y que consumen un gran ancho de banda como la visualización de videos.

Puesto que este protocolo no implementa mecanismos para garantizar la recepción de los datos, los datagramas UDP no necesitan de tantos campos en sus cabeceras, los únicos campos que utiliza es el puerto origen, puerto destino, longitud del datagrama y una suma de verificación.

1.2.4 Capa de aplicación

A esta capa pertenecen los protocolos que son utilizados por los usuarios finales, esto implica que es aquí donde existe la mayor variedad de protocolos. Los datos de las aplicaciones se encuentran encapsulados en los segmentos y datagramas de la capa de transporte. Gracias al modelo TCP/IP basado en capas, los protocolos de esta capa no deben de preocuparse por problemas resueltos en capas inferiores como el envío de señales a través de un medio, la identificación de equipos en una red, las rutas para llegar a otras redes y la garantía de entregar los datos como fueron

enviados. Algunos de los protocolos mayormente usados en la capa de aplicación son DNS, SSH, HTTP, HTTPS y SMTP (Tanenbaum & Wetherall, 2012).

1.2.4.1 Protocolo DNS

DNS (*Domain Name Service*) es indispensable para facilitar el uso de internet a las personas, ya que principalmente sirve para asociar una dirección IP (fácil de entender para las computadoras) a un nombre de dominio (fácil de entender y memorizar para los humanos). Este protocolo no sirve para encontrar la ruta hacia determinado host, sino únicamente para que las personas puedan usar indistintamente una dirección IP o un nombre de host.

El protocolo DNS por defecto utiliza el puerto 53 del protocolo UDP en la capa de transporte pero, dependiendo la situación, utiliza también el protocolo TCP. Por ejemplo, si un servidor DNS va a transferir los dominios que puede resolver a otro servidor DNS (transferencia de zona) el protocolo de capa de transporte a utilizar es TCP (Tanenbaum & Wetherall, 2012).

1.2.4.2 Protocolo SSH

SSH (*Secure Shell*) es un protocolo que sirve para obtener una sesión en un equipo remoto de forma segura, pues todo el contenido entre el cliente y el servidor SSH se encuentra cifrado. Este protocolo sirvió para sustituir a otros protocolos más antiguos y menos seguros como Telnet, ya que estos no cifraban el contenido de la sesión. SSH utiliza por defecto el puerto 22 de TCP en la capa de transporte (Tanenbaum & Wetherall, 2012).

1.2.4.3 Protocolo HTTP

HTTP (*Hypertext Transport Protocol*) es probablemente el protocolo de capa de aplicación más utilizado en nuestros días ya que es el que se utiliza para acceder y navegar por la WWW (World Wide Web). Por ejemplo, si alguien quiere visitar alguna red social a través de un navegador, será a través del protocolo HTTP. Asimismo, es utilizado por prácticamente cualquier aplicación de dispositivos móviles que mande y reciba datos de un servidor a través de una API (*Application Programming Interface*).

Este protocolo consta de varios métodos para su funcionamiento, siendo los dos métodos más utilizados *GET* y *POST*. Asimismo, es común configurar la comunicación entre un cliente y un servidor a través de los llamados encabezados HTTP. Estos encabezados se envían en las peticiones y respuestas HTTP para, por ejemplo, indicar que la comunicación se va a cerrar (*Connection*) o para indicar el tipo de contenido que se está transfiriendo (*Content-Type*). Este protocolo por defecto está asignado al puerto 80 de TCP en la capa de transporte (Stallings, Comunicaciones y Redes de computadores, 2000).

1.2.4.4 Protocolo HTTPS

El protocolo HTTPS es la versión segura del protocolo HTTP, ya que cifra los datos transferidos entre el cliente y el servidor. El formato de las peticiones y respuestas es exactamente el mismo al utilizado en HTTP, con la diferencia de que previo al envío y recepción de cualquier mensaje se establece una conexión cifrada mediante el uso de los estándares SSL (*Secure Sockets Layer*) y TLS (*Transport Layer Security*).

Este protocolo por defecto utiliza el puerto 443 de TCP en la capa de transporte.

1.2.4.5 Protocolo SMTP

SMTP (*Simple Mail transfer Protocol*) es un protocolo que sirve para intercambiar correos electrónicos. Los mensajes utilizados en este protocolo también utilizan encabezados para su correcto envío y recepción, los principales encabezados sirven para indicar la dirección del destinatario, la dirección del remitente y el asunto del mensaje.

Este protocolo no sirve para enviar mensajes entre dos equipos directamente, sino que estos mensajes deben enviarse primero a un servidor SMTP. Esto se debe a que, entre otras razones, el equipo destino no siempre se va a encontrar prendido o conectado a la red. Debido a lo anterior, el mensaje primero es enviado al servidor que lo almacena hasta que el destinatario es capaz de recibirlo. Este protocolo por defecto utiliza el puerto 25 de SMTP en la capa de transporte (Stallings, Cryptography and Network Security, 2006).

1.3 Criptografía y sus aplicaciones

La criptografía es la rama del conocimiento que sirve para generar una versión ininteligible (criptograma) de un mensaje o información almacenada (texto en claro) mediante una llave y un algoritmo de cifrado, de tal modo que esta información solamente pueda ser comprendida por aquellos que conozcan la llave y el algoritmo de descifrado y no por cualquier otra persona que se encuentre espiando sus comunicaciones. Los algoritmos criptográficos deben ser lo suficientemente robustos para asegurar que no sea posible obtener el texto en claro si únicamente se conoce el algoritmo pero no la llave de cifrado.

Cabe resaltar que la criptografía ha sido utilizada a lo largo de la historia, pues desde la época de los romanos era necesario ocultar principalmente información militar. Esto se lograba con algoritmos que sustituían o transponían los caracteres de un mensaje, lo cual se conoce hoy como algoritmos de cifrado clásicos. Sin embargo, actualmente existen algoritmos más avanzados que pueden ocultar la información manipulándola

en su forma más básica: los bits. Gracias a esto, la criptografía ha sido de gran utilidad para la seguridad de la información por ayudar a proteger principalmente pilares como la integridad y la confidencialidad de la información, así como otros servicios como la autenticación (Stallings, Cryptography and Network Security, 2006).

Existen dos grandes tipos de algoritmos criptográficos: los algoritmos de llave secreta y los algoritmos de llave pública. Estos se clasifican de esta manera dependiendo de si la llave usada para cifrar los datos puede ser pública o si solo puede ser conocida por los integrantes de la conversación.

La criptografía es de gran importancia para el funcionamiento de la red Tor, ya que las comunicaciones entre los clientes y los diferentes nodos se encuentran cifradas para proteger la integridad y la confidencialidad de los datos.

1.3.1 Criptografía de llave secreta

Este tipo de criptografía, también conocida como criptografía simétrica, depende exclusivamente de una llave secreta, conocida comúnmente como llave secreta. Esta llave es conocida tanto por el remitente como por el destinatario de un mensaje y, por lo tanto, debe ser acordada previamente al intercambio de mensajes. (Stallings, Cryptography and Network Security, 2006)

Los algoritmos de llave secreta modernos suelen ser muy rápidos y óptimos para cifrar grandes volúmenes de datos. Sin embargo, no son capaces de resolver el problema del intercambio de llaves por su propia cuenta, teniendo que apoyarse de otras ramas de la criptografía para resolver este problema.

Algunos de los algoritmos de criptografía simétrica modernos son DES (*Data Encryption Standard*), 3DES (*Triple DES*) y AES (*Advanced Encryption Standard*). Estos suelen ser muy útiles para cifrar el contenido de los dispositivos de almacenamiento como discos duros, por las grandes cantidades de datos que pueden contener. Asimismo, son utilizados para crear las versiones seguras de protocolos de redes, como HTTPS. Cabe resaltar que, hoy en día, tanto DES como 3DES son considerados inseguros a pesar del gran uso que llegaron a tener, por lo que ante cualquier circunstancia se recomienda el uso de otros algoritmos más seguros como AES.

1.3.2 Criptografía de llave pública

Este tipo de criptografía, también conocido como criptografía asimétrica, utiliza siempre dos llaves que están relacionadas matemáticamente: una privada y una pública, donde cualquiera de las dos llaves puede descifrar lo cifrado por la otra llave.

Para que un remitente y un destinatario puedan intercambiar y comprender mensajes cifrados con algún algoritmo de llave pública, el destinatario debe generar una llave privada y una pública y distribuir la llave pública a cualquier persona que quiera enviarle un mensaje cifrado. El remitente cifraría el mensaje con esta llave y únicamente podría ser descifrado con la llave privada, que está en posesión del destinatario (Stallings, Cryptography and Network Security, 2006). Gracias a esta forma de funcionar, estos algoritmos resuelven el problema del intercambio de llaves existente en la criptografía de llave secreta.

La criptografía de llave pública no solo sirve para proteger la confidencialidad de los mensajes, sino también como medio de autenticación. Por ejemplo, si alguien desea comprobar a los demás que es quien dice ser, puede cifrar un mensaje con su llave privada (que únicamente esa persona conoce). Así, si el mensaje se puede descifrar con la llave pública (que cualquiera puede conocer), significa que efectivamente se trata de la persona que dice ser.

Probablemente el algoritmo más popular de este tipo de criptografía es RSA (*Rivest-Shamir-Adleman*), sin embargo, no es el único algoritmo de cifrado y firma digital, ya que también los algoritmos de Curvas elípticas entran en esta clasificación, funcionando tanto para cifrar datos como para crear firmas digitales.

Aunque estos algoritmos asimétricos eliminan el problema del intercambio de llaves, generalmente no son buenas opciones para cifrar grandes volúmenes de datos pues suelen ser muy lentos. Por esta razón es común que se trabaje con una combinación de criptografía simétrica y asimétrica en la que se utiliza un algoritmo de criptografía asimétrica para intercambiar la llave del algoritmo de criptografía simétrica, el cual finalmente es usado para proteger las comunicaciones.

1.3.3 Algoritmos digestores

Los algoritmos de digestión, o funciones hash, forman parte de una rama de la criptografía que no se usa para cifrar o descifrar conjuntos de datos, sino para obtener un identificador o firma de ellos. Son funciones reciben como entrada un conjunto de datos, como un mensaje o un archivo, de cualquier tamaño y entregan como salida una cadena de caracteres de un tamaño definido conocida como valor hash.

Estas funciones son de un solo sentido, es decir, a partir de un valor hash no es posible obtener el mensaje a partir del cual fue calculado. Esta característica los separa de los algoritmos de otras ramas de la criptografía, en los cuales es posible descifrar un criptograma y obtener el mensaje original en texto claro si se conoce la llave. Esto los vuelve muy útiles para mantener la confidencialidad de la información en situaciones muy particulares. Por ejemplo, al almacenar contraseñas en una base de datos se recomienda nunca almacenar la contraseña como tal, sino su valor hash.

De esta forma, si se llegan a filtrar estos valores debido a un ataque informático, las contraseñas de los usuarios no serían filtradas y no se podría obtener la contraseña original directamente del valor hash.

Otra característica muy importante de estos algoritmos es que, si cambia aunque sea un bit en la entrada, la salida es totalmente diferente. Esta cualidad los vuelve muy útiles para proteger la integridad de la información, no porque se evite la modificación no autorizada de la misma, sino porque permite detectar cuando la información ha sido modificada. Esta cualidad es muy útil para que la salida pueda ser utilizada como un identificador de los datos de entrada.

Existen diversos algoritmos de digestión, algunos de los más populares son MD5 (*Message Digest 5*), SHA1 (*Secure Hash Algorithm 1*) y SHA256 (*Secure Hash Algorithm – 256 bits*). A pesar de su gran popularidad, hoy en día algoritmos como MD5 y SHA1 se consideran inseguros, pues es computacionalmente viable generar colisiones al utilizarlos, es decir, es posible generar dos o más mensajes cuyo valor hash sea exactamente igual, por lo que ya no se podría garantizar que la información no ha sido modificada. Sin embargo, cabe resaltar que es imposible evitar que cualquier función hash tenga colisiones. Esto se debe a que estas funciones aceptan como entrada mensajes de cualquier tamaño y entregan como salida cadenas de un tamaño definido. Dicho de otra forma, el conjunto de posibles entradas tiene una cardinalidad infinitamente mayor que el conjunto de posibles salidas, haciendo imposible la inexistencia de colisiones en cualquier función hash. Esto implica que la confianza depositada en los algoritmos digestores depende exclusivamente de la inviabilidad de encontrar colisiones en ellos con la capacidad de cómputo actual. Por ejemplo, en algoritmos más robustos como SHA256 se podría tardar miles de años en encontrar una colisión. (Stallings, Cryptography and Network Security, 2006).

1.4 Amenazas y vulnerabilidades de la seguridad de la información

Es común que algunos términos utilizados en la jerga de la seguridad de la información sean confundidos y empleados de forma incorrecta; entre ellos están las vulnerabilidades y las amenazas. Una amenaza se refiere a una situación que podría potencialmente causar daño a los activos de información (Acunetix, 2017), estas pueden ser tanto físicas como lógicas. Un ejemplo de una amenaza física es un incendio en los centros de datos de una organización, mientras que un ejemplo de amenaza lógica es la extracción de información sensible, como contraseñas de una organización a manos de un trabajador disgustado. Por otro lado, una vulnerabilidad es una falla o debilidad en un sistema que potencialmente podría ser aprovechada o explotada por una amenaza.

Otro término es el de actor de amenaza y se refiere a la persona o grupo de personas que llevan a cabo una amenaza para dañar los activos de información de una persona u organización. Grupos de cibercriminales y grupos de activistas (*hacktivistas*) son solo algunos ejemplos de posibles actores de amenaza.

Dado que el objetivo de esta tesis incluye determinar cuánta actividad maliciosa existe en la red Tor, desde el punto de vista de la seguridad de la información, cabe destacar algunos ejemplos de amenazas que potencialmente podrían encontrarse en esta red. A continuación, se muestran ejemplos de amenazas a la seguridad de la información y algunas de las vulnerabilidades que estas pueden aprovechar.

1.4.1 Malware

Se puede considerar como malware o programa malicioso a cualquier ejecutable o programa cuyo objetivo sea dañar de alguna forma a los activos de información de una persona u organización. Estos pueden aprovecharse de diversas vulnerabilidades, como el uso de contraseñas débiles para el acceso a los equipos o vulnerabilidades generadas durante la programación de un servicio, como un desbordamiento de búfer, que permitirían eventualmente tomar control de un equipo (Sikorski & Honig, 2012). Existe una amplia variedad de tipos de malware, algunos de ellos se mencionan a continuación.

1.4.1.1 Downloader

Su único objetivo es que, una vez ejecutado, descargue más malware. Es muy útil para los atacantes cuando logran conseguir un acceso inicial a un sistema. Generalmente se trata de programas sencillos que sirven como medio para realizar una infección más robusta en el sistema afectado (Sikorski & Honig, 2012).

1.4.1.2 Gusano

Tipo de malware con la capacidad de replicarse a sí mismo a través de una red de computadoras sin la necesidad de la interacción directa de los usuarios. Generalmente se aprovecha de alguna vulnerabilidad de ejecución de código existente en los equipos afectados (Sikorski & Honig, 2012).

1.4.1.3 Bot

Tipo de malware que, una vez que ha infectado a su víctima, recibe las instrucciones a realizar en el equipo víctima desde otro equipo conocido como Comando y Control (*CC, C2 o C&C*). Estos programas maliciosos generalmente son planeados para infectar grandes cantidades de equipos para formar una red de bots o *botnet* y, de esta forma, tener la posibilidad de realizar otras acciones a gran escala como minería

de monedas criptográficas o ataques de denegación de servicios (Sikorski & Honig, 2012).

1.4.2 Phishing

El activo de información que explota esta vulnerabilidad son las personas que conocen la información. Su objetivo es utilizar técnicas de ingeniería social para engañar a las personas mediante mensajes y sitios falsos, por medio de los cuales convencen de entregar sus credenciales de acceso a diferentes sistemas. Además, es posible engañarlas para instalar malware en su equipo. Algunos tipos de phishing se explican a continuación (Avast, 2018).

1.4.2.1 Phishing genérico

Es el tipo más común, generalmente se distribuyen mensajes genéricos a una gran cantidad de personas esperando que algunas de ellas caigan y entreguen su información. Por esta razón, es muy fácil detectarlos y evitar caer en la trampa (Avast, 2018).

1.4.2.2 Phishing dirigido

Este tipo de phishing va dirigido a personas específicas y no utiliza mensajes genéricos, sino mensajes que tienen que ver con las actividades de la víctima. Requiere que el actor de amenaza que lo lleva a cabo invierta tiempo en investigar a su víctima y de planear el momento para entregar el mensaje. Esto los vuelve muy difíciles de detectar, tanto por la víctima como por herramientas de detección automatizadas (Kaspersky, 2018).

1.4.2.3 Pharming

Este tipo de phishing no necesariamente depende de engañar a las personas a través de mensajes falsos. En su lugar, el actor de amenaza se asegura que cuando la víctima intente acceder a un sitio legítimo, realmente acceda al sitio falso. Esto lo pueden lograr, por ejemplo, mediante una modificación en los registros de los servidores DNS consultados por la víctima (Avast, 2018).

1.4.3 Man in the middle

Una técnica de *man in the middle* u hombre en el medio, se refiere a una persona que se encuentra espiando las comunicaciones establecidas entre dos equipos. Esta amenaza existe debido al uso de comunicaciones no cifradas, ya que esto le permite al actor de amenaza conocer información confidencial o, incluso, modificar los datos

transmitidos, pudiendo afectar tanto a la confidencialidad como a la integridad de la información (Kaspersky, 2013).

1.4.4 Denegación de servicio

Una denegación de servicio se refiere a la incapacidad de un servidor de ofrecer respuesta ante las peticiones realizadas por los clientes, ya sea por algún error en su lógica de programación o por una mayor cantidad de mensajes recibidos a la que es capaz de responder. Esta amenaza atenta principalmente contra la disponibilidad de la información.

Aunque existen múltiples casos en los que un paquete mal formado puede hacer que un servicio falle, la forma más común de materializar esta amenaza es mediante el envío de una gran cantidad de mensajes, lo suficientemente grande como para saturar el canal de comunicaciones o como para que el servidor no pueda procesar todo el tráfico recibido.

A pesar de que, en ciertos casos, aún es viable para los actores de amenaza realizar ataques de denegación de servicio (*DoS*), cada vez se vuelve menos común ver a un único equipo tratando de realizar el ataque, ya sea por la gran cantidad de tráfico a generar o por la facilidad de bloquear a un único equipo atacante. En su lugar, hoy en día es más común realizar un ataque de denegación de servicio distribuido (*DDoS*), en el que múltiples equipos saturan el canal de comunicaciones o el servidor. Una de las formas más comunes en las que hoy en día estos ataques se realizan es mediante *botnets* controladas por los actores de amenaza y que pueden generar enormes cantidades de tráfico proveniente de muchas partes del mundo, dificultando la separación de las peticiones legítimas de las maliciosas (Stallings, Cryptography and Network Security, 2006).

1.4.5 Watering Hole

El nombre *Watering Hole* hace una analogía con un estanque para beber agua, al que se acercan diversos animales para hidratarse, como cebras o antílopes, sin embargo, en este estanque que forma parte de las actividades cotidianas del animal existe una amenaza que puede atentar contra su vida, como un cocodrilo.

Llevando la analogía a la seguridad de la información, se refiere a que un actor de amenaza puede comprometer los activos de información de una persona u organización mediante un sitio en el que los dueños de dichos activos confían. Es decir, que mediante un compromiso inicial de un servidor comúnmente utilizado por diversos usuarios, un actor de amenaza puede simplemente esperar a que sus usuarios lo utilicen y engañarlos para que descarguen y ejecuten archivos maliciosos o entreguen información confidencial. En este caso, la vulnerabilidad de la cual se

aprovecha esta amenaza es nuevamente la forma de pensar de los humanos (Tarlogic, 2016).

1.5 Análisis estructurado de tráfico de red

El análisis estructurado de tráfico de red es un conjunto de técnicas para conocer y comprender lo que sucede en las comunicaciones existentes en una red. Es de especial utilidad para detectar intrusos en una red durante el monitoreo de seguridad, durante la respuesta a incidentes de seguridad de la información y durante investigaciones forenses, ya que la cantidad de tráfico que en estos casos debe analizarse suele ser muy grande (Bejtlich, 2005).

A lo largo del trabajo se mostrará el uso de diversas técnicas de análisis estructurado de tráfico de red, con el fin de encontrar información de utilidad que ayude a cumplir el objetivo de la tesis. Por ejemplo, estadísticas de conexiones en el nodo Tor, conteo de flujos, el análisis del contenido de los paquetes capturados y el uso de sistemas de detección de intrusos para hallar actividad maliciosa.

1.5.1 Análisis estadístico

Suele ser una aproximación inicial para obtener una idea general de lo que sucede en una red. Por ejemplo, sirve para obtener métricas sobre los protocolos más utilizados en las diferentes capas, así como la duración de la captura de tráfico, el número de conexiones, etcétera. Es muy útil para poder determinar la naturaleza de la captura de tráfico y ayuda a determinar por dónde iniciar una investigación y, de esta forma, optimizar el análisis.

1.5.2 Análisis de flujos

Este análisis es útil para aclarar más la naturaleza del tráfico analizado sin necesidad de ver el contenido de los paquetes, ya que es posible ver un resumen de las direcciones IP y cuantas sesiones o flujos han sido establecidas por determinadas direcciones IP origen o destino. Dicho de otra forma, es posible analizar qué orígenes se comunican con qué destinos y con qué protocolos.

1.5.3 Análisis de datos de contenido

Una vez que se tiene una idea más clara de los orígenes y destinos existentes en la red, y en caso de que se tenga acceso a una captura completa de tráfico de red, es posible encontrar más fácilmente qué paquetes son los que se deben analizar más detenidamente. En el análisis de datos de contenido se tiene acceso a todos los bytes enviados y recibidos, por lo que se puede determinar exactamente qué peticiones

fueron enviadas, su objetivo y la respuesta correspondiente. Por ejemplo, al analizar todo el contenido enviado puede determinarse fácilmente si se está transmitiendo información confidencial o inapropiada.

1.5.4 Análisis para la detección de intrusos

Este tipo de análisis sirve para generar alertas y con ellas poder detectar y/o prevenir la presencia de intrusos. Normalmente esta detección se realiza mediante una serie de reglas predefinidas que, al aplicarse en un sistema de detección automatizado y compararse con el tráfico que circula por la red, sirven para identificar tráfico sospechoso y potencialmente malicioso, como escaneos en la red e intentos de explotación de vulnerabilidades. Este análisis, dependiendo del nivel de detalle establecido en las reglas, es muy susceptible a caer en falsos positivos y falsos negativos, es decir, que genere alertas sobre tráfico sospechoso que realmente forma parte de la actividad normal de la red o que no genere alertas cuando realmente sí hay tráfico malicioso. Debido a esto, estas reglas deben ser mantenidas constantemente para agregar reglas o modificar las existentes dependiendo de las nuevas vulnerabilidades que podrían ser explotadas en la red a proteger.

1.6 Privacidad y anonimato en redes

La privacidad y anonimato son dos términos constantemente intercambiados, sin embargo y a pesar de que están relacionados, su significado es distinto. Si se consulta en el diccionario de inglés de Cambridge las palabras “*privacy*” y “*anonymity*”, resulta que el anonimato se refiere a una situación en la que una persona no es conocida mediante su nombre, es decir, que es no posible identificarla. Por otro lado, la privacidad se refiere a un derecho de las personas a mantener sus relaciones y asuntos personales en secreto.

Este par de términos son comunes al hablar sobre el Internet y las redes en general, pues estas son un medio para, potencialmente, vulnerar tanto a la privacidad como al anonimato de sus usuarios. Por ejemplo, al autenticarse con un usuario y una contraseña en una red social se le está otorgando información sobre quién es y quién la está usando. Dicho de otra forma, se está perdiendo el anonimato ante la red social. Por otro lado, esta red social puede estar llevando un registro sobre los gustos de sus usuarios, sus búsquedas, los lugares que han visitado, la cantidad de mensajes que envían por día, etcétera. Es decir, se le está otorgando libertad para revisar los datos personales que deberían ser privados.

Es importante definir y comprender la diferencia de estos dos términos y su relación con las redes de computadoras pues, debido al aumento en la conectividad que experimentan las personas hoy en día, cada vez es más común que se preocupen por proteger su privacidad y mantenerse anónimos ante actores externos, eligiendo

diversas tecnologías para lograrlo. Por ejemplo, puede ser que los usuarios contraten un servicio de VPN (*Virtual Private Network*) para mantenerse anónimos, sin embargo, ante los ojos de los proveedores de estos servicios no se es anónimo pues es necesario autenticarse previamente para acceder a ellos y para ellos es perfectamente posible rastrear una actividad a uno de sus usuarios si es que mantienen bitácoras de sus actividades. En este caso, la conexión a través de la VPN es especialmente útil para mantener privados los datos personales, pues protegen las comunicaciones mediante túneles cifrados. Por otro lado, tecnologías como Tor buscan mantener anónimos a sus usuarios y para lograrlo dependen de que exista una gran cantidad de nodos dándole funcionalidad a la red y una gran cantidad de personas utilizándola, de tal forma que se dificulte relacionar una acción a determinado usuario. El funcionamiento de esta red se profundiza en los capítulos posteriores.

1.7 Planteamiento del problema y solución propuesta

1.7.1 Problemática actual

La sociedad actualmente ha sufrido muchos cambios en la forma de realizar prácticamente cualquier actividad gracias a la computación y a las redes de computadoras. Estas han facilitado a las personas compartir todo tipo de información, acceder a todo tipo de recursos en línea y a comunicarse entre sí de forma prácticamente instantánea. Aunque podría parecer que estos avances tecnológicos implican únicamente ventajas, realmente han permitido también generar nuevos métodos de espionaje masivos, nuevas formas de censurar las opiniones de las personas y nuevas técnicas para generar un perfil de los usuarios de la red basándose en sus hábitos al usarla.

Por lo anterior, muchas personas comunes y corrientes se preocupan más que nunca por mantener sus datos privados, alejándolos de las grandes empresas que los usan para crear y distribuir publicidad muy específica de acuerdo a los perfiles de cada persona. Adicionalmente, hay personas que no solo quieren proteger sus datos, sino que tienen una necesidad de mantenerse anónimos en la red para evitar una censura a su forma de pensar y de expresarse. En este grupo pueden entrar los líderes o partidarios de movimientos sociales, periodistas que tocan temas sensibles, habitantes de países con un alto índice de censura digital, entre otros.

Producto de esta preocupación por mejorar su privacidad y obtener anonimato, muchas personas se interesan en las redes superpuestas a internet como Tor para lograrlo. Sin embargo, en muchas ocasiones las usan bajo la creencia de que estas redes son invulnerables a ser analizadas, por lo que no tienen cuidado al utilizarlas.

Adicionalmente, hay quienes se dejan llevar por la fama que distintos medios y la cultura pop ha creado sobre ellas, creyendo que son utilizadas principalmente por <<*hackers*>> y delincuentes, por lo que no se atreven a utilizarlas. Es decir, creen que el principal uso que se les da es malicioso o ilegal.

Además de las personas que únicamente tienen conocimiento de la existencia de la red, existen otras que comprenden la naturaleza de estas redes y que podrían interesarse en apoyarlas, pero no conocen cuál es el proceso técnico para donar ancho de banda y tiempo de cómputo para fortalecerlas.

1.7.2 Solución al problema.

Para ayudar a desmentir o confirmar las creencias existentes sobre la utilidad de la red Tor, así como del funcionamiento y seguridad de sus comunicaciones, se propone este trabajo de investigación en donde se analiza el comportamiento de esta red y de la seguridad que provee a los datos que transitan a través de ella mediante el uso de diversas técnicas de análisis de tráfico y pruebas de concepto de seguridad de la información.

Del mismo modo, la instalación y puesta en marcha de la infraestructura necesaria para realizar la investigación servirá como una guía para los entusiastas que deseen apoyar a la red Tor mediante contribuciones de tiempo de procesamiento y ancho de banda, pero que actualmente no cuentan con los conocimientos técnicos para llevarlo a cabo.

Como apoyo para realizar la investigación se ha formulado las siguientes preguntas base, que serán respondidas con los resultados mostrados en este trabajo:

- ¿Qué tan complejo es apoyar a la red Tor mediante la administración de un nodo perteneciente a ella?
- ¿En qué medida es usada la red Tor para realizar ataques a los activos de información y usuarios de Internet?
- ¿En qué medida es viable para un administrador de un nodo de salida espiar las comunicaciones de los usuarios de la red Tor, es decir, vulnerar su confidencialidad?
- ¿En qué medida es viable para un administrador de salida modificar las comunicaciones de los usuarios de la red Tor, es decir, vulnerar su integridad?
- ¿Qué tan robusta es la red Tor en términos de la disponibilidad de sus conexiones?

Capítulo 2 La red Tor

2.1 Objetivo

El objetivo de la red Tor es ayudar a las personas a obtener anonimato y a mejorar la privacidad de su información mientras usan internet. Esto lo logra implementando medidas para evitar el análisis del tráfico de red y, de esta forma, prevenir que un tercero conozca con quién habla y qué dice determinado usuario de internet.

Gracias a los beneficios que brinda esta red, es posible ayudar a evitar eficazmente la censura que implementan ciertos países u organizaciones. Tor puede servir como herramienta de anonimato tanto a clientes, como a servicios en la red (The Tor Project, 2018).

2.2 Funcionamiento

La red Tor está formada por un grupo de servidores o nodos con direccionamiento IP público alrededor de todo el planeta. Estos nodos son provistos principalmente por voluntarios que simpatizan con la ideología del proyecto. En estos servidores se instala y configura el software necesario para que se interconecten y formen la red Tor. Por lo tanto, Tor puede verse como una red superpuesta a internet con sus propios algoritmos de enrutamiento (Electronic Frontier Foundation, 2018).

Para poder utilizar Tor como medio de comunicación, es necesario utilizar un software especial capaz de conectarse a la red. El programa más común para lograr esto es el navegador Tor; al tratarse de un navegador, sirve particularmente para utilizar el protocolo de aplicación HTTP. Sin embargo, es posible utilizar la red para enviar todo tipo de tráfico TCP.

2.2.1. Circuitos

Las conexiones realizadas por un cliente a través de la red Tor siguen una ruta generada de forma aleatoria con tres nodos, utilizando los servidores de voluntarios como dichos nodos en el túnel virtual o circuito. De esta forma, la red oculta el origen de las conexiones, pues cada nodo conoce únicamente la dirección IP del punto anterior y del punto siguiente, pero no el origen y destino del circuito entero. Es decir, ningún eslabón de la cadena es capaz de conocer la ruta completa seguida por las conexiones, imposibilitando el rastreo del usuario de la red Tor que lo está usando.

Dicho de otra forma, cada uno de los nodos actúa como un proxy al modificar las cabeceras del protocolo IP, en específico las direcciones IP origen y destino. Gracias

a esto, el nodo de entrada a la red Tor conocerá únicamente las direcciones IP del cliente y del nodo intermedio, mientras que el nodo de salida de la red Tor conocerá únicamente del nodo intermedio y del sitio visitado en internet. (Dingledine & Mathewson, 2018). Un diagrama de las conexiones hechas en la red Tor se muestra en la imagen 2.1.

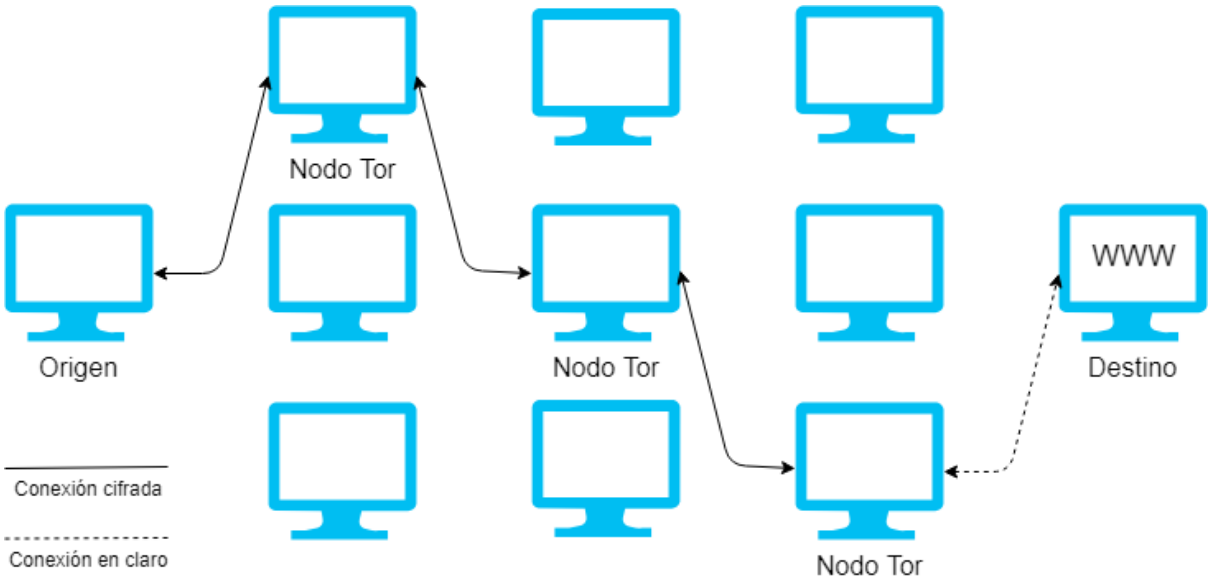

Imagen 2.1. Diagrama de conexiones en la red Tor.

El planteamiento anterior implica que el nivel de anonimato que puede proveer Tor es directamente proporcional a la cantidad de usuarios activos en la red. Esto se debe a que múltiples usuarios pueden estar haciendo uso de uno o más nodos del circuito de forma simultánea. Entre más usuarios existan en la red, más complejo se vuelve el relacionar a determinado usuario con determinada actividad. Por el contrario, si la red cuenta con un número muy bajo de usuarios, alguien con la capacidad de observar la actividad en la red (como los proveedores de servicios de internet) podría hacer análisis estadísticos y correlaciones para determinar qué usuarios han hecho qué cosas.

Debido a que Tor usa como método de enrutamiento los circuitos, las rutas usadas por los clientes están lejos de ser las más óptimas en cuanto a tiempo y velocidad. Razón por la cuál es considerablemente más lento navegar usado Tor que sin usarla. Sin embargo, para hacerlas un poco más óptimas, las rutas se reutilizan por un tiempo determinado y posterior a eso se genera un nuevo circuito. Sin embargo, si un usuario o aplicación desea obtener una nueva ruta o identidad, es posible solicitarla antes de que se llegue al tiempo límite de diez minutos.

Para mantener las comunicaciones confidenciales durante su tránsito por el circuito, Tor hace uso de diversos algoritmos de cifrado que ayudan a cumplir este objetivo, entre los que destacan AES256 y RSA. Para lograrlo, el cliente se encarga de negociar llaves diferentes para cada uno de los nodos pertenecientes al circuito,

posteriormente a los datos enviados por el circuito se le aplican 3 capas de cifrado utilizando las llaves públicas de cada nodo. Dicho de otra manera, entre el cliente y los tres nodos que conforman el circuito, los datos se encuentran cifrados y cada capa de cifrado es retirada por cada nodo, manteniendo cifrado el tráfico desde el cliente hasta el último nodo del circuito, siendo el nodo de salida el que se encarga de obtener el mensaje en claro para poder ser interpretado por el destino original (Dingledine & Mathewson, 2018). Esto se ejemplifica en la imagen 2.2.

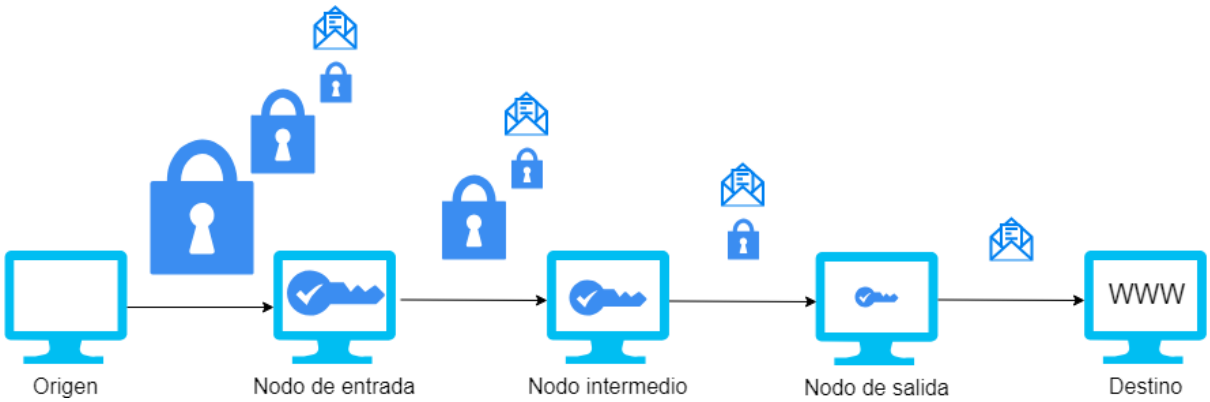

Imagen 2.2. Capas de cifrado removidas por cada nodo en el circuito.

2.2.2. Proceso de consenso

Se le llama “consenso” a una lista constantemente actualizada de todos los nodos que forman a la red. Esta lista es mantenida por nodos conocidos como “Autoridades de directorio”, que son administrados por voluntarios de confianza y que se encuentran en diferentes partes del mundo; tiene como objetivo que todos los clientes tengan información confiable y actualizada sobre los nodos existentes en la red Tor. Existen 10 de estas autoridades, y ellas se encargan de decidir si un nodo es o no válido. Para que cualquier cliente conozca cuáles son las autoridades de directorio, su información se encuentra embebida en su código fuente (Jordan, 2015).

Las autoridades actualizan la lista de todos los nodos y su estado cada hora mediante una votación. A grandes rasgos, el proceso para actualizar esta lista es el siguiente:

- Cada autoridad recopila una lista de todos los nodos conocidos.
- Cada autoridad agrega la información faltante de estos nodos, como sus banderas.
- Se envía esta información recopilada a las demás autoridades como un voto de estado o “*status-vote*”.
- Cada autoridad consigue los votos de las demás autoridades.
- Toda la información de los nodos de cada voto es combinada y firmada por cada autoridad.
- La mayoría de las autoridades de directorio debe estar de acuerdo con la información, validando así el nuevo consenso.
- El consenso es publicado por cada autoridad a través del protocolo HTTP.

De las diez autoridades de directorio, existen 9 dedicadas a validar el consenso y mantener la lista de nodos públicos. Al mantener un número non de autoridades, siempre es posible que una mayoría valide el nuevo consenso. La autoridad de directorio restante se encarga de mantener la lista de nodos puentes. Gracias a este proceso, cualquier cliente que se conecte a la red puede tener información confiable sobre los nodos pertenecientes a la red y sus características. (Jordan, 2015)

Cabe destacar que cada nodo dentro de la red tiene asignado un "peso de consenso" que se trata de un valor basado en el ancho de banda observado en el mismo, dicho valor es publicado en el consenso cada hora. El propósito del peso de consenso es que los clientes puedan usarlo para seleccionar los nodos que formarán sus circuitos.

2.3 Tipos de nodos

Para que la red Tor funcione, necesita de tres grandes tipos de nodos, ya que cada uno cumple con una función en especial al formar los circuitos. Estos son: nodos intermedio y de entrada, nodos de salida y puentes. (The Tor Project, 2018). Adicionalmente, cada nodo sin importar su clasificación tiene determinadas propiedades como "*Fast*" o "*Stable*" que son asignadas por las autoridades de directorio.

2.3.1 Nodo intermedio y de entrada *(guard/ middle relay)*

Un nodo de entrada corresponde al primer salto que se da para para acceder a la red Tor y para convertirse en uno es necesario que sea estable y rápido (al menos 2 MBytes por segundo). Si no puede cumplir estas condiciones, se mantendrá como un nodo intermedio, el cual corresponde al segundo salto en el circuito.

Estos nodos son los que se encargan de proveer la mayor cantidad de ancho de banda a la red y son los que le otorgan mayor robustez, pues representan a la mayoría de los nodos. Estos están constantemente haciéndose públicos dentro de la red para ser capaces de generar los circuitos de forma efectiva y, al no ser de salida, no son usados para parecer ser el origen del tráfico. Es decir, el destino final de la comunicación no verá en sus bitácoras los datos de estos nodos y, debido a esto, estos nodos no suelen recibir quejas originadas por el tráfico de la red Tor.

Otra característica de estos nodos es que los datos que reciben y envían siempre se encuentran cifrados, por lo que no es posible para ellos espiar las comunicaciones entre los clientes y sus destinos.

2.3.2 Nodo de salida (*exit relay*)

Es el último punto a donde llega el tráfico antes de salir a la red pública, pues son estos nodos los que se encargan de enviar los datos a donde el usuario originalmente quería enviarlos. Al igual que los nodos intermedios, los de salida se encuentran de forma regular indicando su presencia en la red para poder ser usados como parte de los circuitos virtuales.

Puesto que los nodos de salida se encargan de sacar el tráfico de la red Tor y dirigirlo a los destinos originales, son los datos de estos nodos los que se registran constantemente en las bitácoras de acceso en los diferentes servidores de internet. Asimismo, es necesario que estos nodos remuevan la última capa de cifrado del tráfico, dejándolo en claro para que pueda ser interpretado y utilizado por el destino original de la comunicación. Debido a lo anterior, los nodos de salida son susceptibles al análisis de tráfico que no es posible realizar en los intermedios y de entrada.

2.3.3 Puente (*bridge*)

Son nodos que forman parte de la red Tor, pero que no son publicados como tal. Este tipo de nodos es importante para poder evadir los diferentes mecanismos de censura que implementan organizaciones o países. Esto se debe a que las listas de nodos intermedios y de salida son públicas, por lo que sus direcciones IP pueden ser bloqueadas y, de este modo, bloquear las conexiones a la red Tor. Sin embargo, como los puentes no están registrados en el directorio público de Tor, hace más difícil a los gobiernos y proveedores de servicios el bloquear el acceso a esta red.

Las listas de estos nodos son mantenidas por nodos especiales conocidos como autoridades de puentes (*bridge authorities*) para posteriormente utilizar otros mecanismos para distribuir sus datos.

2.4 Servicios ocultos

Los servicios ocultos, también conocidos como *servicios cebolla* son servicios de capa de aplicación como páginas web, correo o mensajería instantánea a los que se puede acceder exclusivamente a través de Tor. Estos servicios pueden estar alojados en cualquier nodo de la red Tor y, al igual que sucede con los clientes, sirven para ocultar la dirección IP del nodo que lo está ejecutando. De igual manera, ofrecen una alternativa para acceder a servicios evadiendo restricciones como reglas de firewall.

Puesto que los servicios ocultos se alojan en nodos de la red Tor, para acceder a ellos no es necesario pasar por un nodo de salida. Esto implica que, al no salir de la red Tor, no es posible analizar el tráfico de red destinado a un servicio oculto ya que

siempre se encuentra cifrado, a diferencia de como sucede con servicios convencionales que es necesario descifrar el tráfico en un nodo de salida.

Debido a que los servicios ocultos sirven para mantener oculta la dirección IP de la computadora que los ejecuta, es necesaria la implementación de un sistema de nombres similar a DNS para poder acceder a ellos. A pesar de no tratarse propiamente del servicio DNS, es similar a este pues permite acceder a un servicio utilizando un nombre en lugar de una dirección IP (The Tor Project Metrics, 2018).

Los nombres se asignan de forma aleatoria al configurar el servicio cebolla, lo que implica que generalmente el nombre resultante es difícil de memorizar para un humano. Estos nombres siempre terminan en el dominio de nivel superior ".onion" y, como no están mantenidos en el sistema de nombres de dominio convencional, solo pueden conocerse si se hacen públicos de forma expresa. Es decir, no es posible utilizar las bases de datos de la ICANN para conocer a quien pertenece este nombre y, debido a esto, se suelen publicar los dominios .onion en blogs o sitios como *reddit*.

2.5 Anonimato en redes

A muchas personas hoy en día les importa, aunque sea un poco más que hace pocos años, mantener su identidad oculta dentro de internet y esto puede deberse a razones muy diversas.

Probablemente una de las razones principales por las que a cualquier persona le gustaría aumentar su nivel de anonimato, es que cada vez es más común que los datos relativos a nuestras costumbres de navegación se encuentren de forma casi inmediata a la mano de las empresas que quieren vender algo en internet. Es muy sencillo darse cuenta de esto pues, al hacer clic en algún enlace sobre prácticamente cualquier tema, se muestra publicidad sobre ese tema en cualquier página que se visite. Esto se debe a que empresas muy grandes, como redes sociales, mantienen su gratuidad y fuerza mediante la oferta a diferentes empresas de la posibilidad de que su publicidad sea mostrada únicamente a los usuarios que la podrían querer o necesitar. Es decir, se ha creado un sistema de publicidad mucho más eficaz y no tan general como en los medios tradicionales, a costa de los datos sobre los gustos y costumbres de los usuarios de internet (Wolf, Privacidad y Anonimato en Redes: ¿y la sociedad?, 2018). Esto ha sido un tema común en los últimos tiempos y va de la mano con escándalos muy populares sobre manipulación sobre la forma de pensar de los votantes en asuntos nacionales y filtraciones masivas de datos (Rosenberg & Dance, 2018).

Otra razón muy común por la que a algunas personas les interesan los servicios de anonimato disponibles, es que investigan sobre temas que podrían ser considerados sensibles en su región. Esto podría convertirlos en sujetos de interés para gobiernos

o agencias de investigación. Asimismo, en diversas ocasiones los gobiernos implementan medidas de censura a sitios comunes y corrientes como *YouTube*, por lo que los usuarios suelen buscar medidas para utilizar internet con su funcionalidad básica (Yuan, 2019).

Otro tipo de censura que puede existir en internet ocurre en perjuicio de los periodistas, por lo que en muchas ocasiones no se atreven a ejercer su derecho a la libertad de expresión sin mantener su identidad anónima. Asimismo, en diversas ocasiones los profesionales de la información buscan hacer públicos temas de interés general que implican realizar entrevistas a personas que no quieren ser rastreadas, para mantener su integridad física. Debido a la necesidad que se tiene de obtener conocimiento de muchos tipos de historias es que tanto los periodistas como las personas de quienes obtienen información y su público hacen uso de servicios de anonimato en internet y, de esta forma, ejercer un periodismo más libre y sin fronteras. De igual forma, es común que sufran censura los activistas, disidentes y denunciantes. Tal ha sido el caso de personajes como Edward Snowden que, al hacer público el nivel de intrusión a través de vigilancia en las comunicaciones ejercida por agencias como la NSA, perdió su anonimato y ha sido perseguido desde entonces, viéndose obligado a refugiarse en un país ajeno al asegurar su libertad e integridad física (The Editorial Board, 2014).

Dado los puntos anteriores, podría darse por hecho que cualquier persona, con todo tipo de intereses, puede hacer uso de herramientas como Tor. Lo anterior significa que también personas que realizan actividades ilegales buscan mantenerse lo más ocultos posible. Las herramientas de anonimato tienen mala fama debido a que este tipo de personas la utilizan para fines maliciosos, siendo este punto uno de los temas principales a ser analizados en los capítulos siguientes, con un enfoque en la seguridad de la información mientras se hace uso de la red Tor.

Capítulo 3 Puesta en marcha de un nodo de la red Tor

Este capítulo tiene por objetivo mostrar el proceso a seguir si se quiere levantar un nodo que pertenezca a la red Tor, ya sea de salida o no. Además de seguir las guías desarrolladas y mantenidas por el proyecto Tor, se pretende demostrar qué tan sencillo es llevarlas a cabo y mencionar mi experiencia personal al hacerlo.

3.1 Búsqueda y renta de un servidor público

La robustez y estabilidad de la red Tor depende directamente de la cantidad de personas dispuestas a colaborar donando ancho de banda, es decir, a ejecutar un nodo dentro de la red. Debido a esto, la documentación de este proyecto es bastante explícita en los pasos y requerimientos para lograrlo, así como sus consecuencias (The Tor Project Wiki, 2018).

Puesto que levantar un nodo de salida es, en cuanto a la instalación de infraestructura, uno de los puntos principales de este trabajo, la búsqueda de un proveedor de servidores (VPS*, Virtual Private Server*) debe hacerse considerando las especificaciones de servidores recomendadas por el proyecto Tor, pues cada tipo de nodo tiene sus propios requerimientos para ser ejecutado correctamente. Según la documentación oficial, los requerimientos necesarios para los diferentes tipos de nodo son los siguientes:

Característica	**Nodo intermedio**	**Nodo de salida**
Ancho de banda	10 Mbps de subida (mínimo) 10 Mbps de bajada (mínimo) 16 Mbps de subida (recomendado) 16 Mbps de bajada (recomendado)	10 Mbps de subida (mínimo) 10 Mbps de bajada (mínimo) 16 Mbps de subida (recomendado) 16 Mbps de bajada (recomendado)
Conexiones	7000 conexiones concurrentes	>100000 conexiones concurrentes (si está marcado como un nodo rápido, es decir, tiene más de 100 Mbps disponibles en ancho de banda)
Cantidad de tráfico mensual	~100 GB de salida (mínimo) ~100 GB de entrada (mínimo) >2 TB en total (recomendado)	~100 GB de salida (mínimo) ~100 GB de entrada (mínimo) >2 TB en total (recomendado)

Direccionamiento IPv4	- Directamente en el host (recomendado) o a través de una NAT y utilizando reenvío de puertos. - Dirección estática o que se mantenga por al menos 3 horas	- Directamente en el host (recomendado) o a través de una NAT y utilizando reenvío de puertos. - Dirección estática o que se mantenga por al menos 3 horas
Direccionamiento IPv6	Recomendable pero no necesario	Recomendable pero no necesario
Memoria	512 MB (Ancho de banda <40 Mbps) 1 GB (Ancho de banda >40 Mbps)	1.5 GB
Almacenamiento	Menos de 200 MB	Menos de 200 MB

Tabla 3.1. Requisitos técnicos para montar un nodo en la red.

Hay una gran cantidad de proveedores de servidores privados virtuales, entre los que destacan empresas como *Amazon* (AWS) o *DigitalOcean*. Entre tantas empresas, es sencillo encontrar alguna que ofrezca servidores que cumplan con las características antes mencionadas, sin embargo, hay que considerar factores como el precio para elegir de la mejor manera posible el proveedor. Por ejemplo, algunas empresas cobran a demanda el ancho de banda, es decir, entre más tráfico pase por el servidor (tanto de entrada como de salida) mayor es el precio del servicio. Cabe recalcar que, al tratarse de un servicio que se encarga de dirigir datos de la red Tor, el servidor requerirá procesar mucho tráfico de red (se recomienda un total de 2 TB mensuales para un nodo de salida).

Debido a que no se requiere mucho espacio de almacenamiento para levantar un nodo dentro de la red, prácticamente cualquier servidor puede ser de utilidad, sin embargo, para cumplir los objetivos de este trabajo es necesario almacenar el tráfico de red. Esto implica que otro de los factores técnicos principales para seleccionar un proveedor, además del ancho de banda, es la cantidad de almacenamiento con la que cuentan los servidores.

Además de los factores técnicos, hay que considerar algunos de carácter legal o administrativo. Por ejemplo, puesto que se sabe que un nodo de salida de la red Tor puede contener tráfico considerado ilegal en varios países, es común que los diferentes proveedores incluyan en sus políticas de uso cláusulas en las que explícitamente prohíben levantar un nodo de salida. Asimismo, hay algunos proveedores que, aunque no lo prohíben explícitamente, se sabe que toman medidas inmediatas para tirar el servidor si detectan este tipo de actividad o reciben muchas quejas. Es por esto que la comunidad de Tor mantiene una lista con los diferentes proveedores de servidores virtuales, así como sus comentarios, experiencias y si los

recomiendan o no para ejecutar en sus servidores los diferentes tipos de nodos (The Tor Project Wiki, 2018).

Tomando en cuenta los comentarios de esta lista, así como los requerimientos técnicos necesarios, fue que opté por un proveedor que cumpliera con las siguientes características:

- No prohíbe explícitamente en sus políticas ejecutar un nodo de salida de la red Tor
- Reenvía las quejas recibidas al cliente para que éste se encargue de atenderlas
- Provee ancho de banda ilimitado bajo el mismo precio
- El servidor elegido cuenta con 4 GB de memoria RAM
- El servidor elegido cuenta con 300 GB de almacenamiento (HDD)
- Entrega acceso de administración total al servidor (usuario root)
- El costo mensual de rentar dicho servidor es de 9.90 euros (alrededor de 220 pesos mexicanos, al momento de redactar este documento)

A pesar de que el catálogo de sistemas operativos disponibles para instalar es bastante amplio, se decidió solicitar la instalación del sistema operativo Debian 9 en el servidor. Es importante solicitar al proveedor que la instalación sea en su versión mínima, es decir, sin servicios, programas o aplicaciones web adicionales. Esto se debe a que, si no se solicita esta característica, el servidor se entrega con una gran cantidad de herramientas que no son necesarias para el objetivo de este trabajo. Esto, de una u otra forma, podría posteriormente dificultar las instalaciones y configuraciones a realizar. Algunas características del hardware y software del servidor se muestran en las imágenes 3.1 y 3.2.

```
vcastro@km14102-03:~$ sudo fdisk -l
Disk /dev/vda: 300 GiB, 322122547200 bytes, 629145600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe5081e3f

Device     Boot   Start       End   Sectors   Size Id Type
/dev/vda1          2048   7813119   7811072   3.7G 82 Linux swap / Solaris
/dev/vda2  *    7813120 629145599 621332480 296.3G 83 Linux
```

Imagen 3.1. Espacio disponible para almacenamiento en el servidor.

```
vcastro@km14102-03:~$ cat /etc/*release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
HOME_URL="https://www.debian.org/"
```

Imagen 3.2. Versión del sistema operativo instalado en el servidor.

3.2 Instalación de un nodo Tor

3.2.1 Hardening del servidor

Una vez que se tiene acceso al servidor, es conveniente realizar instalaciones y configuraciones de fortalecimiento o *hardening,* considerado como un procedimiento en el que se reducen las posibles vulnerabilidades existentes en los servicios ofrecidos. Esto debido a que es un servidor que se encuentra expuesto en internet, lo que implica que constantemente será víctima de escaneos, intentos de autenticación, ataques, etcétera. Dado que el servidor únicamente ofrecerá los servicios de SSH y de Tor, estos son los únicos servicios a fortalecer, por lo que podría considerarse como un proceso básico de fortalecimiento.

- Actualización de paquetes, debido a que las últimas versiones pueden tener mejoras o parches de seguridad.
    - Se actualizó la lista de paquetes para tener disponible la última versión en el listado de paquetes candidatos a instalar
    - Se actualizaron los paquetes que pueden ser actualizados.

- Instalar los paquetes necesarios para mantener el servidor con las últimas actualizaciones de seguridad de forma automática.
    - Se instaló un paquete que permite gestionar las actualizaciones sin interacción del usuario.

- Realizar configuraciones de seguridad al servicio SSH en su respectivo archivo de configuración.
    - Se forzó a que se use la versión 2 del protocolo SSH, es decir, la versión más reciente del protocolo.
    - Se forzó a que los registros en las bitácoras únicamente tengan información sobre la autenticación de los usuarios, eliminando datos basura que dificultan el análisis de las bitácoras.
    - Se bloqueó el acceso al usuario root mediante SSH, evitando ataques de fuerza bruta exitosos a ese usuario que se encuentra en todos los sistemas tipo Unix.

    - Se bloqueó el acceso con la contraseña vacía.

- Se instaló fail2ban para evitar ataques de fuerza bruta a SSH. Dicho programa se encarga de procesar las bitácoras de SSH (o algún otro servicio) y, al momento de detectar un ataque de diccionario o fuerza bruta, agrega reglas al firewall del sistema operativo para bloquear la dirección IP origen y mitigar el ataque. Se mantienen las configuraciones por defecto, pues se consideran apropiadas para evitar ataques de fuerza bruta al servidor. Estas son:
    - Revisión de bitácoras de SSH
    - Tiempo de bloqueo de 10 minutos en el firewall
    - Bloquear al encontrar 5 intentos de autenticación fallidos en los últimos 10 minutos.

- Finalmente, se configuró una política de contraseñas para forzar a que todos los usuarios utilicen una contraseña con una longitud de, al menos, 20 caracteres, volviendo de esta manera inviable un ataque de diccionario o fuerza bruta.

3.2.2 Instalación del programa tor

Una vez realizadas estas configuraciones de seguridad, el siguiente paso es instalar tor. Esto puede lograrse mediante dos formas: compilación e instalación a partir del código fuente o instalación mediante paquetes precompilados. En ambos casos, la instalación se logra con pocos pasos. Sin embargo, si se realiza mediante paquetes precompilados utilizando los repositorios del sistema operativo, es más sencillo mantener dicho programa actualizado con los últimos parches de seguridad, por lo que se optó por instalarlo de esta manera. Cabe destacar que se empleará la palabra “tor” (en minúsculas) para el programa o servicio que se encarga de entablar las comunicaciones entre clientes y nodos, mientras que se usará “Tor” (Con la te mayúscula) para hacer referencia a la red y al navegador que utiliza el servicio tor.

En los pasos referentes al *hardening* del servidor se actualizaron las listas de paquetes, por lo que no es necesario volver a hacer esta actualización. En este punto basta con utilizar algún programa para gestionar paquetes como *apt*, *apt-get* o *aptitude* para instalar tor, como se muestra en la imagen 3.3. Cabe resaltar que existen diversas versiones de los paquetes de tor disponibles para instalar. Por ejemplo, las versiones alfa se encuentran disponibles para su instalación pero cuentan con características que no han sido probadas todavía y están pensadas para que los usuarios que decidan instalarlas ayuden a probarlas y reportar *bugs*. Puesto que el objetivo de este trabajo depende de un nodo completamente funcional, se decidió instalar la última versión estable.

```
vcastro@km14102-03:~$ sudo apt install -y tor
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libevent-2.0-5 tor-geoipdb torsocks
Suggested packages:
  mixmaster torbrowser-launcher socat tor-arm apparmor-utils obfsproxy obfs4proxy
The following NEW packages will be installed:
  libevent-2.0-5 tor tor-geoipdb torsocks
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 2909 kB of archives.
```

Imagen 3.3. Instalación de tor mediante paquete.

3.2.3 Actualizaciones de seguridad automáticas

Una vez terminado el proceso de instalación, se configuró *unattended-upgrades*, la herramienta instalada en pasos anteriores. Este es uno de los puntos más importantes para mantener asegurado el nodo, pues significa la instalación de las actualizaciones de seguridad de forma automática. El primer archivo configurado fue */etc/apt/apt.conf.d/50unattended-upgrades,* cuyo contenido se muestra en la imagen 3.4.

```
vcastro@km14102-03:~$ cat /etc/apt/apt.conf.d/50unattended-upgrades
Unattended-Upgrade::Origins-Pattern {
        "origin=Debian,codename=${distro_codename},label=Debian-Security";
        "origin=TorProject";
};
Unattended-Upgrade::Package-Blacklist {
};

Unattended-Upgrade::Automatic-Reboot "true";
```

Imagen 3.4. Archivo de configuración de paquetes a actualizar automáticamente.

La explicación de este archivo de configuración se muestra a continuación:

Línea del archivo de configuración	Significado
Unattended-Upgrade::Origins-Pattern	Sirve para controlar qué paquetes son actualizados. En este caso se actualizarán los paquetes de seguridad de la distribución de Debian usada actualmente (Debian 9 Stretch), así como los paquetes provenientes del proyecto Tor.
Unattended-Upgrade::Package-Blacklist	Sirve para indicar una lista negra de paquetes, por lo que estos no se actualizarán. En este caso no es necesario listar paquetes en este apartado.
Unattended-Upgrade::Automatic-Reboot	Sirve para indicar que el servidor se actualice de forma automática y, con esto, que se apliquen correctamente las actualizaciones.

Posteriormente, es necesario modificar el archivo */etc/apt/apt.conf-d/20auto-upgrades* para habilitar *unattended-upgrades*. El contenido de dicho archivo se muestra en la imagen 3.5.

```
vcastro@km14102-03:~$ cat /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
APT::Periodic::AutocleanInterval "5";
APT::Periodic::Verbose "1";
```

Imagen 3.5. Archivo de configuración para habilitar actualizaciones automáticas.

La explicación de este archivo de configuración se muestra a continuación:

Línea del archivo de configuración	**Significado**
APT::Periodic::Update-Package-Lists “1”;	Habilita la ejecución automática del comando *apt-get update*.
APT::Periodic::Unattended-Upgrade “1”;	Habilita la actualización automática de paquetes.
APT::Periodic::AutocleanInterval “5”;	Habilita la ejecución automática del comando *apt-get autoclean* en un intervalo de 5 días, con lo que se limpia el caché de paquetes, eliminando los que dejan de ser útiles.
APT::Periodic::Verbose “1”;	Enviar un reporte del progreso al usuario root.

Es posible validar la correcta configuración de *unattended-upgrades* al ejecutarlo de forma manual utilizando el comando *unattended-upgrade -d*. Al ejecutarse sin problemas, significa que dicha configuración es correcta, como se muestra en la imagen 3.6.

```
vcastro@km14102-03:~$ sudo unattended-upgrade -d
Initial blacklisted packages:
Initial whitelisted packages:
Starting unattended upgrades script
Allowed origins are: ['origin=Debian,codename=stretch,label=Debian-Security', 'origin=TorProject']
pkgs that look like they should be upgraded:
Fetched 0 B in 0s (0 B/s)
fetch.run() result: 0
blacklist: []
whitelist: []
No packages found that can be upgraded unattended and no pending auto-removals
```

Imagen 3.6. Validación de configuración de actualizaciones automáticas.

3.2.4 Configuraciones y aceptación de un nodo en la red Tor

Finalmente, es necesario configurar tor para que funcione como se espera. Entre las configuraciones disponibles se encuentra el puerto por el que se recibirá el tráfico de

la red, el puerto *SOCKS* o puerto para que las aplicaciones locales puedan redireccionar tráfico a la red Tor, información de contacto, límite de ancho de banda destinado a la red, entre otras. Este archivo de configuración se encuentra por defecto en la ruta */etc/tor/torrc* y no es necesario indicar todas las configuraciones, pues muchas de ellas son opcionales. El archivo de configuración utilizado se muestra en la imagen 3.7.

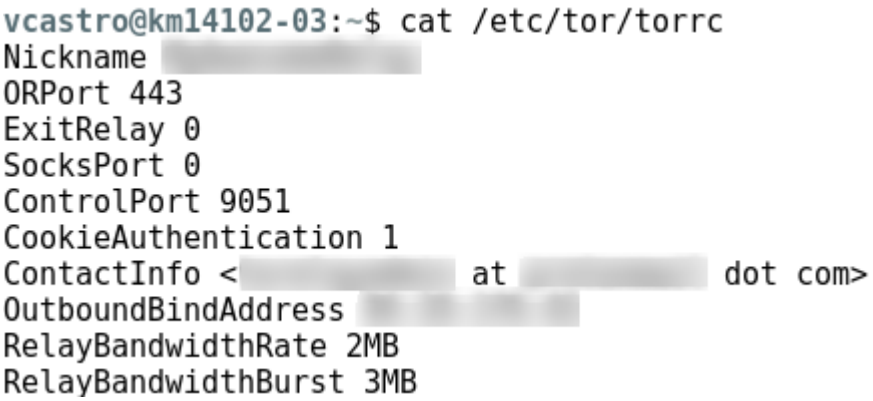

Imagen 3.7. Archivo de configuración de tor.

La explicación de este archivo de configuración se muestra a continuación:

Línea del archivo de configuración	**Significado**
Nickname	Es el identificador fácil de recordar para los humanos del nodo dentro de la red Tor.
ORPort 443	Es el puerto por el que llegará el tráfico proveniente de la red Tor. Se decide utilizar el puerto 443 pues se trata de un puerto que no suele ser bloqueado por los firewalls por tratarse del puerto por defecto del protocolo de aplicación HTTPS.
ExitRelay 0	Se deshabilita la funcionalidad de nodo de salida. Es decir, hasta este punto el nodo actuará únicamente como nodo de entrada o intermedio.
SocksPort 0	Por defecto, tor abre el puerto 9050 para que aplicaciones (locales o no) puedan utilizar el servidor como un proxy de tipo SOCKS y redirigir su tráfico a través de la red Tor. Esta línea deshabilita esa funcionalidad para que funcione exclusivamente como un nodo dentro de la red.
ControlPort 9051	Habilita el uso del protocolo TC (*Tor Control Protocol*) en el puerto 9051, un protocolo que es usado por otros programas para comunicarse con un proceso de tor y, de esta forma, administrarlo. Se decidió su habilitación para poder, posteriormente, administrar el nodo usando el software *nyx*.

CookieAuthentication 1	Sirve para autenticar el proceso de *nyx* ante el proceso de tor, para que sea el único proceso del sistema capaz de controlarlo. Sirve mediante un archivo creado por tor en su directorio de datos, cuyo contenido debe ser conocido por el proceso administrador.
ContactInfo	En esta línea se indica el correo de contacto del administrador del nodo, que puede ser usado para informar por algún error de configuración o cualquier otro tipo de problema. Puesto que se trata de información que será pública e indexada por buscadores como Google, se decidió crear un correo exclusivamente para ser publicado como la información de contacto de este nodo. Esto se debe a que, al ser un correo publicado en internet, puede ser encontrado por *spammers*, por lo que también se publicó en un formato que no facilita su recolección mediante expresiones regulares.
OutboundBindAddress	Esta configuración sirve para indicar la dirección IP por la que Tor estará sirviendo. Dado que el servidor cuenta con dos direcciones IP públicas, esta configuración sirve para controlar cuál de estas es la que se desea asociar a la red Tor. Es de utilidad, ya que más adelante esta dirección IP se asocia con un registro DNS.
RelayBandwidthRate	Sirve para indicar tanto la cantidad promedio de tráfico de entrada como de salida. Se decidió utilizar un valor de 2MB (4MB en total) para poder ser considerado un nodo rápido y, posteriormente, tener más oportunidad de ser aceptado como un nodo de salida.
RelayBandwidthBurst	Esta configuración sirve para indicar la cantidad máxima de tráfico de entrada y salida del nodo.

Una vez lista la configuración, es necesario reiniciar tor para que dichos cambios surtan efecto. En el archivo de bitácoras de tor puede observarse que se realizan pruebas para validar que el servidor y el puerto son visibles desde el exterior y, una vez terminadas dichas pruebas, publica el descriptor del servidor en el directorio de la red para que este sea aceptado como un nodo. El reinicio de tor, así como los últimos registros en la bitácora después de este, se muestran en la imagen 3.8.

```
vcastro@km14102-03:~$ sudo systemctl restart tor@default
vcastro@km14102-03:~$ sudo tail -n 2 /var/log/tor/log
Oct 10 01:52:54.000 [notice] Self-testing indicates your ORPort is reachable
from the outside. Excellent. Publishing server descriptor.
Oct 10 01:52:55.000 [notice] Performing bandwidth self-test...done.
```

Imagen 3.8. Pruebas automáticas por parte de tor para validar configuración.

Posteriormente, es necesario esperar hasta que el nodo sea aceptado y mostrado en el directorio de la red. Una vez que suceda esto, es posible visualizar las características del nodo utilizando el servicio de búsqueda de Tor. En este servicio es posible buscar los diferentes nodos que se encuentran en determinado país, que tienen determinadas características, o a través de su dirección IP o *Nickname.* En la imagen 3.9 se muestra el estado del nodo pocas horas después de ser aceptado en la red utilizando el sistema de búsqueda de Tor.

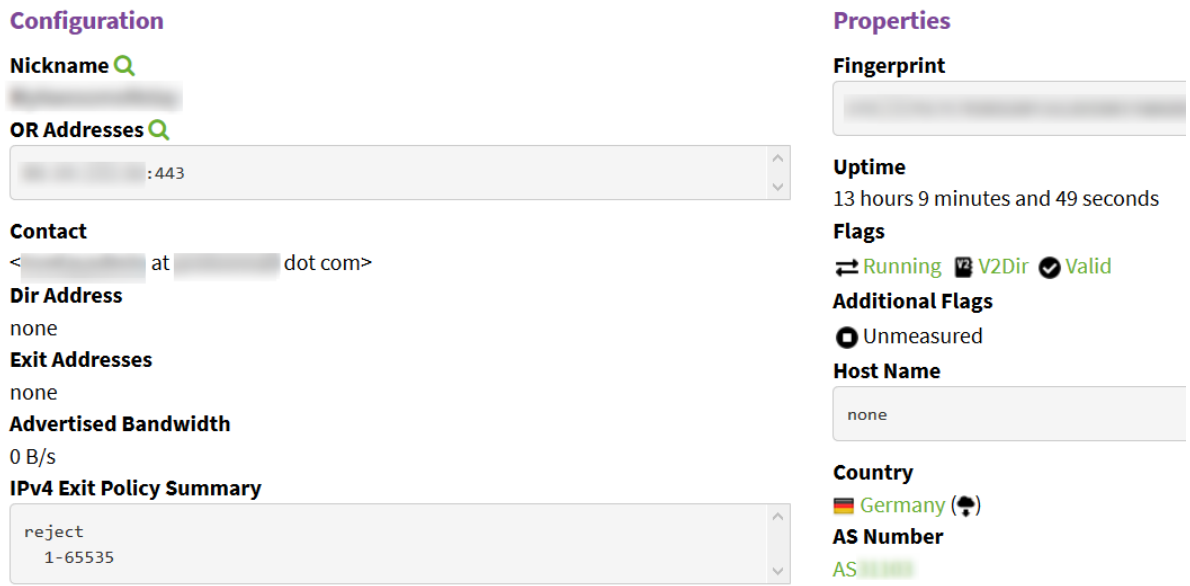

Imagen 3.9. Nodo mostrado en el sistema de búsquedas de la red Tor.

Hasta este punto, el servidor ya forma parte de la red Tor. Sin embargo, en la imagen 3.9 se puede observar que, a pesar de haber transcurrido varias horas desde que se aceptó en la red, aún no se ha registrado tráfico proveniente de Tor. Esto se debe a que existe un *ciclo de vida* para los nodos nuevos. En la documentación oficial, se explica que durante los primeros 3 días en que fue aceptado, no se verá actividad proveniente de tor, sino que únicamente se agrega al directorio y se hacen pruebas locales. La actividad, y por lo tanto el aumento de tráfico, se da entre los días 3 y 8. Esto sirve para que algunos clientes comiencen a utilizar el nodo como parte de sus circuitos, mientras que se comienzan a hacer mediciones externas para validar el ancho de banda disponible en el servidor. Conforme se realizan estas pruebas, el peso de consenso del servidor aumenta y, en consecuencia, sigue aumentando la cantidad de conexiones. Durante este periodo, el nodo únicamente se utiliza como el punto intermedio de los circuitos, puesto que no cuenta con la bandera de entrada ni de salida. Es hasta el día 8 que, en caso de haber sido validado como un nodo confiable, rápido y estable, se le puede asignar la bandera de entrada y comenzar a formar parte de circuitos como el punto de acceso a la red.

3.2.5 Monitoreo de actividad del nodo

Aunque, teóricamente, ya no es necesario hacer nada más para que el nodo funcione con normalidad, se tomó la decisión de instalar un programa especializado en monitorearlo. Aunque existen diversos programas para monitoreo general, se optó por instalar *nyx*, un programa recomendado por el proyecto Tor para monitoreo de los nodos administrados. Entre las ventajas que tiene el utilizar este software se encuentran la generación de gráficas en tiempo real de tráfico enrutado por el servidor tanto de subida como de bajada, una lista de las conexiones entrantes y salientes del servidor relacionadas con los diferentes nodos de Tor mostrando su información pública (número de sistema autónomo, contacto, etcétera), consumo de recursos en tiempo real, visualización de la bitácora, edición de la configuración del nodo, etcétera (Tor Nyx, 2018). En la imagen 3.10 se muestra la pantalla inicial del servidor, donde se puede observar la cantidad de tráfico que enruta el servidor en tiempo real.

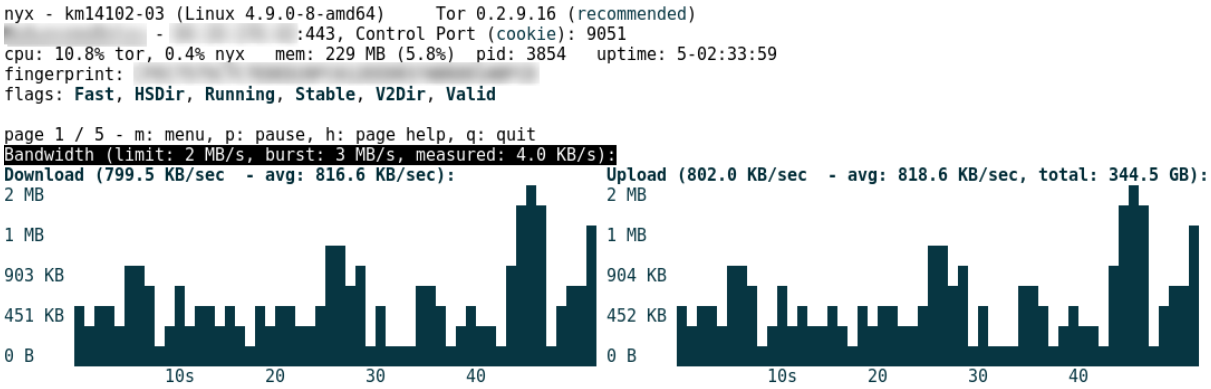

Imagen 3.10. Pantalla de consola de monitoreo del nodo.

Para analizar el comportamiento del nodo en cuestiones de su clasificación, se mantuvo con esta configuración durante un tiempo superior a los diez días. En la imagen 3.11, tomada de las métricas del proyecto Tor, se puede observar las variaciones de probabilidad para funcionar como nodo de entrada, intermedio o de salida.

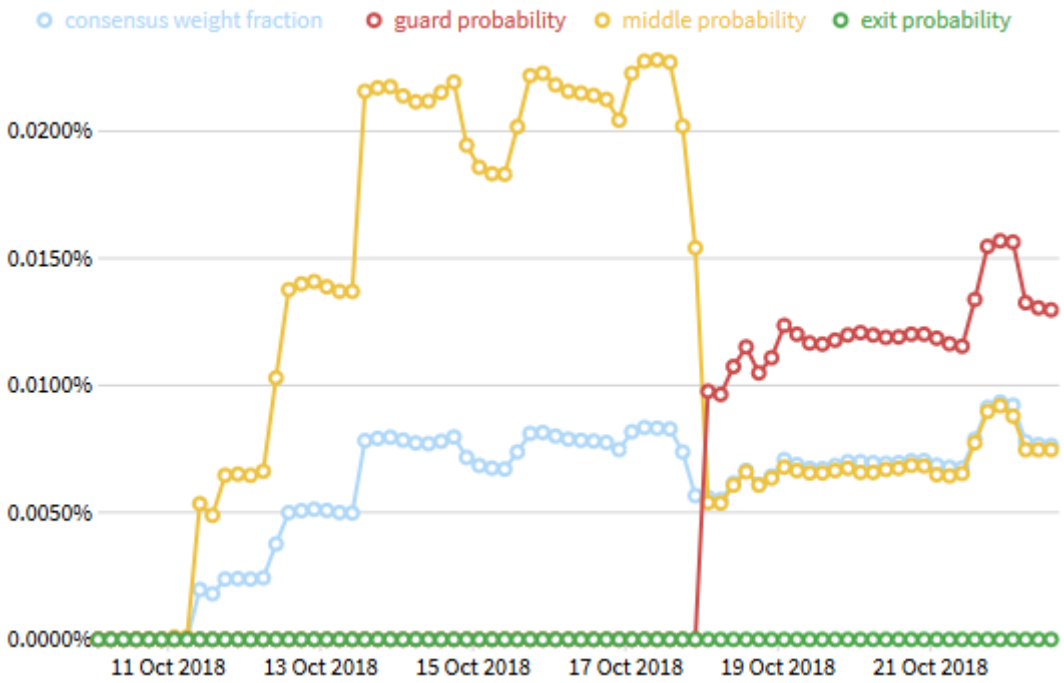

Imagen 3.11. Probabilidad de ser nodo de entrada, intermedio o de salida.

3.3 Configuraciones para tener un nodo de salida

Uno de los primeros pasos recomendados para administrar un nodo de salida es brindar información a las personas sobre el hecho de que esa dirección IP se trata de un nodo de salida. Esto se debe a que, al Tor ser usado por todo tipo de personas, existe la posibilidad que este nodo sea usado para escanear o atacar determinado servidor en internet, que mande spam, que sea usado para descargar material con derechos de autor, etcétera. Los administradores de los sistemas afectados buscarán información relacionada con la dirección IP registrada en sus bitácoras por lo que, para evitar la mayor cantidad de correos de queja posible, es necesario hacerles saber la naturaleza de este servidor (The Tor Project, 2018).

El primer paso para lograr lo anterior es crear un registro DNS de tipo PTR (Pointer) para el servidor en el que sea visible la naturaleza de este. De este modo, al ser consultado este registro por cualquier persona en internet, se puede ver que se trata de un nodo de salida y, asimismo, este registro es de utilidad para mostrar la URL donde se explica de forma más detallada el funcionamiento de la red. Para lograr obtener el registro PTR, fue necesario únicamente utilizar la consola de administración del proveedor de servidores virtuales. Una vez registrado el nombre, es posible hacer una búsqueda inversa de DNS de la dirección IP del servidor. En la imagen 3.12 se muestra dicha búsqueda utilizando el programa *dig* y consultando por el nodo a un servidor DNS público de Google.

```
user@TraffAnalysis:~$ dig +noall +answer \
> -x [redacted] \
> @8.8.8.8 | cut -d ' ' -f 3
IN      PTR     t0r.exit.relay.http [redacted] index.html.
```

Imagen 3.12. Petición DNS para hacer resolución inversa de la dirección IP del nodo.

Posteriormente, se recomienda poner un aviso a través de una página web donde se explique que se trata de un nodo de salida de la red Tor. El enlace a este aviso es el que se muestra en el registro PTR. Para esto, se decidió utilizar el servidor web Apache y una de las plantillas recomendadas por el proyecto Tor para explicar la situación a todas las personas que se hayan visto afectadas por el nodo, basta con modificar la plantilla para que contenga mis datos de contacto. En la imagen 3.13 se puede observar un extracto de la página web mostrada en el servidor web.

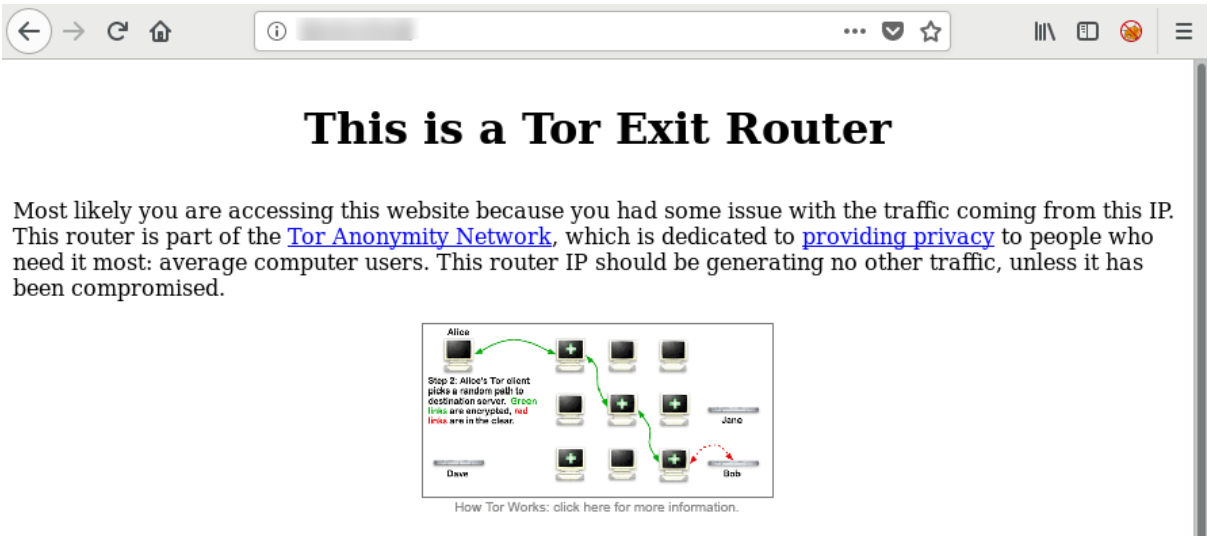

Imagen 3.13. Página web mostrada al visitar el servicio HTTP del nodo.

Una vez que se encuentran listos los respectivos anuncios para reducir la cantidad de mensajes de quejas, se configuró el nodo para servir como salida de la red Tor. Para esto fue necesario modificar nuevamente el archivo */etc/tor/torrc*, de tal manera que la directiva *ExitRelay* quedó con el valor 1. Asimismo, se agregaron múltiples líneas que sirven para indicar la política de salida del nodo. Se decidió bloquear la mayoría de puertos TCP para evitar, en la medida de lo posible, que este nodo sirviera para la descarga de archivos mediante protocolos P2P. Esto debido no solo a que la descarga de archivos de gran tamaño puede dificultar los análisis realizados y la experiencia del resto de los usuarios (Arma, Tor Blog, 2009), sino que además la red Tor no recomienda a sus usuarios el uso de estos protocolos a través de la red, debido a que son altamente propenso a publicar la dirección IP real del cliente. (Arma, Tor blog, 2010)

En total, se permitió que el nodo fuera utilizado con 102 puertos TCP, bloqueando el resto de los puertos. Entre los puertos permitidos para la salida se encuentran los puertos por defecto para los protocolos de capa de aplicación más comunes, entre los que destacan: 20 y 21 (FTP), 23 (Telnet), 25 (SMTP), 53 (DNS), 80 (HTTP), 110 (POP3), 143 (IMAPS), 389 (LDAP) y 443 (HTTPS).

Después de realizar los cambios necesarios en el archivo de configuración de tor, es necesario reiniciar el servicio y esperar a que sea aceptado por la red como un nodo de salida. Cuando esto sucede, dicho cambio se refleja en el directorio de la red, por lo que es posible observar la bandera de salida en el nodo, así como la política de salida configurada. Cabe resaltar que, en el momento en que se acepta al nodo como salida, se quita la bandera de *guard* o entrada. Estas configuraciones se muestran en la imagen 3.14.

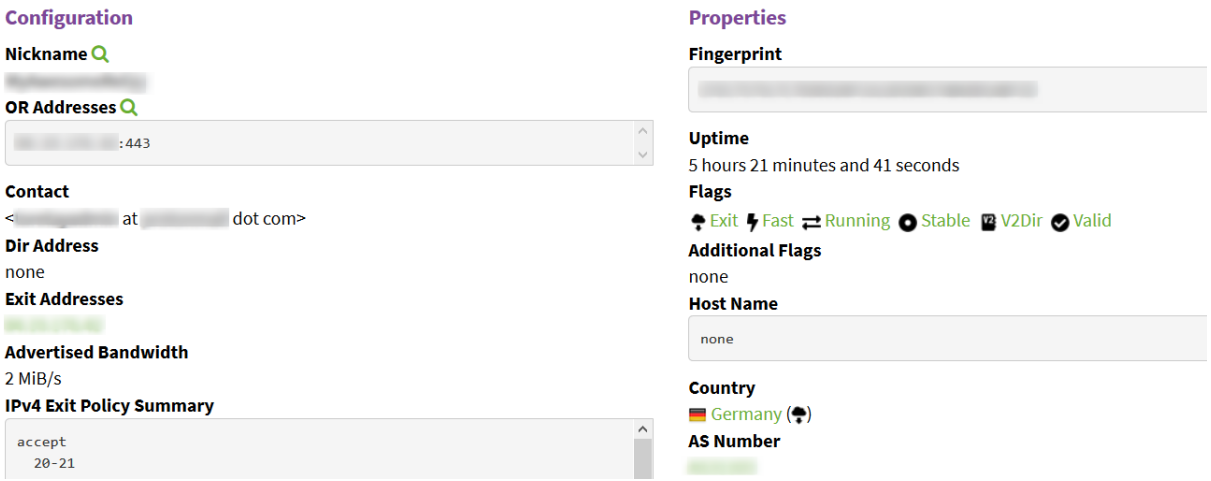

Imagen 3.14. Características del nodo mostradas por el sistema de búsquedas de la red.

3.4 Dificultades encontradas al poner en marcha la infraestructura

El problema principal durante la implementación de la infraestructura fue la búsqueda de un proveedor de servidores privados virtuales que fuera lo suficientemente abierto como para permitir la instalación de nodos de salida, sin embargo, fue de gran utilidad la información sobre proveedores recabada por la comunidad Tor.

Una vez teniendo acceso a un servidor virtual que, además de lo anterior, sea asequible económicamente, es necesario hacer diversas configuraciones de seguridad e instalación en dicho servidor. Aunque estas configuraciones pueden parecer triviales para algunas personas, es un hecho que se necesita determinada experiencia en el uso del sistema operativo donde se instala tor para realizarlas mientras se comprende lo que se está haciendo. Al comprender todo lo que se realiza es mucho más sencillo solucionar cualquier tipo de problema que se presente.

En cuanto a las medidas de prevención de mensajes de quejas, la mayor dificultad es la obtención de un registro DNS de tipo PTR. Sin embargo, si el proveedor facilita la modificación de este registro, se vuelve un asunto trivial. Por otro lado, al mostrar la información explicativa de la red Tor a través de HTTP, se publicó en internet el correo de contacto para cualquier aclaración con respecto a la dirección IP. Esto provocó que dicho correo fuera encontrado por los diferentes bots que constantemente buscan, entre otras cosas, correos electrónicos expuestos en internet. Por lo anterior, fue relativamente común la recepción de spam en esa dirección.

Capítulo 4 Análisis de la actividad en la red Tor

En este capítulo se presentan diversos análisis basados en la actividad de los usuarios para determinar, principalmente, en qué medida estos utilizan la red con fines maliciosos. Se pretende resolver esta duda con análisis basados en el número de conexiones de entrada y salida para obtener un contexto geográfico y cultural de su uso, los nombres de dominio consultados, entre otros.

Adicionalmente, se muestra el proceso que se llevó a cabo para capturar el tráfico y las consideraciones al hacerlo.

4.1 Visualización geográfica basada en las conexiones de un nodo de entrada

Previo a realizar la configuración para tener un nodo de salida, y mientras se tenía la bandera de entrada, se vieron las conexiones establecidas con el servidor en diferentes instantes. Puesto que el proyecto Tor cuenta con un sistema de búsquedas que muestra los datos de los diferentes nodos a partir de su dirección IP, es posible determinar cuándo una dirección IP de estas conexiones es un nodo de Tor o se trata de un cliente accediendo a la red.

Se desarrolló un programa que se encarga de automatizar este análisis y, de esta manera, conocer los países que más uso le dan a la red a partir de la cantidad de conexiones de entrada en diferentes instantes. Para lograr esto, este programa hace peticiones HTTP a la aplicación de Tor que muestra esta información, interpreta el código JavaScript de la respuesta para poder obtener cadenas específicas de cuando una dirección IP introducida es un nodo y cuando no lo es. Una vez que conoce su condición, determina a qué país pertenece a través de las bases de datos *GeoLite* de direcciones IP y lo almacena en una base de datos *SQLite*. Finalmente, utilizando las bibliotecas de visualización de datos *pandas* y *plotly*, genera un mapa del mundo mostrando la cantidad de conexiones por país representada con diferentes tonalidades. En la imagen 4.1 se muestra el resultado de dicho análisis.

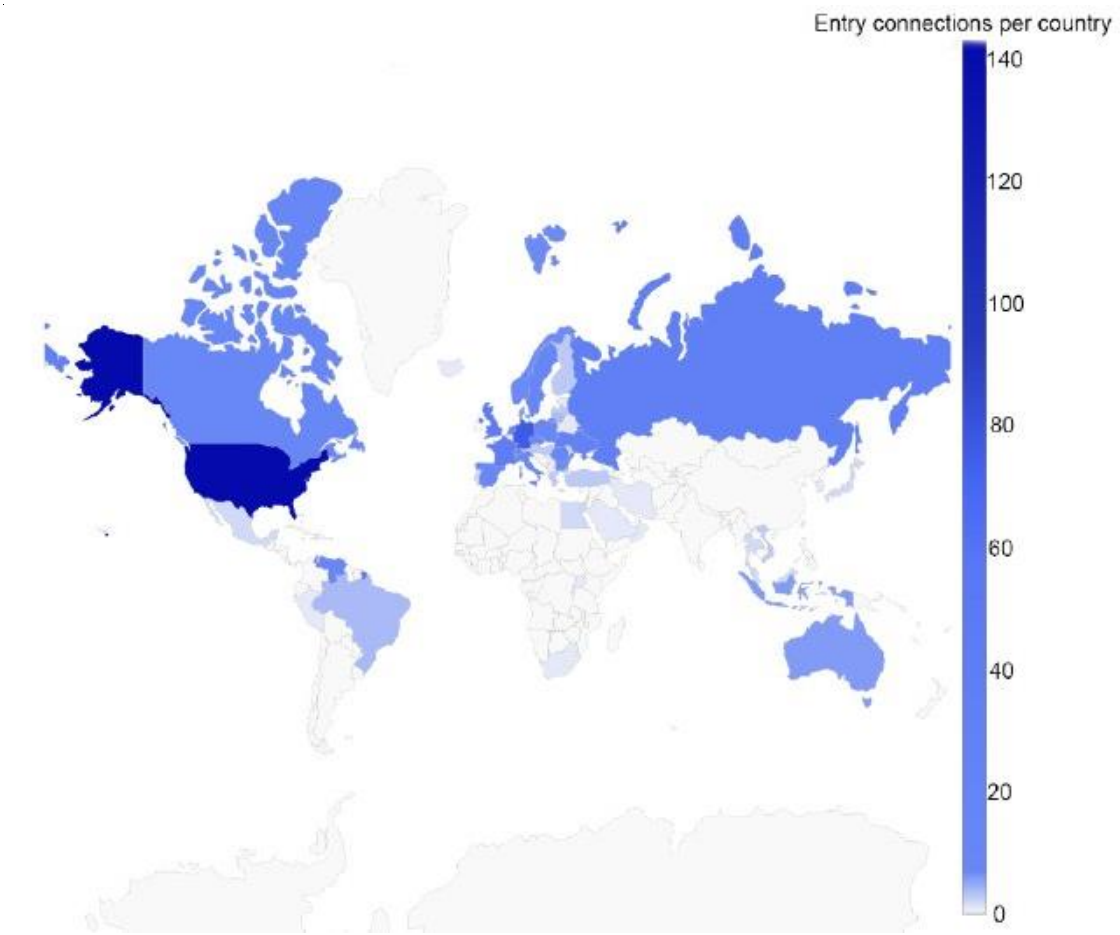

Imagen 4.1. Cantidad de conexiones por país hacia un nodo de entrada.

En este análisis puede observarse que Estados Unidos tiene un índice muy alto de conexiones hacia la red tor en comparación con el resto del mundo. De igual forma, se observa que los países europeos, Canadá y Rusia tienen una gran cantidad de usuarios activos de la red. Por otro lado, se observa que prácticamente todo el continente africano, y gran parte de Asia y el resto de América tienen poca actividad.

Esto indica que los habitantes de los países más desarrollados y con mayor penetración de internet son los que tienen un mayor interés por mantenerse anónimos usando la red Tor. Sin embargo, los resultados de este análisis no son determinantes para decidir si los usuarios de la red la utilizan principalmente con fines de evitar la censura y espionaje o no, ya siguen la tendencia del uso general de la red superficial mostrada en otros estudios como el desarrollado por “Our World in Data” (https://ourworldindata.org/internet). En la imagen 4.2 se muestra un mapa con los datos del uso general de Internet.

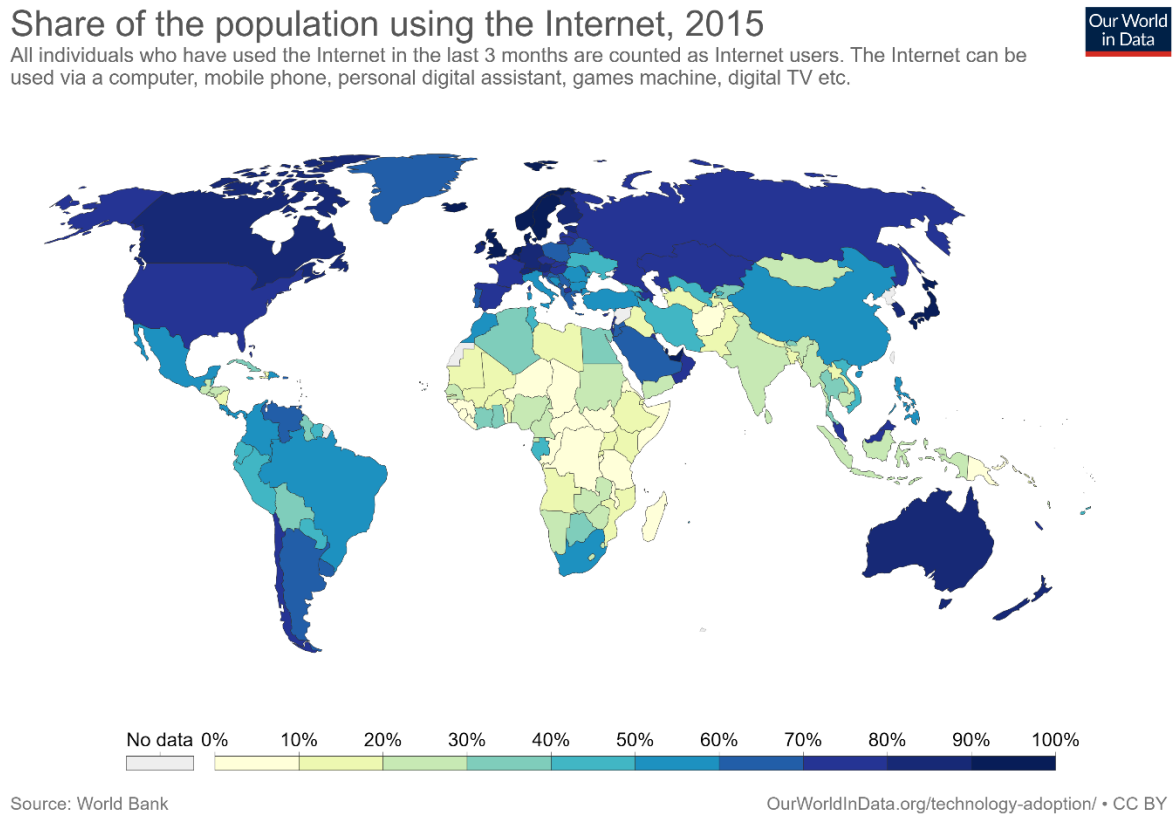

Imagen 4.2. Uso general de Internet en todo el mundo.

4.2 Captura de tráfico de red

En el momento en que se es parte de la red Tor como un nodo de salida es posible capturar el tráfico y analizarlo de diversas maneras. Esto se debe a que en este punto es necesario descifrar los datos que se enviaron a través del circuito para que el destino original sea capaz de interpretarlos.

Para capturar dichos datos se utilizó la herramienta *tcpdump*. Puesto que este programa se apoya de la biblioteca *libpcap* para la captura, escritura y filtrado de paquetes, es de gran utilidad para que, posteriormente, otros programas de análisis puedan utilizar los archivos generados como si de paquetes enviados en tiempo real se tratara.

La instalación de dicho programa se realizó a través de los paquetes de Debian, utilizando el comando '*sudo apt install -y tcpdump"*. Cabe destacar que es necesario contar con permisos de *root* para poder utilizar este programa. Lo anterior debido a que, entre otras cosas, requiere interactuar directamente con la interfaz de red para leer las tramas enviadas por ella y realizar las capturas de tráfico. En la imagen 4.3 puede observarse la versión instalada de tcpdump.

```
vcastro@km14102-03:~# tcpdump --version
tcpdump version 4.9.2
libpcap version 1.8.1
```

Imagen 4.3. Ejecución de tcpdump para mostrar su versión.

Posteriormente, se escribió un script para capturar el tráfico para poder ser ejecutado por un intérprete de comandos como *bash*. Se decidió hacer capturas de tráfico de una hora para realizar el análisis como fue planificado, separando en múltiples capturas para facilitar su procesamiento, considerando que al estar enrutando 4 MB de tráfico por segundo en promedio, en una hora se pueden capturar aproximadamente 14 GB de tráfico. Asimismo, se decidió validar que el espacio en disco fuera mayor a los 20 GB e inicie la captura de tráfico únicamente si se cumple esta condición para evitar el llenado total del disco duro, pudiendo perder así el servicio. En la imagen 4.4 se muestra dicho script.

```
root@km14102-03:~# cat /opt/capture.sh
x=`df -ah | grep /dev/vda2 | cut -d 'G' -f 3`
min_space=20

if (( $(echo "$x > $min_space" |bc -l) )); then
        /usr/sbin/tcpdump \
                -i ens18  \
                -w /opt/traffic/dump_%Y-%m-%d_%H:%M:%S.pcap \
                -G 3600 -W 1 'not port 22'
fi
```

Imagen 4.4. Script usado para capturar el tráfico de red.

La explicación de las secciones de este programa se muestra a continuación:

Secciones de comando	Explicación
/usr/sbin/tcpdump	Sirve para ejecutar el programa tcpdump a través de su ruta absoluta.
-i ens18	Indica que la interfaz de la cual capturará paquetes es la llamada ‘ens18’.
-w /opt/traffic/dump_%Y-%m-%d_%H:%M:%S.pcap	Indica que el tráfico se guardará en la ruta ‘/opt/traffic’ utilizando la fecha y hora como parte del nombre para identificar las diferentes capturas realizadas. Utiliza la extensión ‘pcap’ pues, aunque en sistemas de tipo Unix no importa la extensión, se refiere al tipo de archivo de captura de paquetes.
-G 3600	Los segundos de rotación entre capturas. Indica que cada captura se llevará a cabo durante una hora.
-W 1	Se refiere al límite de capturas a realizar. Al

	indicar un uno, significa que se escribirá una única captura con una duración de una hora.
‘not port 22’	Se trata de un filtro de *tcpdump*. Le indica que capture todo el tráfico, excepto aquellos paquetes que tengan como puerto origen o destino el 22. Esto sirve para evitar contaminar las capturas con la sesión de administración establecida a través de SSH.

Para ejecutar el script anterior se utilizó el programa *crontab*. Dicho programa sirve para instalar o desinstalar tareas programadas a través de *cron*. Se estableció una regla que indica la ejecución del script en el primer minuto de cada hora, de cada día, de cada mes. Con esta combinación de tarea programada y las opciones de ejecución de *tcpdump* se logra una efectiva captura de tráfico en archivos separados durante cada hora. En la imagen 4.5 puede observarse la regla de *cron* que se encarga de ejecutar el script anterior.

```
root@km14102-03:~# crontab -l

# m h  dom mon dow   command
0 * * * * /bin/bash /opt/capture.sh
```

Imagen 4.5. Regla de crontab para ejecutar cada hora el script de captura de tráfico.

Finalmente, después de 19 horas de captura de tráfico, se consiguieron aproximadamente 254 GB de tráfico de red separados en 19 archivos *pcap*. Si bien, puede ser muy poco en comparación con el total de tráfico enrutado por los nodos de salida, se consideró como una muestra significativa para realizar el estudio debido a la aleatoriedad necesaria para eso es proporcionada por la forma en la que se crean los circuitos de la red. Es decir, el tráfico capturado en este nodo tuvo una probabilidad muy similar de pasar por cualquier otro nodo de salida de la red. En la imagen 4.6 puede observarse un listado de dichos archivos y su respectivo tamaño en GB.

```
vcastro@km14102-03:~$ du -h /opt/traffic/*
11G     /opt/traffic/dump_2018-10-25_22:00:01.pcap
14G     /opt/traffic/dump_2018-10-26_00:00:01.pcap
13G     /opt/traffic/dump_2018-10-26_01:00:01.pcap
11G     /opt/traffic/dump_2018-10-26_02:00:01.pcap
13G     /opt/traffic/dump_2018-10-26_03:00:01.pcap
11G     /opt/traffic/dump_2018-10-26_04:00:01.pcap
11G     /opt/traffic/dump_2018-10-26_05:00:01.pcap
16G     /opt/traffic/dump_2018-10-26_06:00:01.pcap
14G     /opt/traffic/dump_2018-10-26_07:00:01.pcap
11G     /opt/traffic/dump_2018-10-26_08:00:01.pcap
13G     /opt/traffic/dump_2018-10-26_09:00:01.pcap
15G     /opt/traffic/dump_2018-10-26_10:00:01.pcap
15G     /opt/traffic/dump_2018-10-26_11:00:01.pcap
16G     /opt/traffic/dump_2018-10-26_12:00:01.pcap
16G     /opt/traffic/dump_2018-10-26_13:00:01.pcap
17G     /opt/traffic/dump_2018-10-26_14:00:01.pcap
16G     /opt/traffic/dump_2018-10-26_15:00:01.pcap
14G     /opt/traffic/dump_2018-10-26_16:00:01.pcap
16G     /opt/traffic/dump_2018-10-26_17:00:01.pcap
```

Imagen 4.6. Listado de archivos pcap generados.

Posteriormente se utilizó la herramienta *tcpflow* para analizar, entre otras cosas, la cantidad de tráfico correspondiente a cada puerto de los permitidos en la política de salida. Para realizar este análisis se utilizó el primer archivo de los 19 generados, para el cual, los resultados indican una gran cantidad de tráfico correspondiente al puerto 443 en comparación con los demás puertos permitidos. Aunque podría parecer que todo este tráfico corresponde al protocolo HTTPS por tratarse de su puerto por defecto, la realidad es que parte de este tráfico corresponde al tráfico de enrutamiento de Tor, pues se configuró el puerto 443 como *ORPort* en el archivo de configuración de tor. En la imagen 4.7, generada por *tcpflow*, puede observarse que más del 80 por ciento de tráfico corresponde a este puerto.

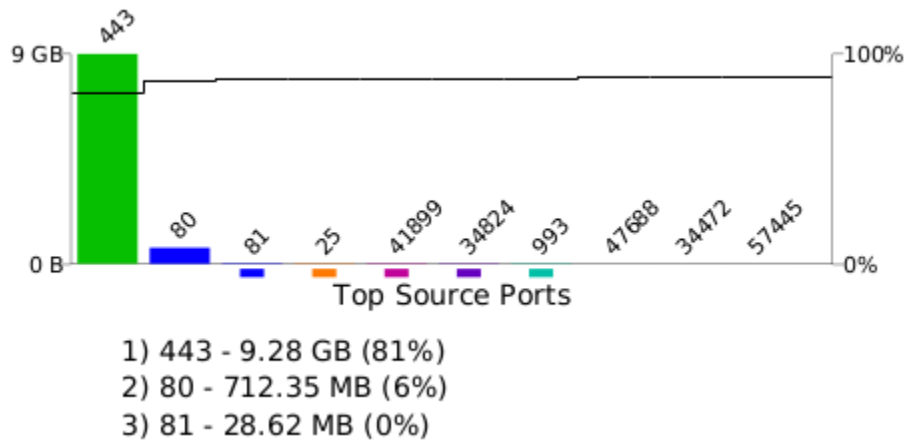

Imagen 4.7. Relación de tráfico del puerto 443 en comparación con el resto de puertos.

Para obtener una mayor facilidad en el manejo de los archivos de tráfico de red, se decidió aplicar un nuevo filtro a dichos archivos para lo cual nuevamente se hizo uso de *tcpdump*. El objetivo de este nuevo filtro es el de eliminar todo el tráfico correspondiente al puerto 443 que está asociado a la directiva ORPort en el archivo de configuración de tor, el cual se trata de tráfico cifrado de la red Tor destinado al nodo. En este filtro elimina todo el tráfico que tiene como puerto origen el 443 y cuya dirección IP origen es la dirección de mi servidor, así como todo el tráfico que tiene como puerto destino el 443 y cuya dirección IP destino es la dirección de mi servidor. El resultado de este nuevo filtro son 19 archivos *pcap*, con un total de aproximadamente 119 GB de datos, lo que significa una reducción de más del 50 por ciento de los paquetes originalmente capturados. En la imagen 4.8 puede observarse el conjunto de las nuevas capturas, así como su respectivo tamaño en GB. En este punto, se tiene listo todo el tráfico para hacer el análisis posterior.

```
vcastro@km14102-03:~$ du -h /opt/traffic/*
5.1G    /opt/traffic/clean_dump_2018-10-25_23:00:01.pcap
6.5G    /opt/traffic/clean_dump_2018-10-26_00:00:01.pcap
5.8G    /opt/traffic/clean_dump_2018-10-26_01:00:01.pcap
5.0G    /opt/traffic/clean_dump_2018-10-26_02:00:01.pcap
5.9G    /opt/traffic/clean_dump_2018-10-26_03:00:01.pcap
4.9G    /opt/traffic/clean_dump_2018-10-26_04:00:01.pcap
4.9G    /opt/traffic/clean_dump_2018-10-26_05:00:01.pcap
7.2G    /opt/traffic/clean_dump_2018-10-26_06:00:01.pcap
6.4G    /opt/traffic/clean_dump_2018-10-26_07:00:01.pcap
4.7G    /opt/traffic/clean_dump_2018-10-26_08:00:01.pcap
6.0G    /opt/traffic/clean_dump_2018-10-26_09:00:01.pcap
6.8G    /opt/traffic/clean_dump_2018-10-26_10:00:01.pcap
6.6G    /opt/traffic/clean_dump_2018-10-26_11:00:01.pcap
7.1G    /opt/traffic/clean_dump_2018-10-26_12:00:01.pcap
7.4G    /opt/traffic/clean_dump_2018-10-26_13:00:01.pcap
7.5G    /opt/traffic/clean_dump_2018-10-26_14:00:01.pcap
7.2G    /opt/traffic/clean_dump_2018-10-26_15:00:01.pcap
6.1G    /opt/traffic/clean_dump_2018-10-26_16:00:01.pcap
7.1G    /opt/traffic/clean_dump_2018-10-26_17:00:01.pcap
```

Imagen 4.8. Listado de capturas de tráfico al aplicarles el nuevo filtro.

4.3 Nombres de dominio consultados

Este tipo de análisis tiene por objetivo obtener una idea general de la principal actividad de los usuarios de la red Tor a través de los nombres de dominio resueltos a través del nodo instalado y, tener una idea general de si estos dominios son maliciosos o si se trata de dominios normalmente consultados por cualquier persona en la red superficial.

Con el tráfico antes capturado es posible obtener una lista de los nombres de dominio consultados por el servidor durante el tiempo de captura. Esto es posible debido a

que el servicio DNS utiliza, en la mayoría de los casos, UDP como protocolo de la capa de transporte, mientras que la red Tor no es capaz de enrutar tráfico UDP, únicamente TCP. Por esta razón, cuando un cliente solicita la resolución del nombre de dominio, este nombre es enviado al nodo de salida para que sea éste quien se encargue de hacer las peticiones DNS correspondientes. Puesto que es el nodo de salida el que se encarga de solicitar dichas resoluciones, basta con filtrar todo el tráfico correspondiente al puerto 53 de UDP y, posteriormente, buscar dichas peticiones para conocer los sitios visitados por los usuarios de la red a través del nodo de salida.

4.3.1 Extracción de consultas DNS con tshark

Para lograr lo anterior, nuevamente se hizo uso de la herramienta *tcpdump*, escribiendo un nuevo archivo por cada captura en donde únicamente se encuentra el tráfico del puerto 53. Una vez generados todos los archivos, en cada uno de ellos se buscaron las solicitudes de resolución DNS utilizando la herramienta *tshark*. *Tshark* es un analizador de protocolos muy similar a *Wireshark* con la peculiaridad de tratarse de una herramienta de línea de comandos, por lo cual es ideal para trabajar con capturas de tráfico grandes.

En primer lugar, se decidió buscar exclusivamente las peticiones correspondientes a registros DNS de tipo A, es decir, la resolución de un nombre de dominio a una dirección IPv4. Para lograr lo anterior, se utilizó el filtro mostrado en la imagen 4.9 en cada uno de los archivos previamente generados.

```
user@TraffAnalysis:/opt/dns$ tail -n +15 dns_queries.sh | head -n -3
for dump in dns_traffic/*; do
                tshark -r "$dump" -T fields -e dns.qry.name \
                        -e dns.a \
                | grep -F '.' \
                | awk '{print tolower($0)}' \
                | grep -e '[0-9a-z][[:space:]]\{1,\}[0-9]' \
                | sort >> /opt/dns/dns_queries_A.txt
done
```

Imagen 4.9. Filtro para obtener los nombres de dominio consultados por el nodo.

La explicación de las secciones de este comando se muestra a continuación:

Secciones de comando	Explicación
tshark	Programa que se encarga de analizar tráfico de red.
-r “$dump”	Leerá el tráfico a partir del archivo indicado en la variable $dump.
-T fields	Formato para mostrar la salida del comando. Con la opción ‘fields’ se imprimirá el campo indicado con la opción -e.

-e dns.qry.name	El primer campo a mostrar será el nombre de dominio consultado.
-e dns.a	El segundo campo a mostrar es la dirección IP correspondiente al nombre consultado, es decir, mostrará los registros de tipo “A”.

Una vez obtenidos todos los nombres de dominio consultados, se observa que durante las 19 horas de captura se realizaron 285044 peticiones de resolución de nombres en total. Sin embargo, únicamente se trata de 186356 nombres consultados. Dicho conteo se muestra en la imagen 4.10.

```
user@TraffAnalysis:/opt/dns$ wc -l dns_queries_A.txt
285044 dns_queries_A.txt
user@TraffAnalysis:/opt/dns$ awk '{print $1}' dns_queries_A.txt | \
> sort -u > unique_domains.txt
user@TraffAnalysis:/opt/dns$ wc -l unique_domains.txt
186356 unique_domains.txt
```

Imagen 4.10. Conteo de dominios consultados totales y únicos.

4.3.2 Visualización de dominios de nivel superior

Posteriormente, se desarrolló un script en Python para poder procesar y visualizar más fácilmente los datos aquí extraídos. Entre sus funciones se encuentra la de obtener una cuenta de los *top-level-domains* y *second-level-domains* más visitados dentro de la red y, con esto, obtener la actividad de sus usuarios y el uso que le dan.

En la imagen 4.11 se puede apreciar una gráfica con el top 10 de dominios de nivel superior consultados por el nodo de salida. Como era de suponerse, dominios como “com” o “net” ocupan un grandísimo porcentaje del total. Cabe destacar que se observa en ese top los nombres de dominio “ru” (Rusia) y “cn” (China), ambos países asiáticos, por lo que fácilmente se puede intuir que los ciudadanos de esos países forman parte de los principales usuarios de la red, algo comprensible pues en ellos existe una gran restricción de acceso a Internet o un alto índice de espionaje y vigilancia a sus habitantes.

Por otro lado, aparecen los nombres de dominio “de” (Alemania), “fr” (Francia), “uk” (Reino Unido) y “pl” (Polonia), todos ellos europeos. Si bien estos países no tienen fama de censurar a sus habitantes, estos resultados reafirman los obtenidos en las conexiones de entrada a la red. Es decir, los habitantes de países más desarrollados y con mayor penetración de internet tienen una mayor preocupación por mantenerse anónimos en Internet.

Cabe aclarar que el que páginas con el nombre de dominio de un país tengan resoluciones DNS no necesariamente implica que son habitantes de ese país quienes las están visitando, ya que cualquier persona puede visitar páginas de todo el mundo. Sin embargo, se utiliza como una forma de medir la actividad de los usuarios por el hecho de que los sitios asociados a esos nombres de dominio fueron realizados para ser accedidos por personas de esos países.

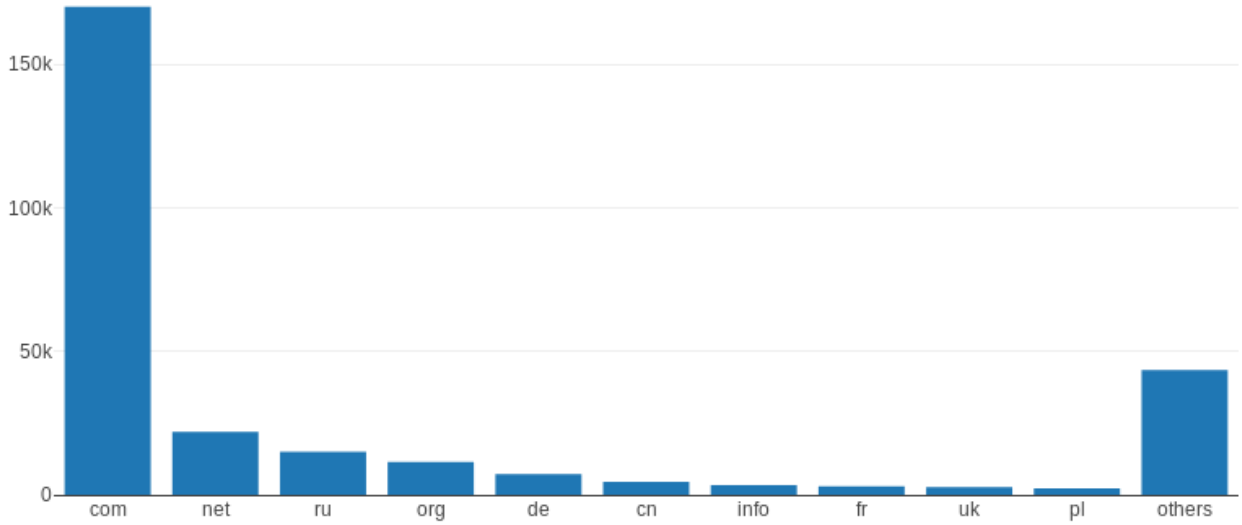

Imagen 4.11. Top 10 de dominios de nivel superior consultados.

Al visualizar el top 15 de los dominios de segundo nivel más consultados puede observarse que dominios como “google.com” y “facebook.com” son dominios altamente visitados, entre otros muy populares como “wordpress.com”, “apple.com” y “yahoo.com”. Este resultado confirma la suposición anterior sobre el origen de los principales usuarios de la red pues se observa el dominio “mail.ru”, correspondiente a servicios de correo en Rusia y “qq.com”, correspondiente a una popular aplicación de mensajería en China. En la imagen 4.12 se tiene la gráfica donde se aprecian estos resultados.

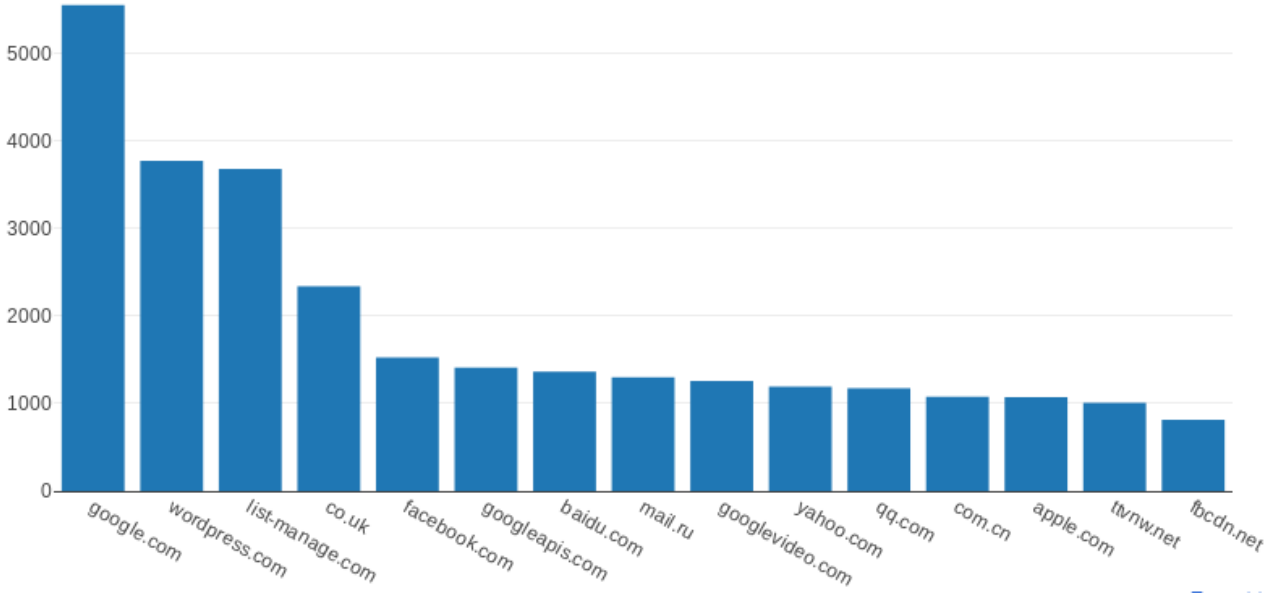

Imagen 4.12. Top 15 de dominios de segundo nivel más consultados.

4.3.3 Visualización geográfica de actividad basada en nombres de dominio

Este análisis tiene por objetivo complementar la vista realizada a partir de las conexiones de entrada a la red, pero considerando como base del análisis el origen de los nombres de dominio consultados y no la ubicación geográfica de direcciones IP.

Se realizó un mapa similar al generado con las conexiones de entrada pero utilizando los nombres de dominio para conocer los principales países que usan la red. Para lograr esto, se consideró en primer lugar el TLD de cada dominio en caso de ser el TLD correspondiente a un país y, en su defecto, se tomó en cuenta el registro de las bases de datos WHOIS para ese dominio. De esta forma se obtuvo un conteo de a qué país pertenece cada dominio y, con ayuda de las bases de datos públicas de GeoIP, se obtuvo el resultado mostrado en la imagen 4.13.

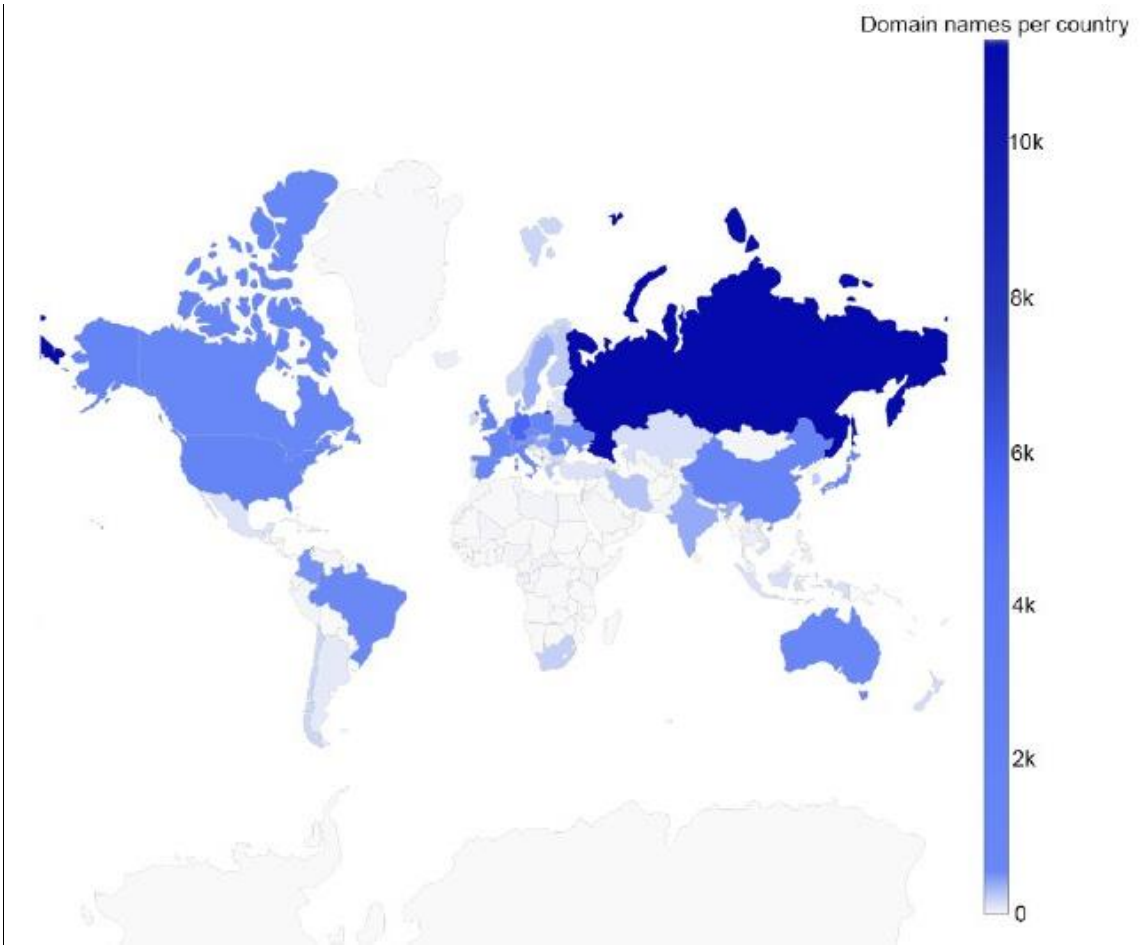

Imagen 4.13. Actividad por país basada en los nombres de dominio consultados.

Del mapa anteriormente mostrado se puede ver que los dominios mayormente consultados pertenecen a Rusia con más de 10 mil, contrarrestando lo mostrado en el mapa generado por las conexiones de entrada donde Estados Unidos tenía la mayor cantidad de conexiones. De igual manera se muestra que China y otros países de Asia tienen una mayor presencia en cuanto a cantidad de dominios consultados. Lo anterior contrasta con América Latina y África, pues en ambos análisis estas zonas muestran un comportamiento muy similar.

El hecho de que China no muestre ninguna conexión de entrada y tenga miles de dominios consultados se puede explicar con los puentes. Lo anterior se debe a que el gobierno chino ha implementado fuertes medidas para restringir el acceso a Tor, bloqueando todas las direcciones IP de los nodos de entrada. Sin embargo, al revisar la actividad de este país en un nodo de salida se puede observar que se trata de un país con un alto índice de actividad en la red, lo que indica que sobrepasan las medidas de su país haciendo uso de los nodos de entrada llamados puentes, cuyo direccionamiento IP no es del dominio público. Este mismo puede aplicar para Rusia, ya que se ve mucha actividad hacia dominios rusos en comparación con las conexiones de entrada, lo que podría significar que los habitantes de este país también tienen una tendencia a utilizar puentes para acceder a Tor.

Salvo por los resultados tan contrastantes existentes en los países como China y Rusia, se observa un resultado muy similar al obtenido en el análisis de conexiones de entrada a Tor. Esto refuerza la idea de que el comportamiento de los usuarios de la red abierta se refleja en la red Tor, por lo que esto no permite asociar el comportamiento de la red a actividades maliciosas.

4.3.4 Actividad de México en la red Tor basada en nombres de dominio

Finalmente, se decidió obtener datos para crear una idea sobre la actividad hacia sitios mexicanos que se realiza a través de la red Tor, usando nuevamente los nombres de dominio resueltos. Esto se logró haciendo un par de búsquedas con *fgrep*, utilizando las cadenas “mex” y “.mx”, obteniendo de esta manera un listado de todos los dominios que probablemente fueron visitados por mexicanos. Una vez obtenido el listado, se usó la misma herramienta de visualización desarrollada anteriormente para graficar los 10 dominios más veces consultados. Esta gráfica puede observarse en la imagen 4.14. Cabe resaltar que existe una gran tendencia por visitar sitios de compras en línea, pues más de la mitad de los dominios resultantes corresponden a aplicaciones web con este objetivo. Es decir, este comportamiento difícilmente puede asociarse directamente con actividad maliciosa, ya que realizar compras en línea cada vez es más común en los usuarios de internet, por lo que estos resultados pueden ser solo un reflejo de lo que sucede en la red abierta.

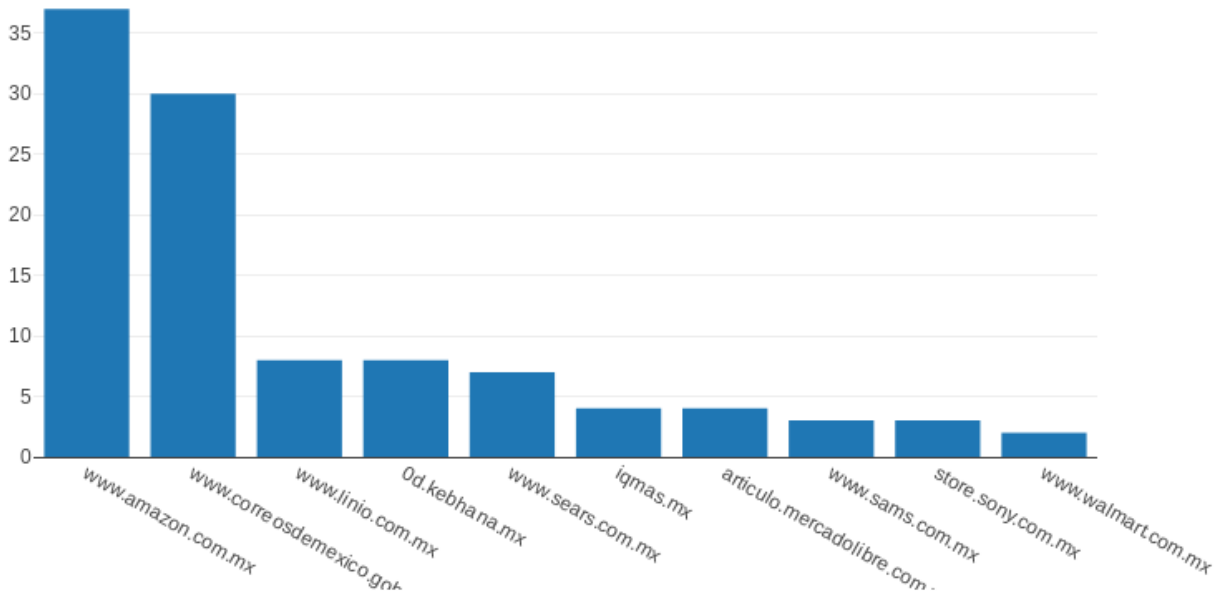

Imagen 4.14. Top 10 de dominios mexicanos consultados.

4.4 Servicios de la capa de aplicación

Este análisis tiene por objetivo conocer cuáles son los protocolos utilizados por los usuarios de la red, ya que existen diversos protocolos con una mayor tendencia a ser usados con fines maliciosos, por ejemplo, los protocolos de correo electrónico e IRC (comúnmente usados por familias de malware de tipo bot). Asimismo, los resultados de este análisis pueden contrastarse con la búsqueda específica de actividad maliciosa, ya que puede aparecer mucha actividad maliciosa relativa en un puerto muy poco usado en comparación con el total del tráfico o viceversa.

Para tener un resultado más real, en este análisis se configuró una política de salida poco estricta, habilitando múltiples puertos TCP. Cabe recordar que en este análisis no se verán protocolos como NTP o SNMP que utilizan UDP como protocolo de capa de transporte, pues la red Tor está diseñada para enrutar únicamente tráfico que utiliza el protocolo TCP.

4.4.1 Análisis estadístico y de flujos: tcpflow, tcpdstat, python

Para obtener una idea general sobre los principales protocolos de capa de aplicación se utilizó la herramienta *tcpflow.* Esta herramienta puede reconstruir los flujos de conexiones a través de los números de secuencia y confirmación que utiliza TCP. Esto permite, entre otras funcionalidades, obtener una gráfica de los puertos TCP con más conexiones en determinada captura de tráfico. Dicha herramienta se utilizó en todas las capturas de tráfico, obteniendo en todas resultados similares al mostrado en las imágenes 4.15 y 4.16, que muestran la relación de tráfico por minuto durante dos horas, con diez horas de diferencia. Este formato de gráfica ayuda, no solo a comprender el uso del nodo en una hora en específico, sino a ver la evolución que este tuvo durante ese lapso.

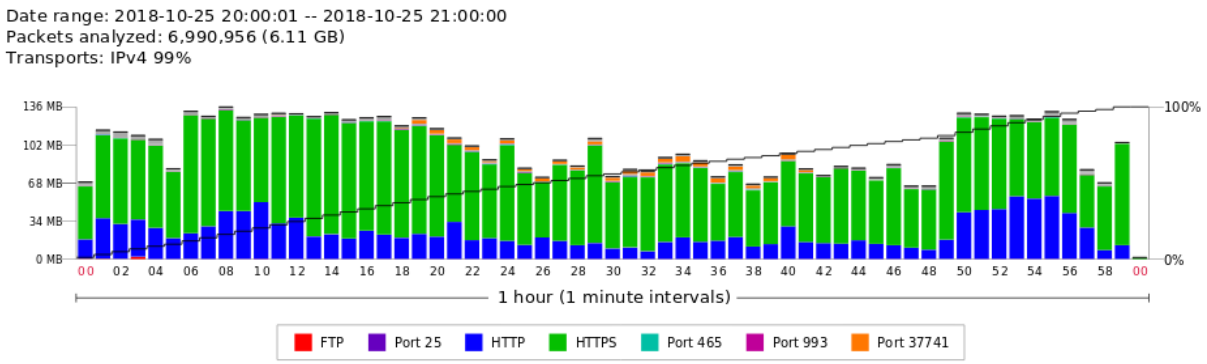

Imagen 4.15. Relación 1 de cantidad de tráfico por puerto en un archivo pcap.

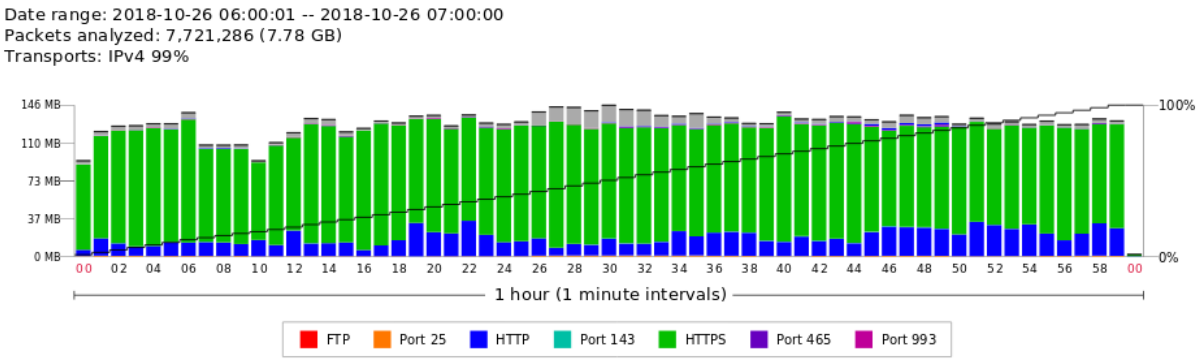

Imagen 4.16. Relación 2 de cantidad de tráfico por puerto en un archivo pcap.

Puede observarse que la mayor parte del tráfico corresponde al protocolo HTTPS, por lo que se deduce que el principal uso que le dan los usuarios a la red es visitar páginas web. Asimismo, se observa una gran tendencia a visitar sitios que utilizan SSL/TLS, es decir, que el contenido de las comunicaciones entre los servidores y los clientes se encuentra cifrado. Considerando el tráfico capturado durante las 19 horas, en promedio se tiene que el tráfico HTTPS corresponde al 65.58 % del total del tráfico. Esto indica que más del 60% de la información que circula a través de la red Tor está totalmente protegida, no solamente por el protocolo cebolla de Tor, sino por el cifrado a nivel de capa de aplicación proporcionado por HTTPS. Esto se debe a que cuando un usuario usa Tor para visitar sitios protegidos con HTTPS, está entregando la menor cantidad posible de información a cualquiera que se encuentre espiando las comunicaciones. Por ejemplo, un administrador de un nodo de salida únicamente conoce qué sitio se está visitando, pero no el tipo de información transmitida a él credenciales de acceso (Electronic Frontier Foundation, 2019). El hecho de que la mayor parte del tráfico se encuentre cifrado puede deberse no solo a las buenas prácticas de los usuarios, sino a que utilizan el navegador Tor para utilizar la red. Lo anterior debido a que el navegador Tor implementa diversas configuraciones para mejorar la seguridad de los datos que viajan por la red, como es la extensión para navegadores *HTTPS Everywhere*, que le da prioridad al uso de HTTPS sobre HTTP (Electronic Frontier Foundation, 2019).

Las imágenes 4.17 y 4.18 muestran los principales puertos usados en las mismas capturas de las imágenes anteriores. Estos resultados indican que los protocolos de aplicación más populares dentro de la red Tor son HTTP y HTTPS, por lo que esto descarta que la red sea utilizada principalmente con fines de explotación de protocolos diferentes a estos. Sin embargo, no descarta la existencia de actividad maliciosa, por lo que se deben realizar análisis enfocados en estos protocolos en específico.

Sin embargo, dado que la herramienta *tcpdump* no puede configurarse para entregar gráficas de forma tabular y a la cantidad de tráfico perteneciente a los puertos 80 y

443, no es posible visualizar correctamente la cantidad de tráfico en los puertos diferentes a estos.

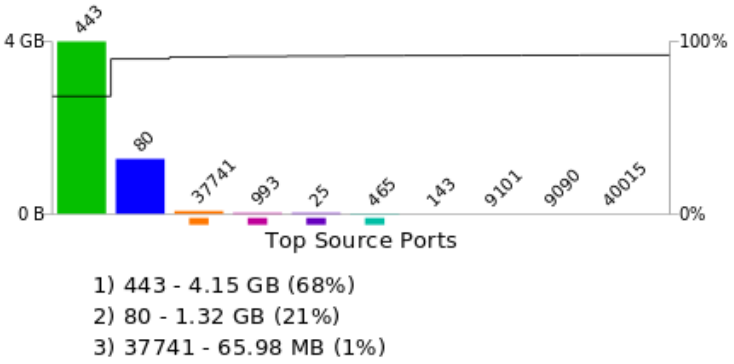

Imagen 4.17. Porcentaje 1 de tráfico en los principales puertos.

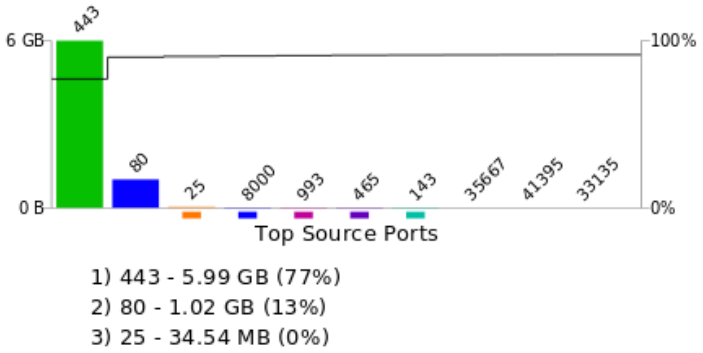

Imagen 4.18. Porcentaje 2 de tráfico en los principales puertos.

La herramienta *tcpflow* permite obtener un top 3 de los puertos más vistos, sin embargo, la política de salida se configuró para soportar más de 70 puertos de salida. Para obtener la cantidad de conexiones por servicio se decidió desarrollar una herramienta en Python que permite contar los flujos en una o varias capturas y generar una gráfica con una cantidad explícita de puertos a graficar para obtener una mejor visualización de los puertos más usados. Cabe destacar que esta herramienta no muestra la cantidad de tráfico por puerto basado en el número de paquetes, sino la cantidad de flujos existentes clasificados por el puerto destino. Lo que proporciona una mejor idea del uso que se le da a la red.

Puesto que *tcpflow* mostró que la mayor cantidad de tráfico pertenece al puerto 443 se decidió hacer un nuevo filtro con *tcpdump* a las capturas para eliminar todo el tráfico correspondiente a este puerto y, de esta manera, permitir que la herramienta desarrollada procese más rápidamente el tráfico capturado y obtener una visualización más clara sobre cuánto uso se le dio al resto de los puertos configurados en la política de salida. La ejecución de la herramienta desarrollada se muestra en la imagen 4.19 donde se observa el archivo que se encuentra procesando y la cantidad de paquetes procesados en ella.

```
# python flow_counter.py -i ifile.txt -o graph.html -g 10 -s
    1) Proccesing file /mnt/tesis/no_https_traffic/new_dump_2018-10-26_05_00_01.pcap
            100000 packets
            200000 packets
            300000 packets
            400000 packets
            500000 packets
            600000 packets
            700000 packets
            800000 packets
            900000 packets
            1000000 packets
```

Imagen 4.19. Ejecución de herramienta en Python que analiza y grafica los flujos de una captura.

Adicionalmente, la gráfica resultante de los principales 10 puertos obtenida con la herramienta aplicada a una sola captura de tráfico se muestra en la imagen 4.20, donde se puede apreciar que predominan las conexiones a puertos comúnmente utilizados para servicios web como el puerto 80 (puerto por defecto para el protocolo HTTP) y los puerto 8080, 8000 (puertos por defecto en algunos servidores HTTP como Apache Tomcat y servidores proxy HTTP). De igual manera aparecen en gran medida los puertos 25, 143, 993, 465 y 587, puertos comúnmente usados en servicios de correo electrónico.

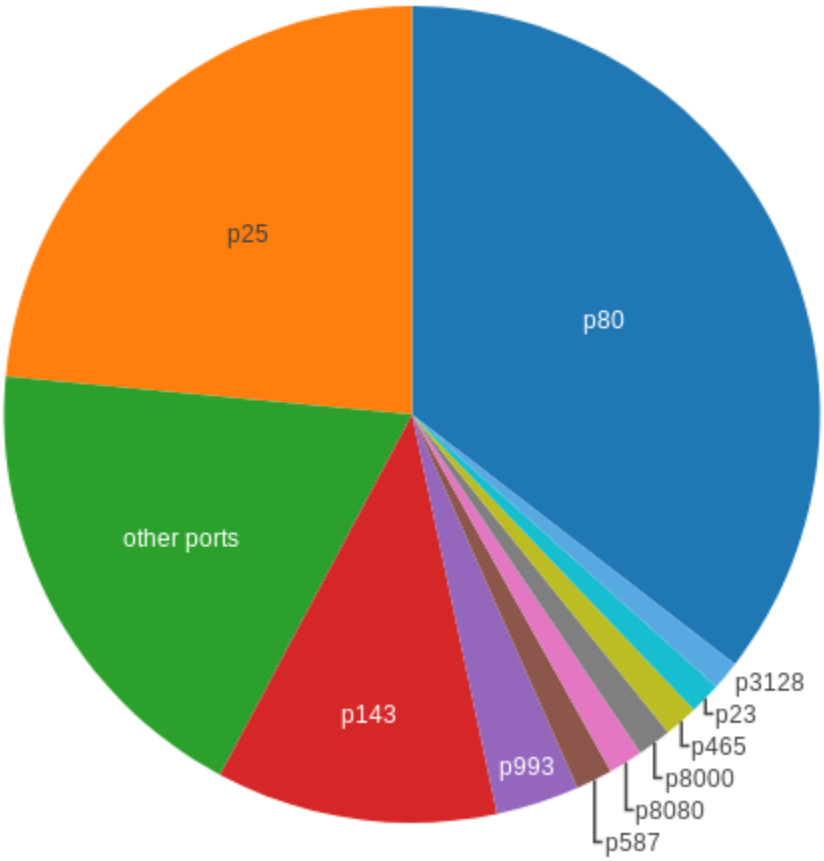

Imagen 4.20. Gráfica de flujos por puerto obtenida de un único archivo de captura.

Posteriormente a generar la gráfica de los puertos más usados en una captura, se obtuvo la gráfica para el total del tráfico capturado, obteniendo los resultados mostrados en la imagen 4.21. Se muestra un comportamiento prácticamente idéntico al visto en una sola captura, lo que indica que no es un comportamiento extraordinario, sino que la actividad general de los usuarios dentro de la red es el envío de correos y visitas a páginas web. Empero, cabe destacar que la política de salida poco restrictiva es aprovechada ampliamente ya que muestra que el resto de puertos, aunque en menor medida, también son utilizados.

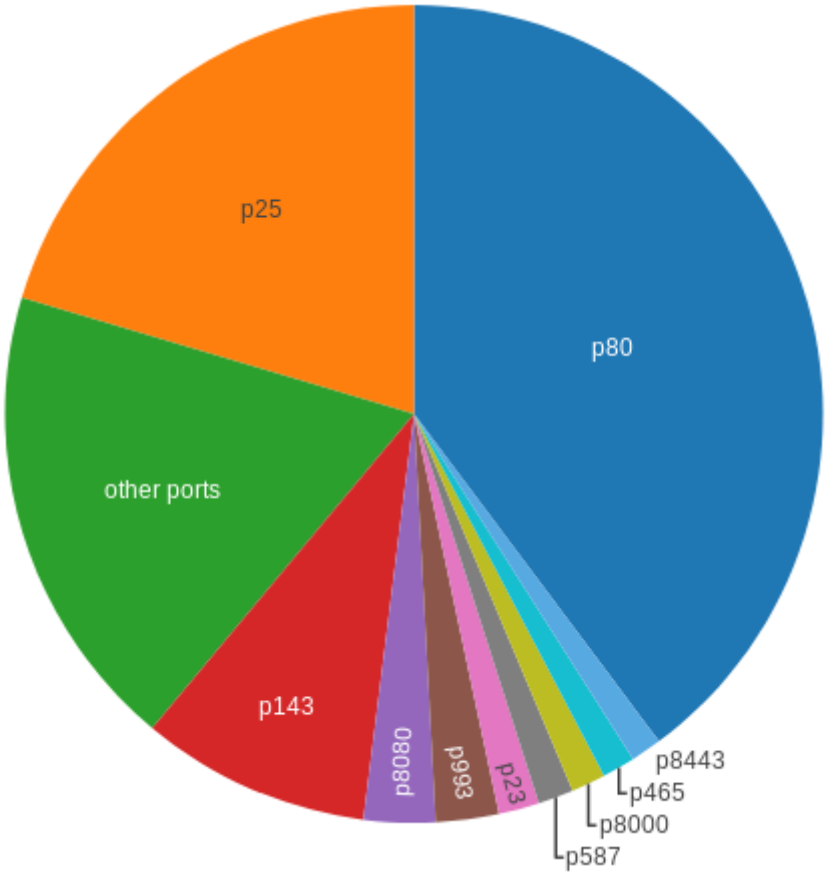

Imagen 4.21. Gráfica de flujos por puerto obtenida de todos los archivos de captura.

Otra herramienta muy común para analizar de forma estadística el comportamiento de una red es *tcpdstat*. Esta herramienta muestra, entre otros datos, el lapso de duración de la captura, el número de paquetes que la conforman y el número de flujos existentes. Esta herramienta se aplicó de forma individual sobre el mismo archivo sobre el que se usó la herramienta desarrollada en Python. Un extracto de la salida de este programa se muestra en la imagen 4.22.

```
DumpFile:  /mnt/tesis/no_https_traffic/new_dump_2018-10-26_05_00_01.pcap
FileSize: 186.42MB
Id: 201810252200
StartTime: Thu Oct 25 22:00:01 2018
EndTime:   Thu Oct 25 23:00:00 2018
TotalTime: 3599.78 seconds
TotalCapSize: 170.70MB  CapLen: 7354 bytes
# of packets: 1030396 (170.70MB)
AvgRate: 397.93Kbps  stddev:476.73K   PeakRate: 7.02Mbps

### IP flow (unique src/dst pair) Information ###
# of flows: 17963  (avg. 57.36 pkts/flow)
```

Imagen 4.22. Salida de tcpdstat con análisis estadístico preliminar.

Tcpdstat es de especial utilidad y funciona de forma complementaria al análisis de flujos, ya que puede mostrar de forma granular la actividad de los protocolos de aplicación basándose en el número de paquetes por protocolo. El extracto de la salida de este programa se muestra en la imagen 4.23, donde se observa que el análisis basado en flujos con la herramienta en Python y el análisis basado en número de paquetes es congruente, ya que predomina la actividad basada en los protocolos HTTP y de correo electrónico.

```
### Protocol Breakdown ###
<<<<
     protocol           packets                 bytes           bytes/pkt
------------------------------------------------------------------------------
[0] total          1030396 (100.00%)        178991639 (100.00%)    173.71
[1] ip             1030396 (100.00%)        178991639 (100.00%)    173.71
[2]  tcp           1030396 (100.00%)        178991639 (100.00%)    173.71
[3]   ftpdata           57 (  0.01%)            26636 (  0.01%)    467.30
[3]   ftp              439 (  0.04%)            32415 (  0.02%)     73.84
[3]   telnet           945 (  0.09%)            67930 (  0.04%)     71.88
[3]   smtp          245131 ( 23.79%)         82220169 ( 45.94%)    335.41
[3]   dns             4817 (  0.47%)           359181 (  0.20%)     74.57
[3]   http(s)          150 (  0.01%)            79008 (  0.04%)    526.72
[3]   http(c)       536049 ( 52.02%)         69250501 ( 38.69%)    129.19
[3]   kerb5            684 (  0.07%)            50466 (  0.03%)     73.78
[3]   pop3             740 (  0.07%)            54230 (  0.03%)     73.28
[3]   ident            114 (  0.01%)             6156 (  0.00%)     54.00
[3]   imap           43860 (  4.26%)          3649484 (  2.04%)     83.21
[3]   ldap             597 (  0.06%)            43978 (  0.02%)     73.66
[3]   ms-ds           1107 (  0.11%)            71469 (  0.04%)     64.56
[3]   rtsp             541 (  0.05%)            40034 (  0.02%)     74.00
[3]   socks              8 (  0.00%)              450 (  0.00%)     56.25
[3]   mssql-s           14 (  0.00%)              798 (  0.00%)     57.00
[3]   squid            834 (  0.08%)            61452 (  0.03%)     73.68
[3]   mysql              1 (  0.00%)               60 (  0.00%)     60.00
[3]   icecast         5321 (  0.52%)           418283 (  0.23%)     78.61
[3]   gnu6349            3 (  0.00%)              162 (  0.00%)     54.00
[3]   irc6666             9 (  0.00%)              652 (  0.00%)     72.44
[3]   irc6667          7394 (  0.72%)           768452 (  0.43%)    103.93
[3]   irc6668             8 (  0.00%)              544 (  0.00%)     68.00
[3]   irc7000             1 (  0.00%)               60 (  0.00%)     60.00
[3]   http-a           1764 (  0.17%)           155153 (  0.09%)     87.96
[3]   http-tw             3 (  0.00%)              174 (  0.00%)     58.00
[3]   other           179805 ( 17.45%)         21633742 ( 12.09%)    120.32
```

Imagen 4.23. Salida de tcpdstat con clasificación de puertos más usados basado en número de paquetes.

4.5 Dispositivos usados para acceder a la red

El objetivo de este análisis es determinar si la tendencia general de los usuarios de internet de usar cada vez más los dispositivos móviles en lugar de computadoras de escritorio se ve reflejada en la red Tor, ya que una navegaciónrealizada desde dispositivos móviles difícilmente se debería a ataques a la seguridad de la información de los activos de internet, debido al poco poder de procesamiento con el que estos dispositivos cuentan.

Para lograr esto, el análisis se basa en el protocolo HTTP pues es el protocolo más usado dentro de la red Tor después de HTTPS y el análisis en este último no es viable pues el tráfico está cifrado. Adicionalmente, estos protocolos utilizan la misma estructura en sus peticiones y respuestas, teniendo ambos un campo dedicado a conocer qué tipo de clientes es el que realiza la petición.

El protocolo HTTP se basa en dos tipos de mensajes: las peticiones hechas por los clientes y las respuestas del servidor. En ambos casos, los mensajes incluyen

encabezados que le dan información a su contraparte. Por ejemplo, en el lado del servidor puede configurarse el encabezado 'Server' que expone la versión del servidor web. De este mismo modo, el cliente envía el encabezado 'User-Agent', el cual le indica al servidor de qué cliente se trata, es decir, los diferentes navegadores envían diferentes agentes de usuario (Manners, 2012). Cabe destacar que el agente de usuario es un campo que, al ser enviado por los clientes HTTP, es fácilmente modificable. Es decir, los resultados del análisis no son totalmente confiables y están en función de cuántas personas no utilicen las configuraciones por defecto, decidiendo modificar este campo.

4.5.1 Análisis de datos de contenido: Agentes de usuario

Para obtener un listado de todos los agentes de usuario se utilizó una herramienta llamada *ngrep* (network grep), la cual implementa las características más comunes del popular programa *grep* de los sistemas *unix* pero aplicado sobre tráfico de red para relacionar determinada expresión con los datos de los diferentes paquetes analizados, así como una herramienta de desarrollo propio para contar correctamente los agentes de usuario considerando el flujo al que pertenecen. En total se extrajeron 557038 agentes de usuario de diferentes flujos, siendo los 10 agentes más vistos los mostrados en la tabla 4.1. Marqué con un asterisco el agente de usuario utilizado por el navegador Tor, el cual corresponde al agente de usuario con más apariciones.

Agente de usuario	**Número de apariciones**
Mozilla/5.0 (Windows NT 6.1; rv:60.0) Gecko/20100101 Firefox/60.0 *	28516
Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0 Google-HTTP-Java-Client/1.23.0 (gzip)	16616
Mozilla/5.3 (compatible; bingbot/2.1; +http://www.bing.com/bot.html)	13732
eBayiPhone/2.5.2	6532
Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b) Gecko/20030516 Mozilla Firebird/0.6	6482
Mozilla/5.0 (Windows; U; Windows NT 6.1; tr-TR) AppleWebKit/533.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27	5889
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36	5681
Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X)	5673

AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36(KHTML, like Gecko) Chrome/41.0.2228.0 Safari/535.39	5648
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.62 Safari/537.36	4893

Tabla 4.1. Agentes de usuario más comunes

Para que un dispositivo logre obtener las versiones móviles de las diferentes páginas, le indica al servidor web que se trata de un dispositivo móvil agregando determinadas palabras en el agente de usuario, como “Mobile”. Por lo que, para obtener el porcentaje de estos dispositivos con respecto al total, se filtraron los agentes encontrados con la herramienta desarrollada y *ngrep* utilizando las cadenas “Mobile” y “Android”. Dicho filtro se muestra en la imagen 4.24.

```
user@TraffAnalysis:/opt/devices$ grep -E 'mobile|android' -i devices_output.txt
> mobile_agents.txt
user@TraffAnalysis:/opt/devices$ head -n 3 mobile_agents.txt
Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML,
 like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25 (compatible; Googlebot/2
.1; +http://www.google.com/bot.html): 5673
Mozilla/5.0 (BlackBerry; U; BlackBerry 9850; en) AppleWebKit/534.11+ (KHTML, lik
e Gecko) Version/7.0.0.254 Mobile Safari/534.11+: 2179
Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gec
ko ) Version/5.1 Mobile/9B176 Safari/7534.48.3: 1930
```

Imagen 4.24. Filtro a partir de cadenas para obtener los agentes de usuarios de dispositivos móviles.

Finalmente, se encontraron 38985 conexiones que enviaron agentes de usuario de dispositivos móviles en las peticiones HTTP. A pesar de que este número podría parecer muy grande, corresponde únicamente a un 7% del total de peticiones. Puesto que el agente de usuario es un valor establecido en el lado del cliente es fácilmente modificable, lo que implica una posibilidad de que no todos los agentes de usuario móviles encontrados hayan salido efectivamente de un dispositivo móvil. Por lo que ese porcentaje realmente podría ser menor. Esto nos indica que, aunque es cada vez más común utilizar celulares y tabletas para acceder a internet, no se observa esta misma tendencia en la red Tor, donde principalmente se siguen utilizando computadoras personales.

Lo anterior implica que, al menos el 7% de las peticiones HTTP realizadas en la red Tor tienen una probabilidad muy baja de ser maliciosas. Adicionalmente, el bajo porcentaje de uso de dispositivos móviles puede deberse a una gran cantidad de factores, como por ejemplo, la renuencia de las tiendas de aplicaciones a incluir navegadores distintos a los predefinidos o a la desconfianza de las personas a utilizar los navegadores disponibles para acceder a la red Tor. Adicionalmente, muchas

personas desconfían de los dispositivos móviles para cuidar de su privacidad, ya que los ven como posibles mecanismos de espionaje. Esta relación se muestra en la imagen 4.25.

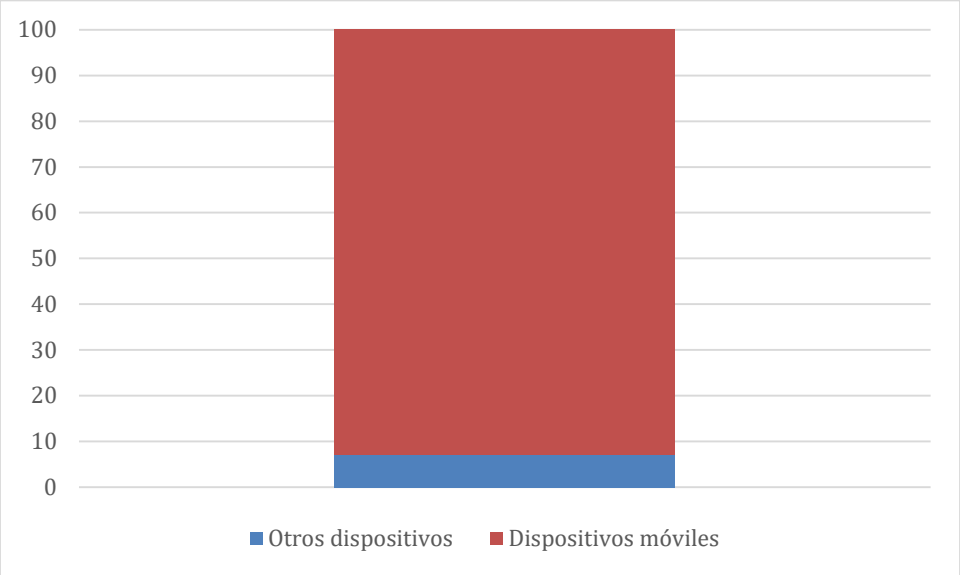

Imagen 4.25. Relación de dispositivos móviles y estáticos.

4.6 Búsqueda de actividad maliciosa

El objetivo de este análisis es encontrar los principales ataques informáticos que se llevan a cabo utilizando la red Tor como medio de anonimato y la cantidad de acciones potencialmente maliciosas realizadas por sus usuarios para contar con datos más precisos sobre este tipo de actividades. Adicionalmente, de esta forma se pretende determinar la viabilidad del análisis de esta red para obtener información relevante sobre nuevas técnicas, campañas de ataques o sitios potencialmente vulnerables.

4.6.1 Análisis de datos de contenido: Correo electrónico

Para el primer análisis, se revisó únicamente el tráfico correspondiente el puerto TCP 25 puesto que es el puerto por defecto para el protocolo de capa de aplicación SMTP, un protocolo de envío de correos electrónicos que puede ser fácilmente usado para ataques sencillos pero muy comunes como spam y phishing. Para lograr esto, se utilizó la herramienta *xplico* que sirve para hacer análisis forense en tráfico de red y, entre otras funcionalidades para distintos protocolos, sirve para contar los correos enviados y recibidos, realizar operaciones básicas de ordenación y búsqueda y guardar los correos en formato *eml* para su análisis manual.

Esta herramienta permite acceder a sus funcionalidades a través de una aplicación web y permite capturar y analizar tráfico en tiempo real, así como cargar archivos en

formato *pcap*. Para este análisis, se cargaron en la aplicación archivos pcap que resultaron de un nuevo filtro al tráfico capturado en los que únicamente se dejó el tráfico correspondiente al puerto 25. Como se muestra en la imagen 4.26 una vez que terminó de procesar los archivos se encontró que en total salieron 75249 correos del nodo de Tor, lo que da un promedio de 3960 correos por hora.

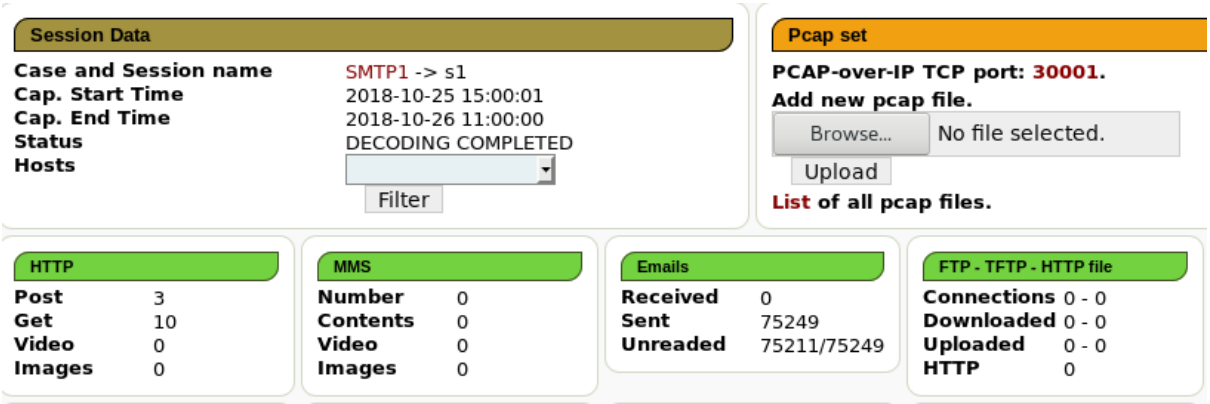

Imagen 4.26. Consola principal de Xplico con el conteo de correos enviados.

Como se trata de una gran cantidad de correos electrónicos y de forma local no se implementa ningún tipo de medida para evitar el correo potencialmente malicioso, existe una alta probabilidad de encontrar spam. Debido a la gran cantidad de correos extraídos se realizó un filtro para mostrar únicamente aquellos que en algún campo del correo contengan la cadena ".mx", es decir, el dominio de nivel superior correspondiente a México y, de esta forma, encontrar campañas de correos dirigidos a personas de este país o salientes de algún servidor de correos de este país. De esta forma se encontró un total de 279 correos, de los cuales se determinó que un 100% son maliciosos o no deseados. Cabe resaltar que muchas personas utilizan servicios de correo electrónico que no necesariamente utilizan un TLD de algún país, como "gmail.com" o "outlook.com", por lo que este análisis está enfocado en encontrar posibles víctimas que utilicen servicios de correo que sí los usen debido a la posibilidad de filtrado que esta consideración brinda.

En la imagen 4.27 puede observarse un ejemplo de un correo destinado a que se abra un archivo malicioso, asegurando que se trata de una confirmación de pago. En total se extrajeron 27 de estos correos que incluían a un destinatario cuyo correo incluía el dominio ".mx", sin embargo, al buscar más muestras de este correo eliminando el filtro anterior, se encontró que no se trata de una campaña dirigida a México, sino de una campaña mundial, pues se extrajeron más de 520 muestras de este correo, juntando entre todas ellas más de 26,000 destinatarios de todo el mundo.

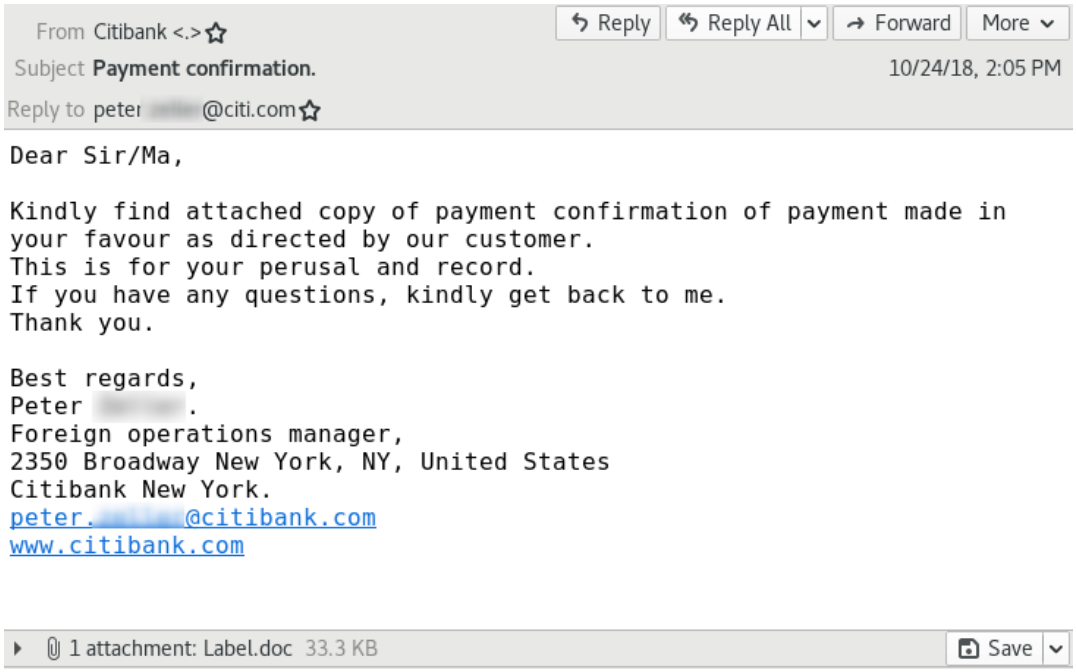

Imagen 4.27. Correo con archivo malicioso.

A continuación, en la imagen 4.28, se muestra un ejemplo de correo electrónico no deseado en español aparentemente dirigido a la sociedad mexicana. De este correo se encontraron más de 30 muestras en donde todos los remitentes pertenecen a dominios diferentes; el principal factor en común, además del contenido, es que los destinatarios pertenecen a dominios mexicanos, incluyendo "derechoshumanos.org.mx", "elfinanciero.com.mx" y "nafin.gob.mx", entre otros.

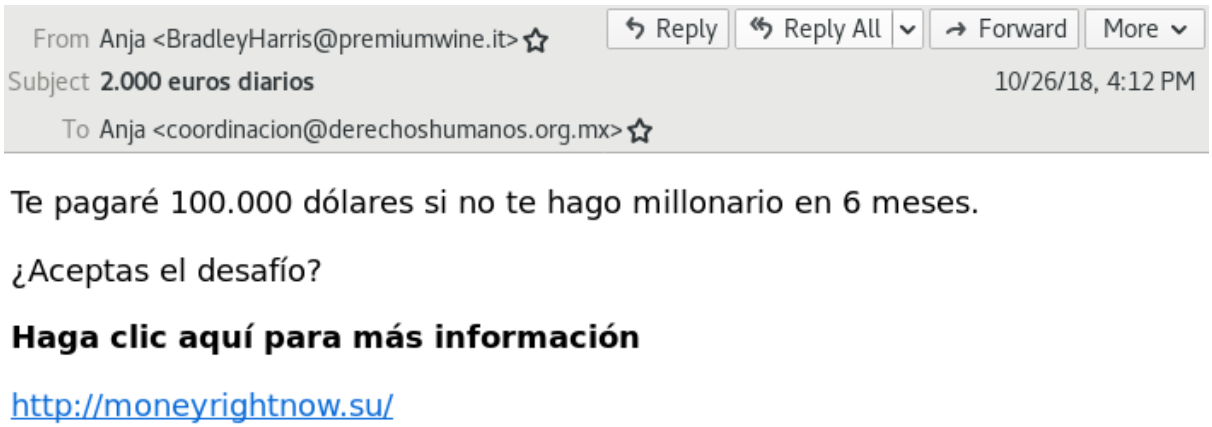

Imagen 4.28. Correo no deseado suplantando la identidad de un dominio mexicano.

En la imagen 4.29 se muestra un correo en portugués que, a diferencia de los ejemplos anteriores, tiene el dominio mexicano "chihuahua.com.mx" en el remitente mientras que el destinatario es brasileño. Este correo y otros con las mismas características mencionadas forman parte de una campaña en la que se encontraron 105 muestras y más de 500 destinatarios brasileños. Esto indica que, probablemente, un servidor de correo con ese dominio se encuentra mal configurado, permitiendo un

"*Open Relay*" o envío de correos a cualquier destinatario sin validación, siendo encontrado y aprovechado por *spammers* brasileños.

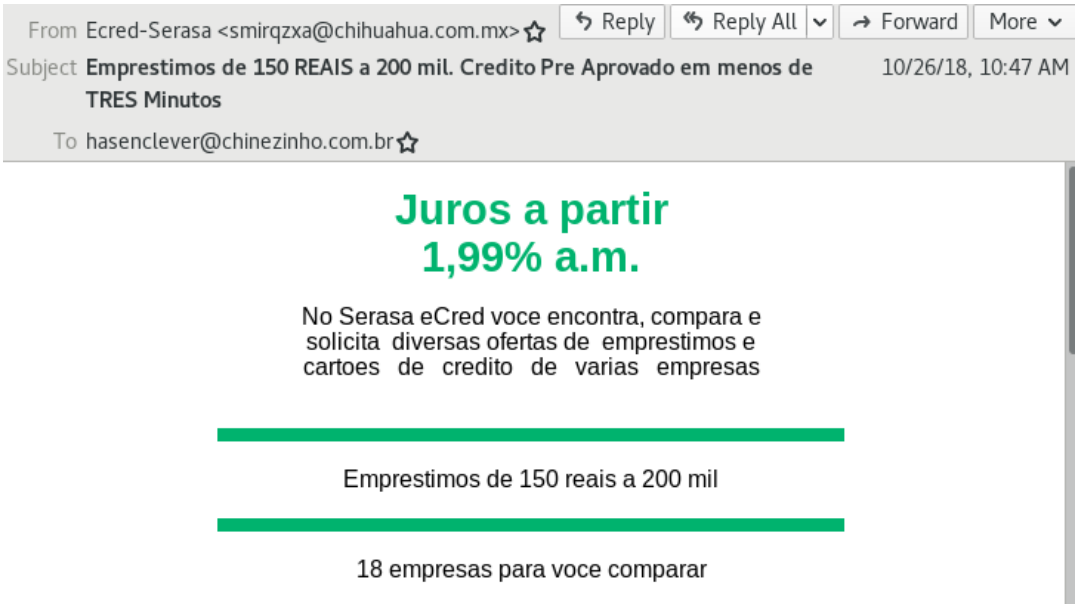

Imagen 4.29. Correo electrónico enviado desde un servidor con Open Relay.

A pesar de que la muestra de correos analizados de forma manual es mínima en comparación con el total de correos, se puede decir que los protocolos de correo electrónico son principalmente aprovechados para enviar mensajes maliciosos a través de la red Tor, pues el análisis manual mostró que un 100% de los correos tenía este objetivo. En este caso, el análisis muestra un fuerte contraste en comparación con la red abierta pues, aunque seguramente existen muchos correos electrónicos maliciosos en esta, no el 100% de los correos lo es.

4.6.2 Análisis de datos de contenido: Agentes de usuario

El siguiente protocolo a analizar de forma manual en busca de actividad maliciosa fue HTTP, por tratarse del protocolo más común en la red. Se analizaron las cabeceras de las peticiones HTTP, en especial la cabecera de "User-Agent" que, como ya se explicó anteriormente, sirve para indicarle al servidor qué tipo de cliente lo visita. Sin embargo, al tratarse de un campo dependiente del cliente es fácilmente modificable (Manners, 2012). Este campo suele modificarse como una medida básica para pasar algunos filtros de sistemas de detección de intrusos (IDS) o firewalls de aplicaciones web (WAFs). A pesar de poder servir como un filtro básico a medidas de seguridad, hay programas que se encargan de hacer peticiones HTTP que no cambian este valor, como malware o los bots de buscadores como Google. Otra razón por la cual son modificados son para efectuar escaneos de vulnerabilidades o ataques a aplicaciones web que hagan uso de esta cabecera como, por ejemplo, guardar su valor en una base de datos y explotar alguna vulnerabilidad en estas aplicaciones. Es por eso que esta cabecera es de utilidad al momento de buscar actividad maliciosa. Se aprovecharon los datos extraídos en el anterior análisis sobre los dispositivos para

entrar a la red, es decir, se utilizaron los resultados de las herramientas *ngrep* y una de desarrollo propio.

Sin tratarse de una lista exhaustiva para filtrado y detección de agentes de usuario maliciosos, se trabajó este análisis utilizando la siguiente lista de cadenas. Se revisó sin sensibilidad a mayúsculas que los agentes de usuario tuvieran como subcadena alguna de las siguientes cadenas para considerarlos maliciosos o, en el mejor de los casos, sospechosos. Esto se debe a que suelen hacer referencia a herramientas comúnmente usadas durante pruebas de penetración, ya sea para escanear puertos, servicios y versiones o, incluso, vulnerabilidades. A continuación se muestra una lista de cadenas, la razón de su uso y la cantidad de apariciones.

Cadena	**Razón**	**Repeticiones**
‘iceweasel’	Navegador por defecto en el sistema operativo Kali Linux que está orientado a las pruebas de penetración y seguridad informática en general.	155
‘acunetix’	Escáner de vulnerabilidades en aplicaciones web.	23
‘sqlmap’	*Framework* para buscar y explotar inyecciones SQL en aplicaciones web.	159
‘python’	Lenguaje de programación especialmente popular entre las personas que hacen pruebas de penetración por bibliotecas como Requests.	6204
‘nessus’	Escáner de vulnerabilidades con módulos para el protocolo HTTP.	17
‘dirb’	Programa que sirve para encontrar, mediante fuerza bruta, archivos y directorios sensibles en servicios web.	7
‘nmap’	Programa que es usado principalmente para escanear puertos y redes, pero que tiene scripts que aumentan su funcionalidad, incluyendo scripts para escanear vulnerabilidades en aplicaciones web.	41
‘nikto’	Escáner gratuito y libre de vulnerabilidades en aplicaciones web.	0
‘meterpreter’	Programa que ayuda principalmente en el proceso de explotación de vulnerabilidades. Muy usado contra servicios web.	0
‘null’	Un agente de usuario “nulo” es sospechoso.	2
‘crawler’	Tipo de programa que se encarga de obtener información de aplicaciones web a través de múltiples peticiones HTTP.	2714
‘cve’	Cadena que forma parte de la clasificación más común de vulnerabilidades. Indicaría de forma explícita que intenta explotar determinada vulnerabilidad.	4
‘xss’	Contracción de “Cross-Site Scripting”, una vulnerabilidad de inyección de código en el lado del cliente.	23
‘src’	Cadena común al explotar XSS para indicar el origen del script a ejecutar.	4
‘onload’	Cadena común al explotar XSS para que se ejecute el código al cargar determinado elemento de la	0

	página.	
‘onmouseover’	Cadena común al explotar XSS para que se ejecute el código al pasar el mouse sobre determinado elemento de la página.	3
‘sqli’	Contracción de “SQL Injection”, una vulnerabilidad de inyección de código en el lado del servidor.	3
‘ or ‘	Cadena común al explotar SQLi, utilizada para crear tautologías en el manejador de bases de datos.	27
‘sleep’	Cadena común al explotar SQLi, utilizada para hacer que el manejador de bases de datos “duerma” determinados segundos.	3
‘select’	Cadena común al explotar SQLi, utilizada para seleccionar determinados campos de las tablas.	23

Tabla 4.2. Cadenas de Agentes de usuario potencialmente maliciosas.

Como resultado de utilizar estas cadenas como filtro, se encontraron 9412 agentes de usuario explícitamente maliciosos de un total de 557038, siendo 215 únicos. Es decir, únicamente un 1.7% del total. Cabe resaltar que este resultado no significa que los demás agentes de usuario no formen parte de peticiones maliciosas, pues este valor es fácilmente modificable por los clientes HTTP. Sin embargo, sirve para obtener un panorama general de la cantidad de peticiones maliciosas y de los tipos de ataques efectuados.

Los 5 agentes de usuario más comunes con estas características fueron los siguientes, demostrando que el lenguaje de programación Python es muy popular entre los que desarrollan scripts para automatizar el envío de peticiones HTTP y, si bien no puede considerarse malicioso sin tener en cuenta el contexto, puede considerarse sospechoso de realizar ataques automatizados o enumeración de información. Lo anterior puede concluirse, por tratarse de un agentes de usuario que hacen referencia explícita al crawling de páginas web y a lenguajes de programación, pues estos demuestran un alto nivel de automatización, muy útil para encontrar y explotar vulnerabilidades y muy difícilmente usados para navegar por sitios de forma convencional:

1. Google Crawler: Googlebot/2.1 (+http://www.google.com/bot.html) (2682 veces)
2. python-requests/2.18.4 (1311 veces)
3. python-requests/2.13.0 (845 veces)
4. python-requests/2.19.1 (621 veces)
5. Python/3.6 aiohttp/3.4.4 (541 veces)

4.7 Automatización del análisis

El objetivo de automatizar el análisis de la actividad maliciosa presente en la red Tor es el de determinar la viabilidad de su estudio para obtener información significativa

sobre los diferentes ataques efectuados en internet, desde ataques genéricos hasta campañas dirigidas. Aunque se ha encontrado información relevante hasta este punto, la extracción de los datos ha sido automatizada y el análisis únicamente ha sido manual.

4.7.1 Análisis para la detección de intrusos: Bro y Suricata

Se decidió utilizar herramientas especializadas en la detección de actividad maliciosa, principalmente se utilizaron los sistemas de detección de intrusos en red (NIDS) Bro y Suricata a partir de un sistema de detección en la nube y el sistema de análisis de malware VirusTotal.

El primer paso fue analizar una de las capturas de tráfico generadas al filtrar todo el tráfico perteneciente a los puertos de los protocolos HTTP y de correo. Es decir, se eliminó el tráfico que contenía los puertos 80, 8080, 8000, 443, 25, 143, 465, 587 y 993 para obtener una mejor idea de la actividad maliciosa realizada en los puertos menos comunes, pero utilizados, en la política de salida. Este análisis se realizó a través de una plataforma en línea llamada *PacketTotal* que se encarga de ayudar en el análisis forense y detección de actividad maliciosa mediante el uso de los sistemas de detección de intrusos Bro y Suricata a archivos en formato *pcap* (PacketTotal, 2019). Una vez procesados los archivos, provee gráficas generadas de forma automática muy útiles para categorizar la actividad en la red.

Utilizando esta plataforma se analizaron dos archivos diferentes, encontrando en ambos el mismo tipo de actividad maliciosa. En general, se puede hablar de 6 alertas generadas: ataques misceláneos, actividad miscelánea, comportamiento no convencional en los protocolos, potencial violación a la privacidad corporativa, tráfico potencialmente malicioso y detección de troyanos. En las imágenes 4.30 y 4.31 se muestran las gráficas que muestran la relación entre el tipo y número de alertas.

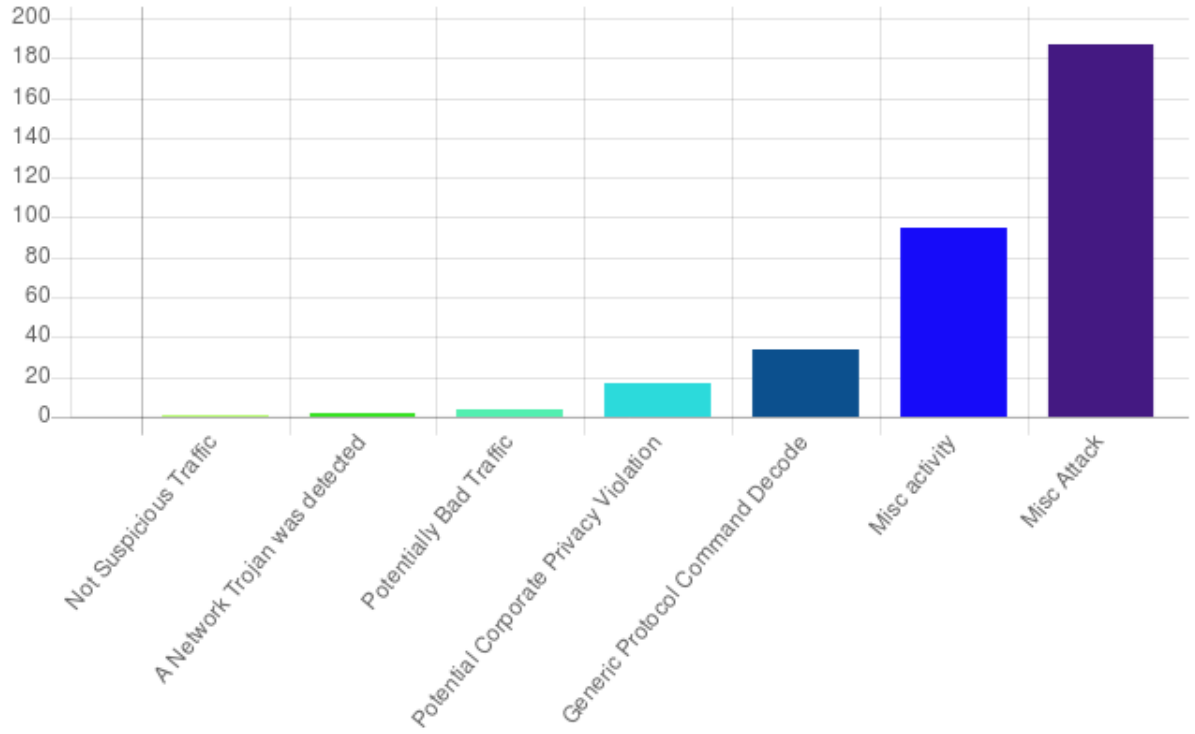

Imagen 4.30. Resultado 1 del análisis de un archivo en packettotal.

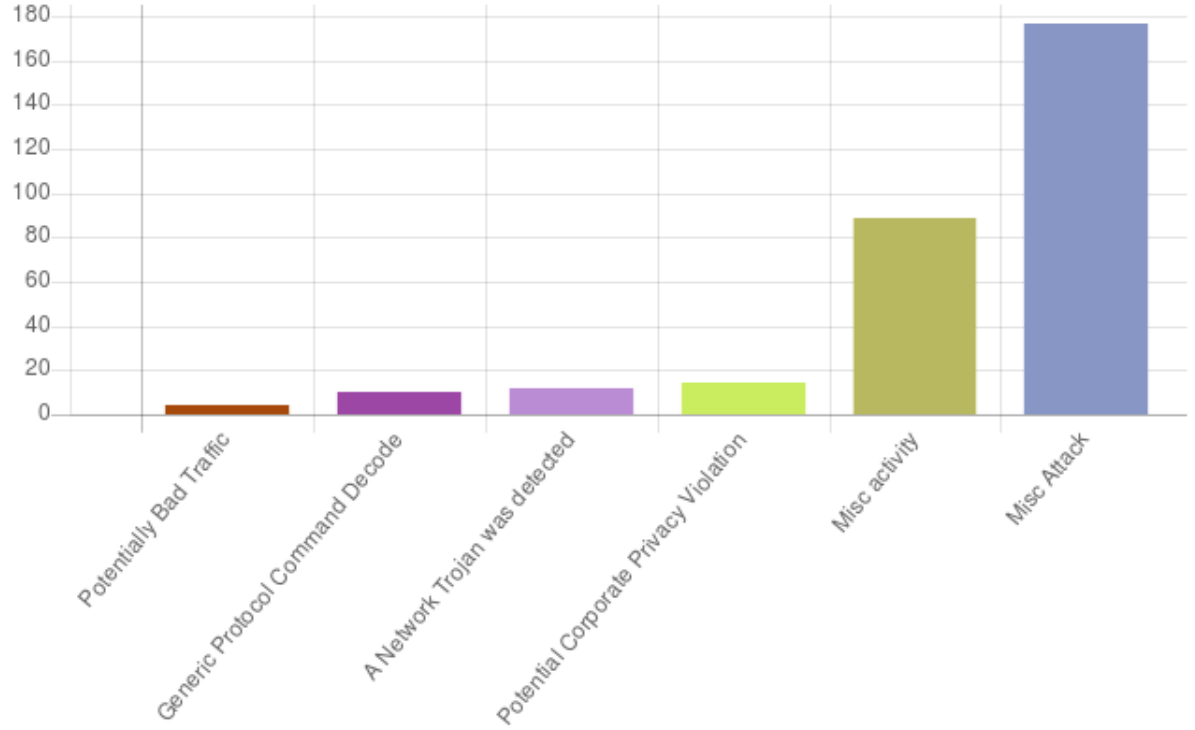

Imagen 4.31. Resultado 2 del análisis de un archivo en packettotal.

Puede observarse que la alerta de ataques misceláneos es la más común teniendo, entre las dos gráficas, un total de 364 entradas. Sin embargo, al revisar los detalles de estas entradas se encontró que la mayoría corresponde a la detección de tráfico de la red Tor. De hecho, únicamente 7 entradas corresponden a otro tipo de actividad

(mala reputación en la dirección IP por envío de spam). La razón por la que se consideraría a este tipo de tráfico como parte de un ataque es que se trata de tráfico no convencional, sin embargo, puesto que este estudio es sobre la red Tor, el tráfico sobre su enrutamiento no se considera malicioso. Lo anterior indica que la alerta principal generada por PacketTotal tiene un 98.07% de falsos positivos, demostrando que la red Tor no es comúnmente utilizada para la explotación de vulnerabilidades en los servidores de la red abierta. En la imagen 4.32 se muestran los detalles de varias alertas con esta clasificación. Además, este mismo comportamiento de clasificar el tráfico como malicioso por tratarse de la red Tor se encuentra en otras categorías de alertas como la de “Potencial Violación a la Privacidad Corporativa”, lo que indica un altísimo porcentaje de falsos positivos en cuanto a actividad maliciosa real.

Alert Description	Alert Signature	Severity	Sender IP	Sender Port	Target IP	Target Port
Misc Attack	ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 80	2	95.130.11.170	37136	192.168.1.2	18666
Misc Attack	ET TOR Known Tor Exit Node Traffic group 80	2	95.130.11.170	37136	192.168.1.2	18666
Misc Attack	ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 468	2	62.210.92.11	9101	192.168.1.2	37741
Misc Attack	ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 187	2	163.172.209.161	9001	192.168.1.2	41313
Misc Attack	ET TOR Known Tor Exit Node Traffic group 15	2	171.25.193.78	57756	192.168.1.2	51413

Imagen 4.32. Detalles de alertas generadas por el tráfico de Tor.

En ambos análisis, la segunda alerta más común es la de actividad miscelánea. En esta clasificación se encuentran principalmente alertas generadas por tráfico IRC (*Internet Relay Chat*) al encontrar comandos de este protocolo como “JOIN” que sirve para unirse a una sala de chat. Debido a que este protocolo de mensajería instantánea es cada vez menos convencional y es comúnmente usado por diversos tipos de *malware* como medio de comunicación es que se considera como tráfico malicioso. Sin embargo, este protocolo no es por sí mismo malicioso y como, en la mayoría de los casos, no se detectó actividad maliciosa al revisar detalladamente las alertas, también se les considera como falsos positivos.

La clasificación más interesante y con mayor relevancia, aunque con menos entradas, es la de detección de troyanos de red a través de indicadores de compromiso como direcciones IP que han sido reportadas como servidores de comando y control

(*Command and Control*) por organizaciones como *Shadowserver*. Aunque la cantidad de alertas generadas por *malware* es bastante pequeña en comparación con las demás categorías, y en general con la cantidad de tráfico capturado, se demuestra que es un buen método para obtener muestras de *malware* y poder analizarlas. En la imagen 4.33 se muestran los detalles de varias alertas con esta clasificación.

Alert Description	Alert Signature	Severity	Sender IP	Sender Port	Target IP	Target Port
A Network Trojan was detected	ET CNC Shadowserver Reported CnC Server IP group 5	1	192.168.1.2	46039	131.188.12.138	6667
A Network Trojan was detected	ET TROJAN W32.Duptwux/Ganelp FTP Username - onthelinux	1	192.168.1.2	44097	209.202.252.54	21
A Network Trojan was detected	ET TROJAN W32.Duptwux/Ganelp FTP Username - onthelinux	1	192.168.1.2	44097	209.202.252.54	21
A Network Trojan was detected	ET POLICY IRC Channel JOIN on non-standard port	1	192.168.1.2	32977	203.142.68.139	3128
A Network Trojan was detected	ET CNC Shadowserver Reported CnC Server IP group 48	1	192.168.1.2	35383	91.243.70.250	6667

Imagen 4.33. Alertas generadas por el tráfico de Tor sobre malware y servidores CNC.

Cabe resaltar que, aunque la mayor cantidad de detecciones son consideradas como falsos positivos debido a la naturaleza de este trabajo, estos sistemas de detección han demostrado ser un buen método para detectar conexiones hacia la red. Esto es de gran utilidad en redes corporativas, pues las conexiones hacia este tipo de redes sí debería considerarse como potencialmente maliciosas ya que los usuarios comunes de estas redes pocas razones tendrían para intentar mantenerse anónimos.

5.7.2 Búsqueda de dominios maliciosos: VirusTotal

Posteriormente, como otro método para detectar posible actividad maliciosa se utilizó la herramienta VirusTotal. Se trata de una aplicación web ampliamente utilizada por analistas de *malware* y especialistas en seguridad informática en general que, en su funcionamiento básico, sirve para detectar si un ejecutable o URL es maliciosa utilizando múltiples sistemas antivirus y de detección. VirusTotal tiene una API que permite interactuar con la aplicación a través de diversas maneras que, en su versión gratuita, otorga un máximo de 4 peticiones por minuto. Puesto que VirusTotal tiene bases de datos con información relacionada con las diferentes muestras de *malware*

que recibe diariamente, es posible realizar peticiones a través de la API con nombres de dominio a analizar, posteriormente la herramienta revisa si existe alguna URL que incluya el nombre de dominio indicado que haya sido relacionada con algún *malware* (VirusTotal, 2019). Es decir, detecta si el nombre de dominio forma parte de alguna URL potencialmente maliciosa por haber sido parte de las comunicaciones de alguna muestra de *malware*. En la imagen 4.34 se muestra la URL de un nombre de dominio extraído del tráfico de red detectada como maliciosa por varios de los motores de VirusTotal por tratarse de un sitio de *phishing*.

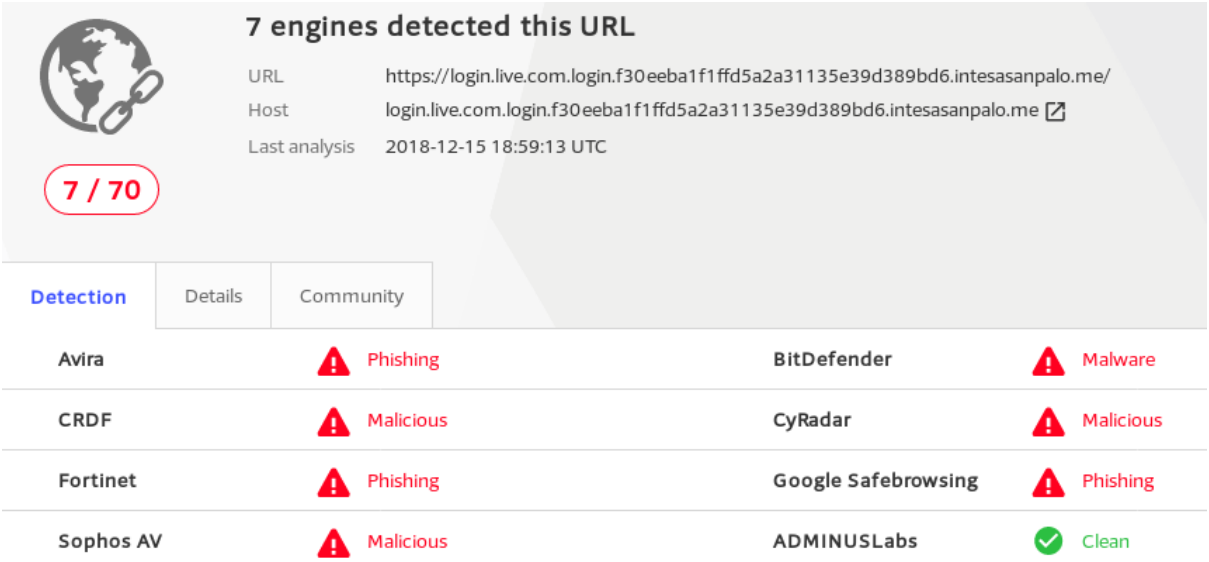

Imagen 4.34. Detección de una URL maliciosa por varios motores de VirusTotal.

Aprovechando esta funcionalidad de la aplicación, se desarrolló un programa en Python que, considerando y respetando el límite de peticiones impuesto por la API gratuita, envía peticiones HTTP con los nombres de dominio extraídos del tráfico de red solicitando información relevante sobre URLs potencialmente maliciosas. Esta herramienta almacena los resultados de estas peticiones en forma de un diccionario de Python serializado en un archivo regular. De esta forma es capaz de almacenar de forma eficaz los resultados y realizar búsquedas rápidas en la estructura de datos, sin necesidad de utilizar un manejador de bases de datos. Puesto que se trata de una herramienta que requiere estar constantemente en ejecución por la cantidad de dominios extraídos, se configuró en forma de servicio a través de un archivo *service* de *systemd* de Linux en el servidor previamente rentado. Es decir, se aprovechó que este equipo está constantemente prendido para poder realizar de forma efectiva todas las peticiones HTTP necesarias a VirusTotal. En la imagen 4.35 puede verse la ejecución del programa haciendo uso de la bandera para reportar resultados, así como el tiempo que ha estado en ejecución el programa.

```
vcastro@km102-03:~/malicious$ python find_malicious_domains.py –pkl-file
domain_db.pkl –report > report.txt
vcastro@km102-03:~/malicious$ head report.txt
186356 domains loaded
186356 domains submitted
19187 domains have at least one malicious URL detected:
 -   http://12uo6321knrzxj5ecc2b8avsh8.org/
 -   https://login.live.com.login.f30eebalflffd5a2a31135e39d389bd6.intesas
     anpalo.me/
 -   http://soudant.portiomollis.beigetre.pasteboa.slumwise.sullely.foicuc
     od.ro/
```

Imagen 4.35. Salida del programa donde se muestra el total de dominios relacionados con actividad maliciosa.

Los resultados anteriores muestran que en total se analizaron los 186356 dominios. Obteniendo que un 10.29% de los dominios forman parte de al menos una URL potencialmente maliciosa. Lo anterior demuestra que, aunque se trata de miles de URLs detectadas por VirusTotal como maliciosas, un altísimo porcentaje (cercano al 90%) de los nombres de dominio no han sido catalogados como tal.

Esto demuestra que el uso principal de la red Tor está lejos de ser malicioso, aunque sea un método común para que diferentes tipos de malware se comuniquen. Debido a esta característica de contener actividad maliciosa, es que el análisis de un nodo de la red puede ser una gran oportunidad para detectar campañas de ataques o, incluso, ataques dirigidos a organizaciones. En la imagen 4.36 se muestra una gráfica con los resultados obtenidos a partir de VirusTotal.

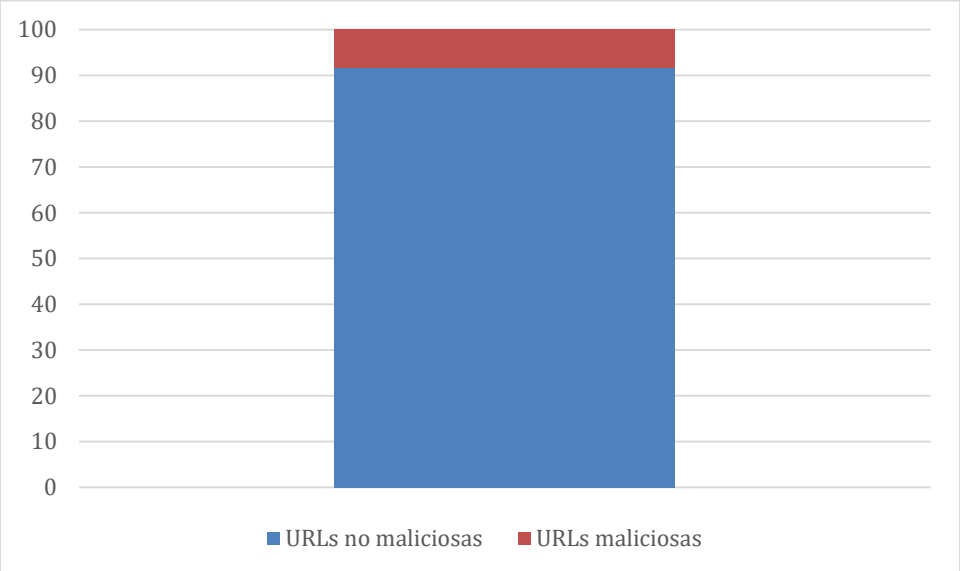

Imagen 4.36. Total de URLs relacionadas y no relacionadas con actividad maliciosa.

Capítulo 5 Seguridad en la red Tor

El objetivo de este capítulo es demostrar qué tan seguros se encuentran los datos de las personas que utilizan la red Tor, en específico en los nodos de salida; y por qué es importante seguir las recomendaciones de seguridad realizadas por el proyecto Tor.

Esto se pretende lograr, principalmente, mediante pruebas de concepto. Se escogió utilizar las técnicas mostradas durante este capítulo, pues son lo suficientemente simples como para ser llevadas a cabo por cualquier administrador de un nodo de salida. Es decir, se pretende dar una visión real de lo que podría estar sucediendo en las comunicaciones de la red Tor.

5.1 Análisis de la confidencialidad

La confidencialidad es la cualidad de la información de ser accedida únicamente por aquellos con autorización de hacerlo. Es decir, su acceso debe estar restringido para cualquiera que no cumpla esta condición. No obstante, la confidencialidad de la información puede verse vulnerada de múltiples formas. Por ejemplo, al escribir credenciales de acceso a un sistema en una computadora personal, estas podrían ser capturadas por un *keylogger* instalado en el equipo o por una persona que realiza un ataque conocido como *shoulder* surfing, es decir, que las observa al momento de ser escritas. De este mismo modo, la confidencialidad de la información puede verse afectada en las redes de computadoras, ya sea por ataques realizados a las mismas, por problemas de configuración o, incluso, por la naturaleza de estas. En este sentido, se analizará el nivel de confidencialidad de la información que viaja mediante la red Tor a través de la búsqueda de información sensible.

5.1.1 Extracción de información sensible: *ngrep* y *tcpdump*

Esta prueba de concepto tiene como fin demostrar la viabilidad de capturar información sensible, como credenciales de autenticación, en un nodo de salida. Para buscar información sensible se considerará únicamente el protocolo de capa de aplicación no cifrado HTTP debido a que se trata del protocolo más usado después de HTTPS y los mensajes de ambos protocolos tienen la misma estructura. Esta búsqueda puede realizarse con herramientas que capturan y filtran los paquetes en una interfaz de red.

Seleccioné *ngrep* para la prueba de concepto ya que es una herramienta que facilita el filtrado de paquetes considerando cadenas de caracteres y expresiones regulares, teniendo una funcionalidad muy similar al popular programa de los sistemas operativos tipo Unix *grep*. Adicionalmente, se utilizó una aplicación web de prueba llamada *Altoro Mutual*, la cual es una aplicación web falsa que simula ser un sitio

bancario real, fue publicada por IBM y sirve para realizar pruebas de concepto sobre vulnerabilidades en sitios web. Gracias a que no ofrece ninguna funcionalidad real, se utilizó para esta prueba de concepto.

Para evitar vulnerar la confidencialidad de datos de clientes reales, decidí utilizar un nombre de usuario y una contraseña de prueba. Decidí utilizar datos poco comunes y lo suficientemente complejos para evitar tener más de una coincidencia durante su captura, es decir, para capturar únicamente mis datos de prueba. Para realizar la prueba de concepto, configuré un cliente Tor forzándolo a utilizar únicamente mi nodo de salida en sus circuitos. De esta forma, garantizo que puedo capturar los datos de prueba en el nodo de salida. Estos datos de prueba se muestran en la tabla 5.1 y se muestra su uso en la imagen 5.1.

Campo	Variable HTTP	Valor de prueba
Nombre de usuario	uid	pruebaTesis
Contraseña	passw	P4sswordMuySeguroPorSuLongitudYCaracteres.1029384756

Tabla 5.1. Valores utilizados en la prueba de concepto.

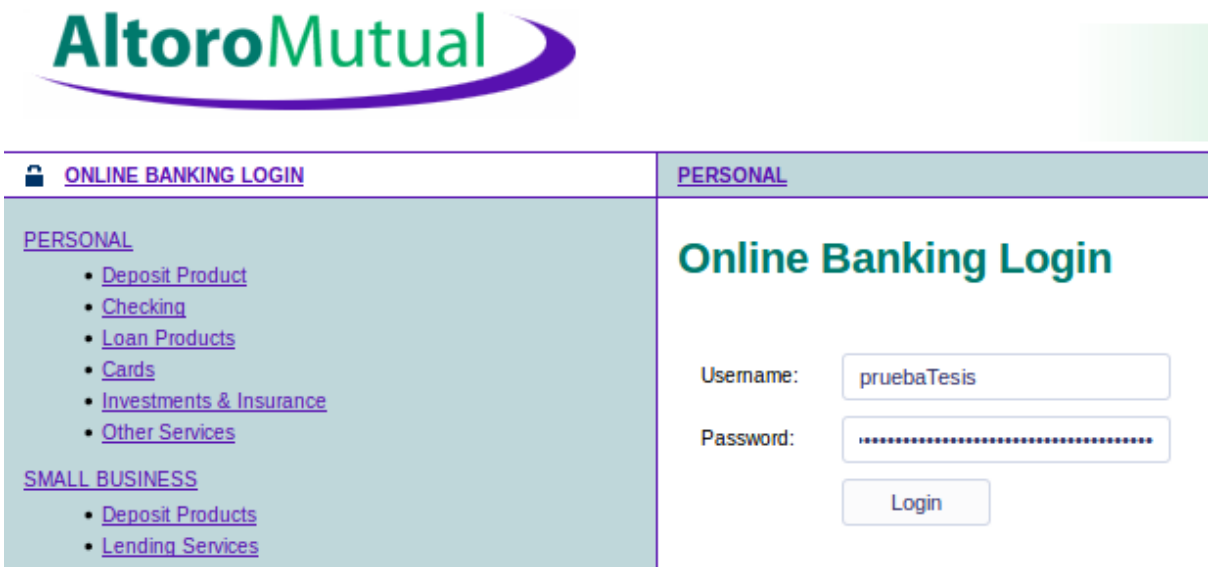

Imagen 5.1. Datos de autenticación falsos en la página Altoro Mutual.

En el nodo de salida, ejecuté el comando mostrado en la imagen 5.2, el cual utiliza como filtro la cadena correspondiente a la contraseña de prueba, imprimiendo en la salida estándar la única petición HTTP que coincide, al mismo tiempo que la escribe en el archivo *poc.pcap*. Cabe resaltar que la única petición que coincide con el filtro es la correspondiente al aplicativo *Altoro Mutual*, por lo que corresponde a los datos de prueba, resultando exitosa la prueba de concepto.

Cabe destacar que la contraseña de prueba utilizada consta de 8 letras mayúsculas, 32 letras minúsculas, 11 dígitos y 1 carácter especial; dado el número de caracteres que la conforman podría considerarse una contraseña muy robusta en términos de complejidad. Sin embargo, esta cualidad podría ofrecer únicamente una falsa sensación de seguridad, pues pierde todo su valor si es usada en un canal no seguro, como se realizó en la prueba.

```
root@km14102-03:~# ngrep P4sswordMuySeguroPorSuLongitudYCaracteres.1029384756 -q -O poc.pcap
interface: ens18 (                              )
match: P4sswordMuySeguroPorSuLongitudYCaracteres.1029384756
output: poc.pcap


T                :36062 -> 65.61.137.117:8080 [AP]
  POST /doLogin HTTP/1.1..Host: www.altoromutual.com:8080..User-Agent: Mozilla/5.0 (Windows
  NT 6.1; rv:60.0) Gecko/20100101 Firefox/60.0..Accept: text/html,application/xhtml+xml,appl
  ication/xml;q=0.9,*/*;q=0.8..Accept-Language: en-GB,en;q=0.5..Accept-Encoding: gzip, defla
  te..Referer: http://www.altoromutual.com:8080/login.jsp..Content-Type: application/x-www-f
  orm-urlencoded..Content-Length: 90..Cookie: JSESSIONID=8522B280B232C1023D69A4E1D06254FC..C
  onnection: keep-alive..Upgrade-Insecure-Requests: 1....uid=pruebaTesis&passw=P4sswordMuySe
  guroPorSuLongitudYCaracteres.1029384756&btnSubmit=Login
```

Imagen 5.2. Credenciales de acceso de prueba capturadas en nodo de salida.

Utilizando el archivo *poc.pcap* generado con *ngrep*, se realizó y probó un filtro que sirve para capturar peticiones POST con la herramienta *tcpdump*, pues esta no sirve únicamente para capturar tráfico de red, sino también para filtrarlo considerando una amplia gama de condiciones. Estas condiciones pueden ser muy generales, filtrando por protocolos de interés, ya sea de transporte (TCP o UDP) o de red (IP, ICMP, etcétera), así como direcciones IP o puertos en específico. Sin embargo, los filtros escritos en esta herramienta pueden ser mucho más específicos al considerar el contenido de los diferentes campos que forman las cabeceras de estos protocolos.

Gracias a la encapsulación de protocolos del modelo TCP/IP y a que este tipo de filtros que se pueden aplicar a las cabeceras de los protocolos de red y de transporte es posible identificar paquetes interesantes a nivel de capa de aplicación. Por ejemplo, en el protocolo HTTP lo más común es enviar las peticiones con credenciales de autenticación utilizando el método POST para que estas no se muestren en la URL y, por lo tanto, en la barra de búsquedas de los navegadores. Aprovechando el uso que se le da al método POST y que la definición del protocolo HTTP indica que el método a usar ocupa el primer campo de una petición, se puede escribir un filtro como el mostrado en la imagen 5.3.

```
root@km14102-03:~# tcpdump -r poc.pcap -A -vv \
> 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504f5354'
reading from file poc.pcap, link-type EN10MB (Ethernet)
```

Imagen 5.3 Filtro en tcpdump para obtener las peticiones HTTP POST.

La explicación de las secciones de este comando se muestra a continuación:

Secciones de comando	Explicación
-r <nombre_archivo>	Sirve para indicar qué archivo a leer.
-A	Imprime cada paquete en formato ASCII.
-vv	Ejecución verbosa del comando.
tcp[12:1]	Obtiene el decimosegundo octeto del encabezado TCP, el cual en su primer nibble tiene la longitud del encabezado.
tcp[12:1] & 0xf0	Sirve para mantener los primeros cuatro bits del octeto seleccionado, mientras que los últimos cuatro los convierte en cero. con esto se obtiene el número de palabras (32 bits) que forman al encabezado
(tcp[12:1] & 0xf0 >> 2)	Realiza un corrimiento de dos bits al resultado anterior para obtener el número de octetos que forman al encabezado. Es decir, multiplica el resultado anterior por cuatro.
tcp[(tcp[12:1] & 0xf0 >> 2):4]	A partir del octeto resultante, lee los siguientes 4 bytes, es decir, los primeros cuatro bytes correspondientes a la capa de aplicación encapsulada en TCP.
tcp[(tcp[12:1] & 0xf0 >> 2):4] = 0x504f5354	Compara los primeros cuatro bytes de la capa de aplicación con los valores hexadecimales ‘0x504f5354’ o ‘POST’ en ASCII. Es decir, muestra únicamente aquellos paquetes que tengan en los primeros cuatro bytes de la capa de aplicación la cadena ‘POST’.

Como resultado de la ejecución de este comando se obtiene, en formato ASCII, todos aquellos paquetes que tienen la cadena ‘POST’ en los primeros cuatro bytes de la capa de aplicación. Es decir, se muestran peticiones HTTP que probablemente contienen información sensible en sus variables, especialmente usuarios y contraseñas. En la imagen 5.4 se muestra una petición POST obtenida con el filtro mencionado, donde se puede ver el método usado, el recurso solicitado, la versión de HTTP, los encabezados de la petición y el cuerpo de la misma donde se envían las credenciales de autenticación de prueba utilizadas en *Altoro Mutual*.

```
POST /doLogin HTTP/1.1
Host: www.altoromutual.com:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.altoromutual.com:8080/login.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Cookie: JSESSIONID=8522B280B232C1023D69A4E1D06254FC
Connection: keep-alive
Upgrade-Insecure-Requests: 1

uid=pruebaTesis&passw=P4sswordMuySeguroPorSuLongitudYCaracteres.1029384756&btnSubmit=Login
```

Imagen 5.4. Petición HTTP POST con credenciales para autenticarse en aplicación web.

Para obtener un conteo del número de peticiones HTTP que utilizan el método POST, se ejecutó el comando mostrado en la imagen 5.5 sobre las capturas del tráfico del nodo de salida. Posteriormente, con ayuda de los programas *grep, sort y wc* se realizó un conteo de las peticiones POST, eliminando las líneas duplicadas para evitar contaminar el conteo con ataques de fuerza bruta. Resultado de este análisis, se encontraron 9393 peticiones en el lapso de una hora y, aunque no necesariamente todas transportan credenciales, no se debe olvidar que una petición POST puede contener otros datos sensibles como cookies de sesión que podrían servir para atacar a un usuario de la red.

```
root@km14102-03:/opt/traffic# tcpdump -r clean_dump_2018-10-25_23\:00\:01.pcap -A -q \
> 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504f5354' \
> | grep 'POST /' | sort -u | wc -l
reading from file clean_dump_2018-10-25_23:00:01.pcap, link-type EN10MB (Ethernet)


9393
```

Imagen 5.5. Conteo de peticiones HTTP POST encontradas en una captura de tráfico.

En la gráfica mostrada en la imagen 5.6 se recopila la cantidad de peticiones POST encontradas por cada hora en que se capturó tráfico de red.

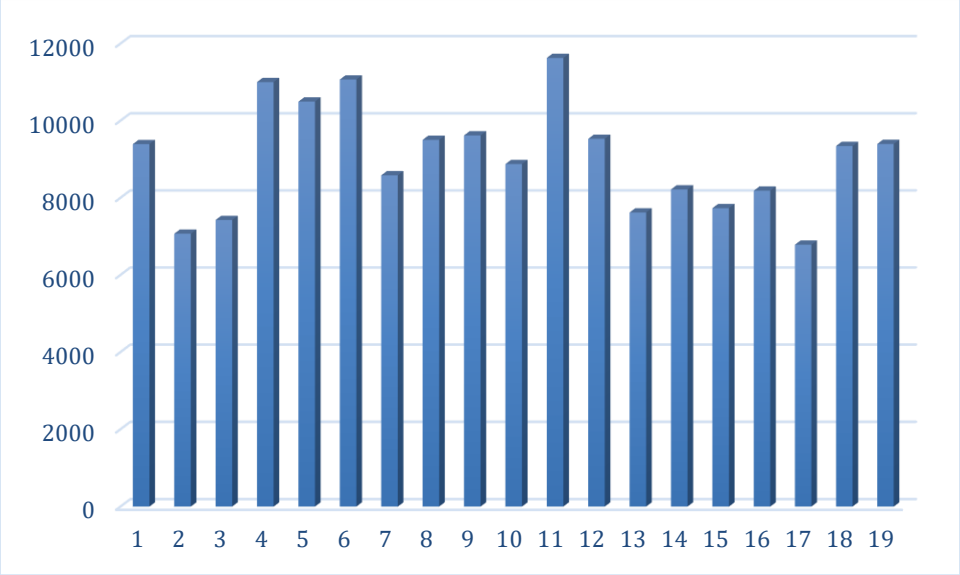

Imagen 5.6. Número de peticiones POST por cada hora de captura.

Estos resultados arrojan un total de 171474 peticiones POST únicas y un promedio de 9025 peticiones POST por hora vulnerables a ser analizadas y aprovechadas por quien administre un nodo de salida en la red Tor.

Cabe destacar que, con el análisis realizado para determinar la viabilidad de encontrar información sensible, es posible vulnerar también el anonimato provisto por la red Tor. Esto es particularmente cierto cuando los datos obtenidos se refieren a credenciales para iniciar sesión en diferentes servicios, ya que al entregar esos datos se está autenticando ante el sitio, es decir, se le está indicando quién es quien intenta acceder a él, perdiendo así el anonimato ante el sitio web y ante quien espíe las comunicaciones. Cuando usar protocolos seguros no es una opción y para evitar que un administrador de un nodo de salida conozca la identidad de sus usuarios al espiar el tráfico, estos pueden evitar el uso de datos que los puedan identificar de forma unívoca. Por ejemplo, un usuario puede utilizar un seudónimo que no pueda ser relacionado directamente con esa persona.

5.2 Análisis de la integridad

La integridad de la información es la cualidad de la información de ser modificada únicamente por aquellos que tiene autorización para hacerlo. En ese sentido, un ataque a la integridad es cuando esta se ve modificada durante su transporte o almacenamiento, sin que esta modificación sea autorizada por los dueños de la información. Con el objetivo de determinar la viabilidad de que un ataque de este tipo sea cometido a la información que viaja sobre la red Tor, se realizó una prueba de concepto en la que se ataca el protocolo DNS a través de la modificación de las

peticiones DNS para que estas sean resueltas por un servidor controlado por el atacante.

5.2.1 Infraestructura de red necesaria para la prueba de concepto

En esta prueba de concepto en específico, la víctima del ataque es el cliente conectado a la red Tor y el equipo malicioso es el nodo de salida usado por el cliente. Como fin último, la prueba de concepto pretende demostrar la viabilidad de ataques de tipo *pharming* en esta red, recordando que un ataque *pharming* es un tipo especial de ataque *phishing* en el que se redirige a la víctima a servidores maliciosos al hacer peticiones DNS a servidores comprometidos o controlados.

En la imagen 5.11 se muestra una situación común en la que un cliente entra a un sitio de la red abierta a través de Tor. En primer lugar, el cliente realiza una petición HTTP al sitio que visitará (sitio1.com). En segundo lugar, esta petición es redirigida a través de un nodo de entrada y uno intermedio hacia el nodo de salida. En tercer lugar, el nodo de salida, al no conocer la dirección IP del dominio que se va a visitar, realiza la consulta DNS a un servidor configurado, en este caso, a un servidor DNS público de Google. En cuarto lugar, el servidor de Google responde al nodo de salida la dirección IP (x.x.x.x) del dominio consultado. Finalmente, el nodo de salida realiza la petición GET al dominio original “sitio1.com”.

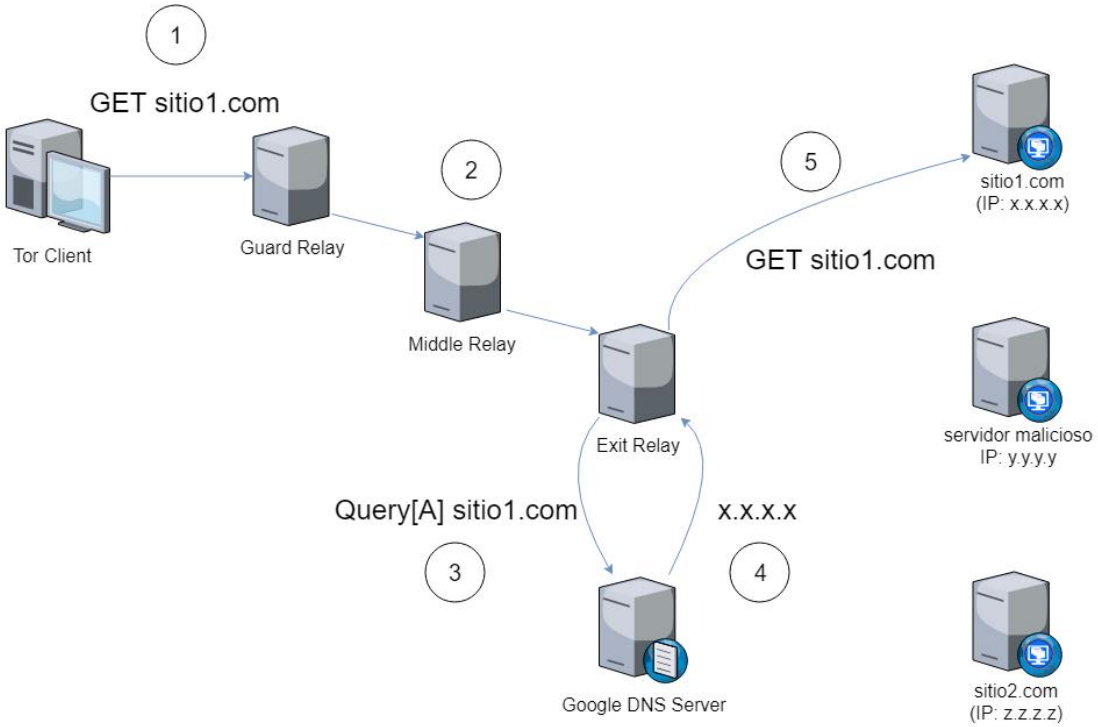

Imagen 5.7. Infraestructura de red para acceder a un sitio de forma común.

En la imagen 5.12 se muestra una situación en la que el nodo de salida controla la respuesta a las peticiones DNS. En primer lugar, el cliente hace una petición HTTP al

sitio que visitará (sitio2.com). En segundo lugar, esta petición es redirigida a través de un nodo de entrada y un nodo intermedio al nodo de salida. En tercer lugar, el nodo de salida envía la petición a un servidor DNS malicioso, en esta prueba este servidor es el mismo nodo de salida. En cuarto lugar, el servidor DNS indica que la dirección IP es la y.y.y.y correspondiente a un servidor malicioso, en lugar de z.z.z.z que corresponde al sitio legítimo. Finalmente, el nodo de salida hace la petición HTTP al servidor malicioso a través del dominio legítimo.

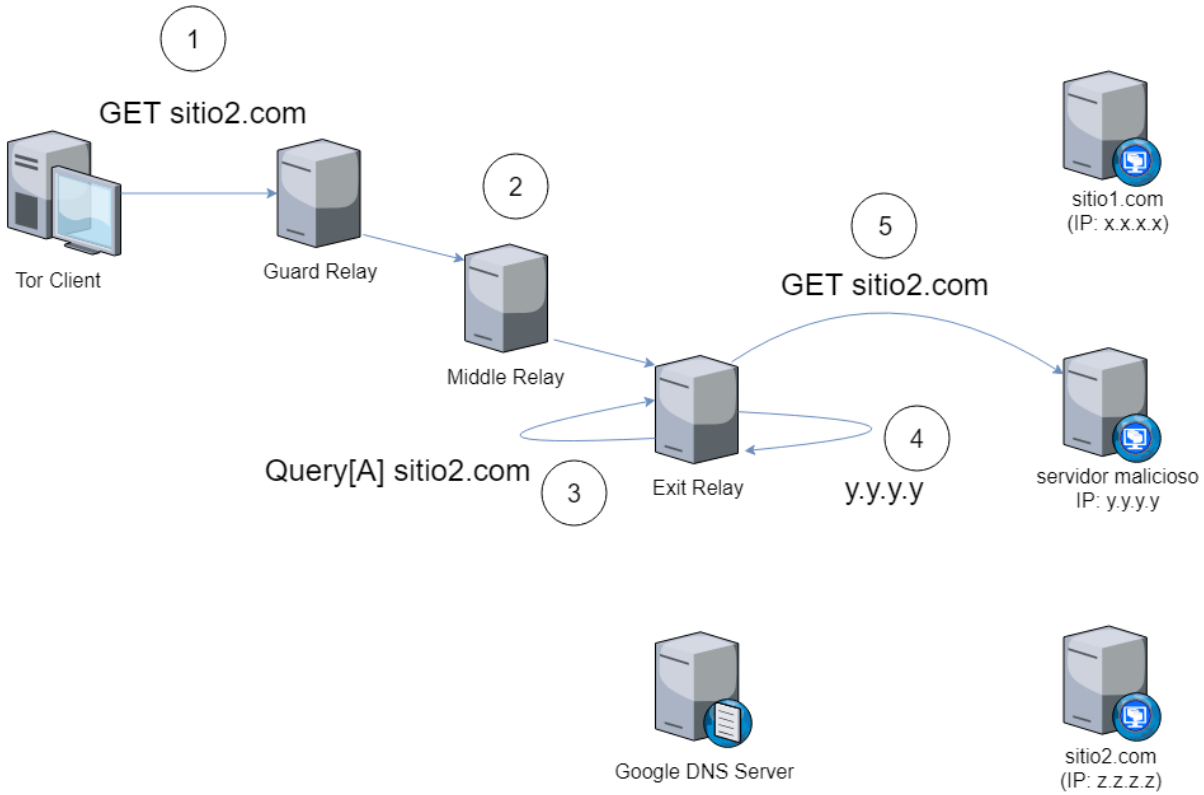

Imagen 5.8. Infraestructura de red modificada por atacante para acceder a recursos en internet.

5.2.2 Configuración de la infraestructura de red

Para esta prueba de concepto se utilizó un nombre de dominio inexistente para, de esta forma, evitar dañar a algún usuario de la red que se encuentre utilizando el nodo de salida configurado para este trabajo. El dominio utilizado es “pruebaintegridaddns.com” y en la imagen 5.13 puede observarse en la página de la ICANN (Corporación de Internet para la Asignación de Nombres y Números por sus siglas en inglés) que dicho dominio no ha sido encontrado.

Imagen 5.9. Validación de la inexistencia del dominio de prueba con WHOIS.

De igual forma en la imagen 5.14 se muestra una petición DNS usando *nslookup* al servidor público de Google con la dirección IP 8.8.8.8 preguntando por el dominio "pruebaintegridaddns.com". Puesto que se trata de un dominio inexistente, no es posible obtener una dirección IP asociada al nombre.

```
vcastro@km14102-03:~$ nslookup
> server 8.8.8.8
Default server: 8.8.8.8
Address: 8.8.8.8#53
> pruebaintegridaddns.com
Server:         8.8.8.8
Address:        8.8.8.8#53

** server can't find pruebaintegridaddns.com: NXDOMAIN
>
```

Imagen 5.10. Validación de la inexistencia del dominio de prueba mediante una resolución DNS.

Una vez que se validó que el dominio no existe y está disponible para realizar pruebas sin afectar a nadie, se configuró la topología explicada previamente en la imagen 5.12. El primer paso se muestra en la imagen 5.15 y se trata de agregar una entrada a un archivo *hosts* que será utilizado para traducir nombres de dominio a direcciones IP. En este caso, se asignó el nombre de dominio a una dirección IP que pertenece a *Altoro Mutual*, la misma página de pruebas utilizada durante la prueba a la confidencialidad.

```
vcastro@km14102-03:/etc$ cat /opt/dnsmasq/hosts
65.61.137.117 pruebaintegridaddns.com
```

Imagen 5.11. Contenido del archivo hosts para resolver dominio de prueba.

Posteriormente, como se muestra en la imagen 5.16, se ejecuta el programa *dnsmasq* con las siguientes opciones para levantar el servicio de DNS:

Sección de comando	**Explicación**
-C /opt/dnsmasq/dnsmasq.conf	Archivo de configuración que utiliza el programa dnsmasq. En este archivo se especifica: - Interfaz de red donde brinda el servicio (interfaz local) - Servidor DNS al que reenviará las peticiones en caso de no tener la respuesta (servidor público de Google) - Que mantenga un registro de las consultas recibidas - Archivo donde se registran las consultas recibidas
--no-hosts	Se indica que no lea el contenido del archivo hosts por defecto del sistema operativo.
-H /opt/dnsmasq/hosts	Se indica de forma explícita el archivo hosts creado previamente.
--no-resolv	Se indica que no lea el contenido del archivo resolv.conf del sistema operativo, sirve para que tome como servidor DNS el indicado en el archivo de configuración.
-d	Se indica que se ejecute en modo demonio o servicio.

```
root@km14102-03:~# dnsmasq -C /opt/dnsmasq/dnsmasq.conf \
> --no-hosts -H /opt/dnsmasq/hosts \
> --no-resolv -d | \
> tee /opt/dnsmasq/dnsmasq.log
dnsmasq: started, version 2.76 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: read /opt/dnsmasq/hosts - 1 addresses
```

Imagen 5.12. Ejecución del programa dnsmasq.

Asimismo, se debe modificar el archivo */etc/resolv.conf* para que los programas instalados en el equipo utilicen el servicio DNS instalado localmente por defecto, como se muestra en la imagen 5.17. En este caso, el programa del cual se tiene interés que lo use es tor.

```
vcastro@km14102-03:/etc$ cat /etc/resolv.conf
nameserver 127.0.0.1
```

Imagen 5.13. Configuración del servidor DNS en la interfaz local.

Una vez instalado y configurado el servicio DNS, se configuró el nodo para funcionar como de salida. Asimismo, se configuró un cliente de prueba para que utilice el nodo de salida en todos sus circuitos forzosamente. De esta manera, se garantiza que, al solicitar la página en este cliente, la petición llegará hasta el nodo de salida configurado para la prueba. Como se muestra en la imagen 5.18, se configura el archivo *torrc* del cliente, agregando las directivas *ExitNodes* y *StrictNodes*.

```
root@TraffAnalysis:~# head -n5 /etc/tor/torrc
ControlPort 9051
CookieAuthentication 1

ExitNodes [redacted]
StrictNodes  1
```

Imagen 5.14. Configuración del cliente para utilizar preferentemente el nodo de salida.

Al iniciar el nodo como salida, se puede ver en la bitácora de dnsmasq peticiones de nombres de dominio inexistentes y otros muy comunes. Esto se debe a que, para aceptar al nodo como salida, este debe pasar pruebas para verificar que no se está suplantando la identidad de ningún dominio. Un ejemplo de las peticiones realizadas al servicio configurado se muestra en la imagen 5.19, en esta misma imagen se observa que las peticiones a las cuales no se tiene una respuesta configurada son reenviadas a la dirección IP 8.8.8.8, es decir, al servidor DNS público de Google.

```
reply 7xkE7i2Z2.nET is NXDOMAIN
reply T3qBjUqCm.CoM is NXDOMAIN
reply BUW4DjUs.COM is NXDOMAIN
reply k76OYEUQqUoDPqQ.cOM is NXDOMAIN
reply AuIqu2i5.Net is NXDOMAIN
reply CTvGlaYOTIqE57A.nEt is NXDOMAIN
reply pSy25qAz.Net is NXDOMAIN
reply iTDpmksMz.ORG is NXDOMAIN
reply Fjf5PD6gxSFNIAgY.OrG is NXDOMAIN
reply ZYRNAN6UpJX is NXDOMAIN
reply L6OYiqzaUkMa6.OrG is NXDOMAIN
reply ohczn56tizNuy.ORg is NXDOMAIN
query[A] WWW.gooGLe.CoM from 127.0.0.1
forwarded WWW.gooGLe.CoM to 8.8.8.8
query[AAAA] WWw.gOOGLe.cOM from 127.0.0.1
forwarded WWw.gOOGLe.cOM to 8.8.8.8
query[A] WWw.mIT.EDU from 127.0.0.1
forwarded WWw.mIT.EDU to 8.8.8.8
```

Imagen 5.15. Extracto de dominios consultados en el servidor DNS.

Como aclaración, cabe resaltar que las peticiones mostradas en las bitácoras del servidor DNS configurado aparecen en combinaciones de mayúsculas y minúsculas debido a que este es uno de los mecanismos recomendados para agregar entropía a las peticiones y, de esta manera, evitar ciertos tipos de ataques de envenenamiento de caché a los servidores DNS. (Google Public DNS, 2019)

5.2.3 Ejecución y resultados de la prueba de concepto

Para validar que la prueba de concepto es exitosa se esperó hasta que el nodo fuera aceptado como salida y, posteriormente, se accedió al dominio "pruebaintegridaddns.com" en el cliente configurado previamente. Al acceder a este dominio, automáticamente se muestra la página de pruebas de *AltoroMutual*, como se muestra en la imagen 5.20. De esta forma, se demuestra que la prueba de suplantación de identidad fue realizada exitosamente, pues mediante un nombre de dominio inexistente se redirigió a una página de pruebas.

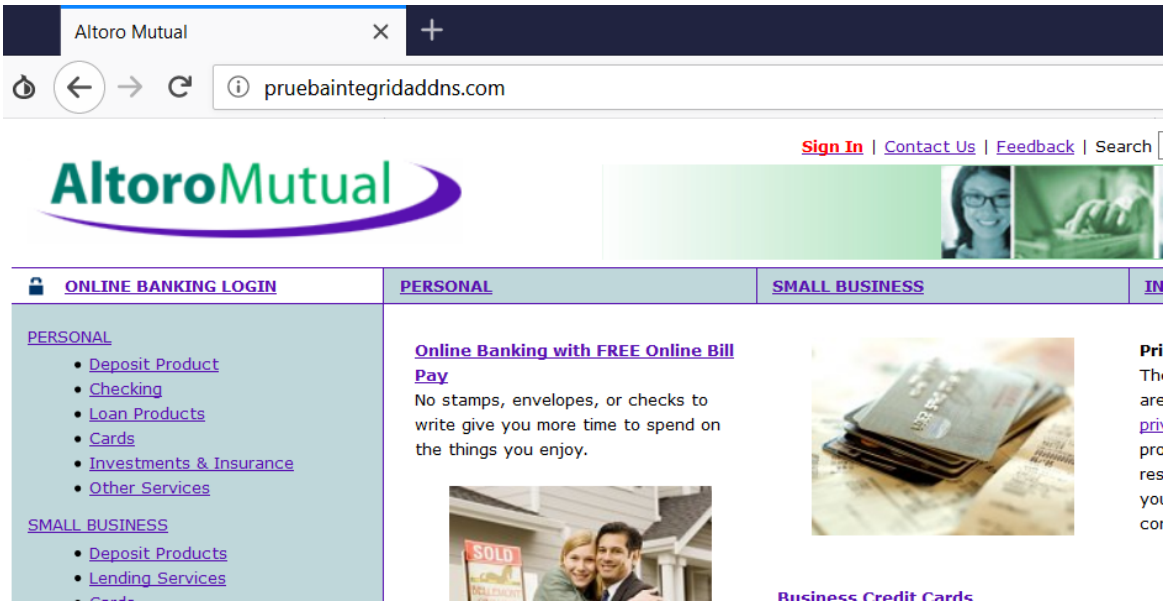

Imagen 5.16. Ejecución y validación de la prueba de concepto.

Al revisar la petición en la bitácora, se observa que esta no es reenviada al servidor DNS de Google, sino que se entrega una respuesta autoritativa inmediatamente por parte del servidor DNS malicioso. Por otro lado, el dominio "www.ingenieria.unam.mx" no está siendo suplantado, por lo que esa petición sí es reenviada al servidor DNS público de Google. En la imagen 5.21 se muestran estos registros en la bitácora de *dnsmasq*.

```
root@km14102-03:~# date
Mon May  6 12:38:39 CEST 2019
root@km14102-03:~#
root@km14102-03:~# grep -i pruebaintegridad /opt/dnsmasq/dnsmasq.log
May  6 12:35:07 dnsmasq[19964]: query[A] PruEBaiNtegRIdaDdns.com from 127.0.0.1
May  6 12:35:07 dnsmasq[19964]: /opt/dnsmasq/hosts PruEBaiNtegRIdaDdns.com is 65.61.137.117
root@km14102-03:~#
root@km14102-03:~# grep -i www.ingenieria.unam.mx /opt/dnsmasq/dnsmasq.log
May  6 12:36:54 dnsmasq[19964]: query[A] WwW.INGeNIErIA.UNam.mX from 127.0.0.1
May  6 12:36:54 dnsmasq[19964]: forwarded WwW.INGeNIErIA.UNam.mX to 8.8.8.8
May  6 12:36:54 dnsmasq[19964]: reply WwW.INGeNIErIA.UNam.mX is <CNAME>
```

Imagen 5.17. Registro del dominio de prueba en la bitácora del servicio DNS.

Como resultado de la prueba de concepto, se tiene que es posible desde un nodo de salida redirigir a los usuarios de la red Tor a páginas diferentes a las que planean acceder mediante la manipulación de los servidores DNS que este utiliza. En este sentido, a pesar de que la vulnerabilidad se encuentra en el protocolo DNS, realizar el ataque fue posible nuevamente debido al uso de protocolos no seguros (HTTP) en lugar de sus versiones seguras (HTTPS). Lo anterior se debe a que el protocolo HTTPS no solo sirve para cifrar el tráfico entre el cliente y el servidor, sino para autenticar a los sitios web mediante el certificado. Si se utiliza HTTPS para acceder a los diferentes sitios, los navegadores avisarían a los usuarios que el sitio al que se intenta acceder tiene un nombre de dominio que no concuerda con el certificado que tiene configurado, lo que haría aún más difícil que las personas cayeran en la trampa, evitando así entregar sus credenciales.

Cabe señalar que, aunque usar las versiones seguras de los protocolos ayuda en gran medida a mejorar la confidencialidad e integridad de la información, también existen ataques sobre estos protocolos. Uno de estos ataques se conoce como *SSL Stripping*. Este tipo de ataques aprovecha una posición de hombre en medio (*MitM*) entre un cliente y un servidor y evita que el cliente establezca una conexión directa con el servidor mediante HTTPS, obligándolo a utilizar HTTP. Este tipo de ataques también se conocen como ataques de degradación, pues obligan al cliente a utilizar un modo de operación menos seguro (HTTP) del que podría utilizar (HTTPS) (Venafi, 2018). A pesar de que la red Tor por su naturaleza no puede evitar este tipo de ataques, se recomienda ampliamente el uso del navegador Tor, que viene configurado por defecto con medidas que ayudan a mitigarlos, como es el caso de la extensión *HTTPS Everywhere*, que obliga al cliente a utilizar HTTPS si este protocolo se encuentra disponible.

5.3 Análisis de la disponibilidad

La disponibilidad en la red Tor depende de varios factores, principalmente de la cantidad de nodos existentes en la red y de las conexiones que puedan soportar estos nodos. Durante el análisis a la disponibilidad de esta red se considera principalmente el comportamiento de esos dos factores, de tal modo que se obtenga una idea más clara sobre las capacidades de la red.

5.3.1 Análisis de la disponibilidad de la red basado en el número de nodos

Utilizando el sistema de búsqueda de nodos del proyecto Tor, es posible conocer la cantidad de nodos que han existido en la red durante intervalos de tiempo determinados, es decir, el tamaño de la red en fechas en específico. Gracias a esta funcionalidad provista por el propio proyecto, es posible realizar un análisis desde el punto de vista de la red y determinar si su capacidad de mantener las conexiones disponibles ha aumentado, disminuido o se ha mantenido.

En la imagen 5.22, extraída directamente de este servicio, se puede ver que el tamaño de la red tuvo dos picos a la baja a principios del año 2019, pasando de más de 6000 nodos a menos de 4000. Esto indica que existe la posibilidad de que, bajo condiciones particulares, la red Tor pierda parte de su funcionamiento o que este se vea afectado ya que es posible una disminución considerable y repentina en la cantidad de nodos que la conforman; cabe resaltar que la red fue capaz de recuperarse en poco tiempo.

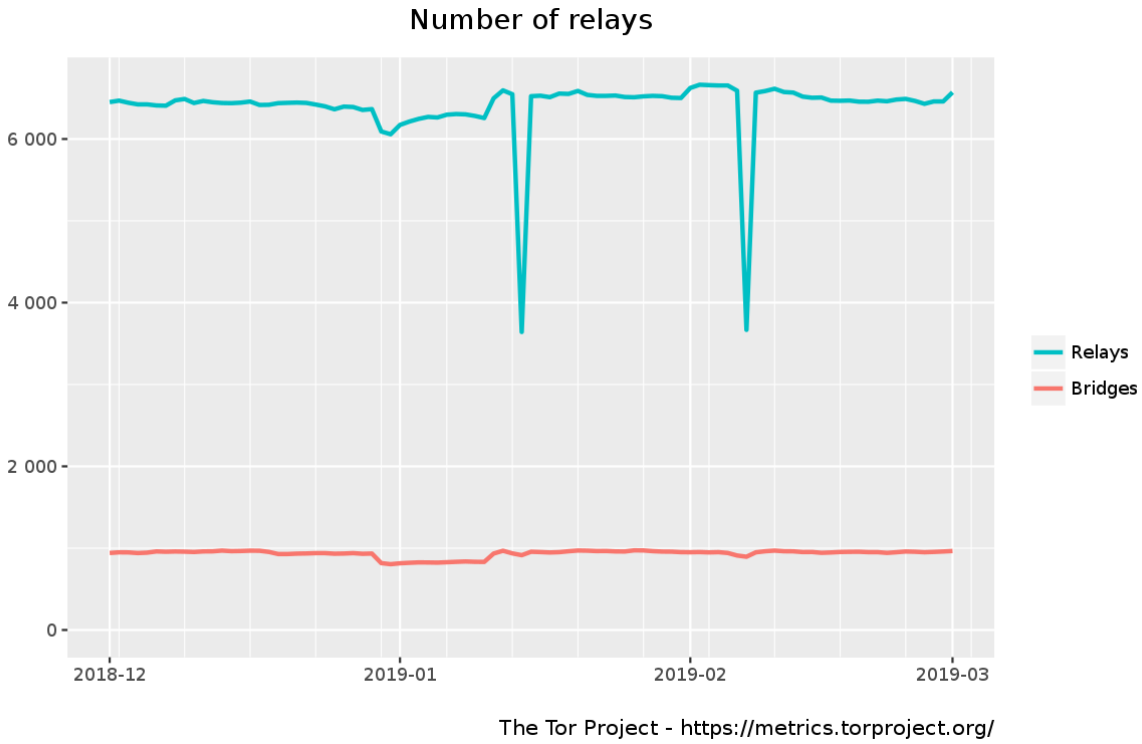

Imagen 5.18. Histórico de número de nodos en últimos meses.

Al analizar un lapso mayor, es posible obtener una idea más amplia del crecimiento que ha sufrido la red. En este caso, se decidió revisar el crecimiento que ha tenido en los últimos 10 años, es decir, desde el año 2009 al año 2019. La gráfica correspondiente a este lapso se muestra en la imagen 5.23.

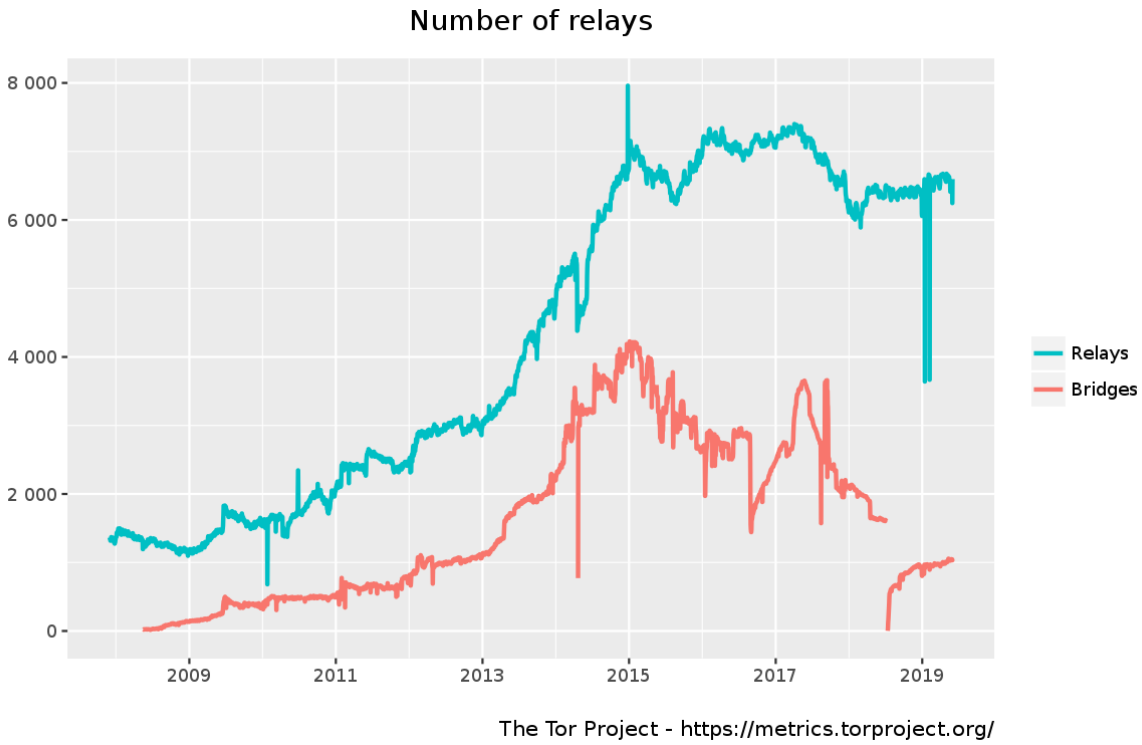

Imagen 5.19. Histórico de número de nodos en los últimos años.

Al analizar las gráficas anteriores, se observa un crecimiento considerable, pasando de poco más de 1000 nodos en el año 2009 a más de 6000 en el año 2019. En esa gráfica se observan los picos mostrados en la imagen 5.22, con lo que se puede observar que se trató de dos eventos muy excepcionales, pues en más de 10 años no se ven caídas en el tamaño de la red tan dramáticas.

Cabe señalar que, aunque en 10 años la red ha crecido prácticamente 6 veces, en el año 2019 se observa una disminución generalizada si se compara con los años inmediatos anteriores. Por ejemplo, entre los años 2016 y 2017 se tenían más de 7000, bajando a 6000 para el 2018 y manteniendo un tamaño similar hasta 2019.

Asimismo, al analizar los puentes se observa un comportamiento similar, pasando de unos cuantos cientos de puentes en el año 2009 a más de 4000 en el año 2015. Sin embargo, del año 2018 al 2019 se observa una disminución dramática de estos, llegando a aproximadamente 1000 puentes.

Estos datos muestran que en 10 años la cantidad de personas y organizaciones dispuestas a apoyar al proyecto mediante la instalación de nodos y puentes ha crecido considerablemente. Sin embargo, es probable que actualmente, en el año 2019, la red se encuentre en dificultades pues tiene considerablemente menos infraestructura que un par de años atrás. Lo que indica que la red, aunque ampliamente apoyada, fácilmente puede verse afectada si deja de haber personas dispuestas a apoyar el proyecto. De esta forma, si la cantidad de demanda de uso de la red crece mientras

que la oferta de nodos se reduce, la disponibilidad en las conexiones puede verse afectada.

5.3.2 Análisis de la disponibilidad basado en el número de conexiones por nodo

Para poder analizar la disponibilidad, visto desde el punto de vista de un nodo, se realizó un registro de las conexiones existentes y la cantidad de errores que el nodo ha detectado. Cabe destacar que, además de las 19 horas de captura mencionadas previamente durante su configuración de nodo de salida, el nodo se mantuvo con una configuración de nodo de entrada durante más de 6 meses. En este lapso, únicamente se mantuvo un monitoreo constante del servicio tor, el cual únicamente se reinició mediante instrucciones explícitas, es decir, no presentó errores que impliquen una afectación en el servicio provisto por parte del nodo hacia la red. En la imagen 5.24, obtenida nuevamente del servicio de búsquedas del proyecto Tor, se muestra la cantidad de bytes recibidos y enviados hacia la red desde que inició su funcionamiento. Se destaca que en ningún momento se vio afectado el servicio, lo que indica una amplia disponibilidad en el servicio tor instalado en los diferentes tipos de nodos.

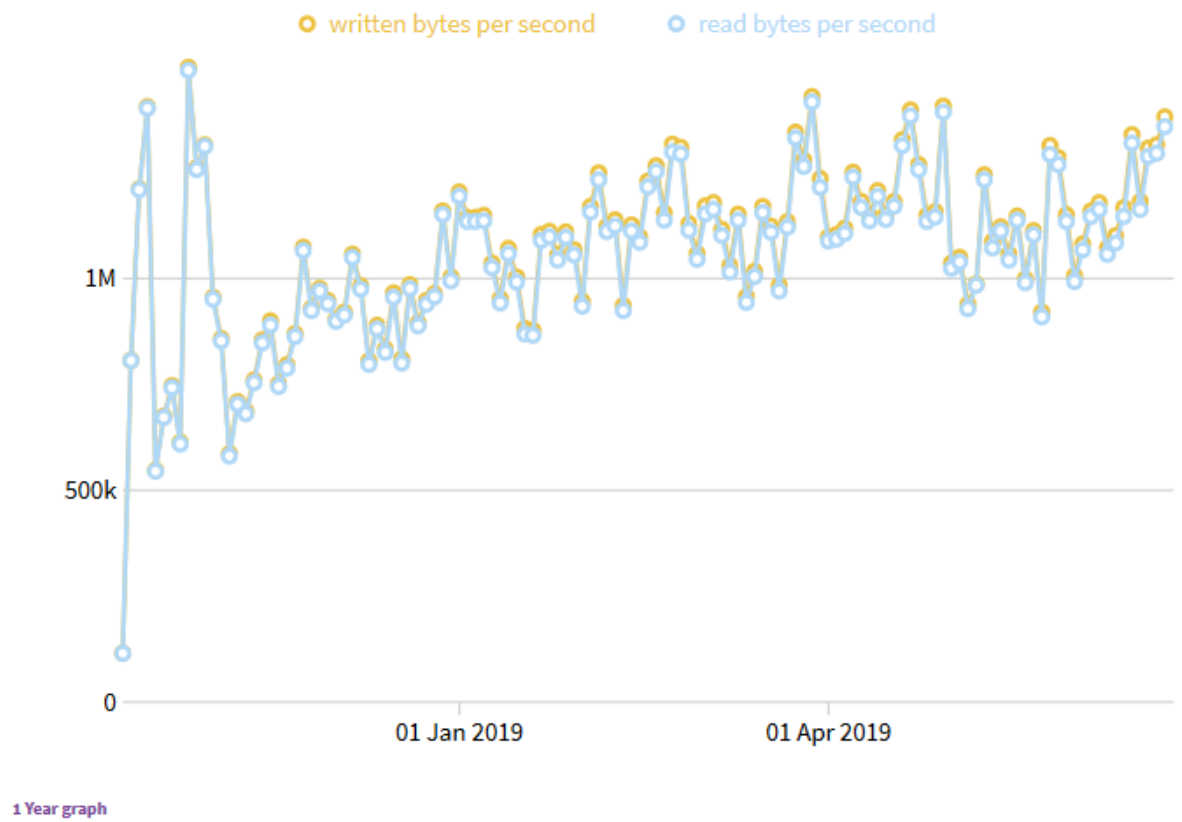

Imagen 5.20. Cantidad de tráfico visto en el nodo reportado por la red.

Asimismo, se analizó la cantidad de conexiones establecidas con el servicio tor en el nodo durante cada minuto de un día entero. Para lograrlo, se escribió una tarea programada que se ejecuta cada minuto y que ejecuta un listado de todas las conexiones, las filtra para obtener únicamente las correspondientes al servicio tor y las cuenta. Resultado de este conteo, se obtuvo la gráfica mostrada en la imagen 5.25. Se observa que, por lo menos, el nodo es capaz de mantener sin problemas más de 2500 conexiones por minuto, logrando un promedio de 3206 conexiones. Estos resultados indican que el programa tor, encargado de enrutar el tráfico recibido en cada nodo, puede mantener una gran cantidad de conexiones simultáneas en equipos con pocos recursos sin generar errores, garantizando un alto nivel de disponibilidad durante el funcionamiento normal del servicio. Es decir, sin recibir ataques de denegación de servicio.

Imagen 5.21. Conexiones por minuto durante un día entero.

5.3.3 Análisis de la disponibilidad en las conexiones de un cliente

Posteriormente, se analizó la disponibilidad visto no desde el punto de vista de la red ni de un nodo, sino de un cliente. Para lograrlo, se configuró una máquina virtual con conexión a la red Tor, junto con un puerto de administración para poder, entre otras cosas, monitorear los circuitos creados por el cliente para acceder a la red.

Se utilizó un programa creado por miembros de la comunidad Tor, cuya función es listar los nodos que forman los diferentes circuitos que el cliente ha creado, incluyendo su alias y su dirección IP. En la imagen 5.26 se muestra la salida de éste al ser ejecutado. Cabe resaltar que el cliente crea más de un circuito, lo que ayuda a la disponibilidad de las comunicaciones si es que el que se encuentra en uso actualmente falla. Sin embargo, se muestra que los diferentes circuitos utilizan el

mismo nodo de entrada a la red, es decir, todos tienen un mismo punto de falla. Por lo que, si el nodo de entrada es incapaz de dar el servicio, todos los circuitos que han sido creados se vuelven inútiles y nuevos circuitos deben ser creados.

```
root@TraffAnalysis:/opt/circuits# python list_circuits.py

Circuit4(GENERAL)
|-3CAE19138E4E025CFB42770075DAAAF8FA0A1439(drjohn,51.75.144.67)
|-5BCC48732850A596D72A5B33D66711ECDA59BC7B(Unnamed,137.226.111.123)
+-7E6E9A6FDDB8DC7C92F0CFCC3CBE76C29F061799(Quintex48,199.249.230.71)

Circuit5(GENERAL)
|-3CAE19138E4E025CFB42770075DAAAF8FA0A1439(drjohn,51.75.144.67)
|-72A33E374DE5AD1E617394896F0FD8B092D3E7C4(absturztaubetor,81.201.201.169)
+-14678AB26ED87E843B232EAEDF2A7BB384652473(rendon,178.175.132.210)

Circuit6(GENERAL)
|-3CAE19138E4E025CFB42770075DAAAF8FA0A1439(drjohn,51.75.144.67)
|-00E1649E69FF91D7F01E74A5E62EF14F7D9915E4(dragonhoard2,116.203.50.182)
+-95A964842E11F9E33C432995811587742922DDD0(GermanPeach,37.120.162.126)
```

Imagen 5.22. Circuitos creados en un cliente de la red.

Tomando en cuenta que todos los circuitos utilizan el mismo nodo de entrada, el script se modificó para mostrar únicamente el alias de los nodos de salida en los diferentes circuitos formados en el cliente, así como la hora en la que se obtuvo este listado. Esta modificación se realizó con el fin de determinar cuánto tiempo tarda un cliente en formar nuevamente circuitos para navegar dentro de la red una vez que falla el nodo de entrada que se encuentra utilizando.

Para lograrlo, se configuró el nodo para ser un nodo de entrada y se agregó una configuración en el cliente para usar preferentemente este nodo. Cabe resaltar que no se forzó a usarlo, ya que de ser así, no utilizaría otros nodos bajo ninguna circunstancia, incluso si el nodo configurado llegara a fallar. Una vez que el cliente comenzó a utilizar el nodo como entrada, se ejecutó el script de forma cíclica para mostrar los diferentes nodos de salida utilizados y, posteriormente, se detuvo el servicio en el nodo de entrada.

En la imagen 5.27, se muestra que el servicio tor en el nodo se detuvo a las 04:12:26 UTC, mientras que en la imagen 5.28 se muestra que el cliente usó los circuitos formados con este nodo hasta las 04:12:28 UTC. Asimismo, a las 04:12:35 UTC, es decir 7 segundos más tarde, el cliente comienza a formar nuevos circuitos.

```
root@km14102-03:~# date --utc "+%H:%M:%S" ; service tor stop
04:12:26
```

Imagen 5.23. Detención de servicio tor y su hora.

```
04:12:26.515743 niftyrabbitrat,radieschen,cryptocrax0r,niftydiatomys,TWCUGRelayNode
04:12:27.082275 niftyrabbitrat,radieschen,cryptocrax0r,niftydiatomys,TWCUGRelayNode
04:12:27.668263 niftyrabbitrat,radieschen,cryptocrax0r,niftydiatomys,TWCUGRelayNode
04:12:28.572253 niftyrabbitrat,radieschen,cryptocrax0r,niftydiatomys,TWCUGRelayNode
04:12:35.806534 CENSURFRITITDKEXIT4**,UOfMichigan
04:12:37.670155 CENSURFRITITDKEXIT4**,UOfMichigan,MerkelMussWegIAfD,chaosDelroth
04:12:39.100060 CENSURFRITITDKEXIT4**,UOfMichigan,MerkelMussWegIAfD,chaosDelroth,zwewwlLV2
04:12:40.387566 CENSURFRITITDKEXIT4**,UOfMichigan,MerkelMussWegIAfD,chaosDelroth,zwewwlLV2
```

Imagen 5.24. Creación de nuevos circuitos al fallar los existentes.

Lo anterior indica que, una vez que el cliente detecta que sus circuitos actuales dejan de ser útiles por una falla en el nodo de entrada, este es capaz de solicitar nuevos circuitos a la red. Asimismo, esta logra otorgarlos en un tiempo menor a 10 segundos. Cabe señalar que, dado que la técnica de enrutamiento de Tor se basa en circuitos estáticos y no en la redundancia de las conexiones, se concluye que la red tiene una alta capacidad para mantener la disponibilidad en las conexiones de sus clientes.

Capítulo 6 Conclusiones

6.1 Instalación y configuración de un nodo en la red Tor

Administrar y configurar un nodo, del tipo que sea, dentro de la red Tor requiere pocos recursos en cuanto a ancho de banda y a hardware, por lo que es posible hacerlo con una inversión pequeña mensual para la renta de un servidor privado virtual o desde dispositivos hogareños si se tiene el ancho de banda suficiente. En este sentido, únicamente se requieren habilidades técnicas para configurar y administrar de forma segura un equipo que estará expuesto a internet y que, por lo tanto, estará expuesto a diversas amenazas. Asimismo, gracias a que el proyecto Tor ha desarrollado paquetes de software y ejecutables que son compatibles con diferentes sistemas operativos, la instalación y configuración de un nodo se simplifica. Por lo anterior, es viable que prácticamente cualquier persona que cuente con conocimientos básicos o intermedios de administración de servidores pueda instalar y mantener un nodo de entrada o intermedio.

Por otro lado, si se quiere administrar un nodo de salida se requiere una amplia investigación para poder determinar la mejor forma de llevarlo a cabo, considerando factores como la potencialmente alta cantidad de quejas que podrían recibirse y si el proveedor de servidores lo permite.

6.2 Actividad de los usuarios de la red Tor en la red abierta

En este trabajo de investigación se demostró que un gran porcentaje del uso que se le da a la red Tor es para visitar sitios muy comunes para los usuarios promedios de internet, como buscadores y redes sociales, lo que indica que existen personas que utilizan la red como un esfuerzo por mantener su anonimato en sus actividades digitales cotidianas. Sin embargo, cabe destacar que estos esfuerzos podrían ser inútiles si no se tiene el cuidado suficiente al usarla, ya que si se utilizan estos servicios otorgando datos personales no hay nada que la red Tor pueda hacer para mantener su anonimato, ya que su funcionamiento se basa en el ocultar el origen de las conexiones, más no en evitar el envío de información personal.

Por otro lado, el trabajo realizado sirvió para mostrar un panorama sobre el uso que se la da a la red desde un punto de vista geográfico. En este sentido, se demuestra que las necesidades sociales y los intereses digitales varían drásticamente en las diferentes partes del mundo. Por un lado, la mayor cantidad de usuarios de la red Tor pertenecen a países cuyo desarrollo ha llevado a tener un alto índice de penetración del internet o que tienen un alto nivel de vigilancia y espionaje entre su población, lo que naturalmente implica en las personas una mayor preocupación por su anonimato

y la privacidad de sus datos. Por otro lado, se ve mucha menor interacción con la red Tor desde países con un menor desarrollo tecnológico, lo cual es un reflejo del comportamiento de la red abierta, dados los recursos tecnológicos y penetración de internet existentes en estos países. Adicionalmente, se puede asociar con un menor interés de sus habitantes por mantenerse anónimos y por involucrarse en todo tipo de proyectos en general, dado que suelen tener otro tipo de preocupaciones.

Finalmente, dado que la red Tor ha adquirido fama de ser el origen de mucha actividad maliciosa, haciendo uso de diversas técnicas se analizó la actividad sospechosa o maliciosa encontrada dentro de la red. Como resultado de dicho análisis se encontró una amplia variedad de actividades potencial o totalmente maliciosas, como escaneos de vulnerabilidades, tráfico relacionado con diferentes tipos de malware, sitios *phishing*, envío de spam y correo malicioso, etcétera. Empero, estas actividades están lejos de ser mayoría durante el análisis realizado por lo que las personas deberían descartar la idea de que esta red es usada meramente con fines maliciosos.

Sin embargo, dado que no es complicado instalar y analizar nodos de salida, es probable que esto sea realizado por personas como especialistas de seguridad e investigadores, por considerarlo como un método viable, adicional a los bien conocidos *honeypots*, para generar inteligencia de amenazas y poder alertar sobre posibles campañas maliciosas, así como para comprender las técnicas, tácticas y procedimientos de los atacantes que aprovechan esta red.

6.3 Seguridad de la información en la red Tor

La red Tor ha implementado medidas para mantener la información que circula a través de ella confidencial, íntegra y disponible, por ejemplo, al cifrar todos los paquetes IP con las llaves públicas de los nodos que forman los circuitos. Sin embargo, su naturaleza basada en un conjunto de servidores proxy interconectados para proveer anonimato contiene inherentemente una debilidad al tener que descifrar totalmente los datos en su último punto antes de salir de la red Tor hacia la red abierta, es decir, en los nodos de salida. Si bien se demostró que la mayor parte del tráfico se encuentra cifrado en la capa de aplicación, también se demostró que existe una gran cantidad de datos no cifrados que podrían ser usados de forma maliciosa por los administradores de los nodos de salida.

Mediante pruebas de concepto, se demostró que en los nodos de salida es posible espiar datos sensibles que no se encuentran cifrados en la capa de aplicación, vulnerando así la confidencialidad de la información de los clientes. Adicionalmente, al vulnerar la confidencialidad, también se demostró que es viable eliminar el anonimato provisto por la red Tor, ya que se puede obtener información de las personas que usan la red si estas se autentican en servicios que no cifren los datos, ya que las credenciales de acceso a estos pueden ser utilizadas para rastrear a los usuarios.

En cuanto al análisis a la integridad de la información, se realizó una prueba de concepto de un ataque al protocolo DNS, con el que se analizó y comprobó que es viable manipular lo que los clientes envían y reciben si estos no utilizan las versiones seguras de los protocolos de capa de aplicación. De esta forma, al igual que sucede con el anonimato que la red puede proveer, la eficacia para que los datos se encuentren seguros depende directamente de las costumbres de los usuarios y sus acciones al hacer uso de ella, así como de seguir las recomendaciones de uso del proyecto como usar únicamente el navegador Tor para acceder a la red Tor.

En cuanto a la disponibilidad de la red Tor, así como los datos que se envían en ella, se observan medidas para protegerla, como por ejemplo la creación de múltiples circuitos en los clientes y una alta estabilidad en los nodos. De igual manera, se observó una buena capacidad de respuesta por parte de la red para crear nuevos circuitos para los clientes en caso de que los actuales fallen. A pesar de que esta creación de circuitos es mucho más lenta comparada con la velocidad en que se buscan nuevas rutas en redes cuyo objetivo no es anonimizar conexiones, dada la naturaleza de la red se considera una ventaja en cuanto a disponibilidad.

Sin embargo, dada la alta demanda de uso que tiene la red, la disponibilidad de esta depende en gran medida de la cantidad de personas dispuestas a apoyar económicamente o con la administración de nuevos nodos, lo que podría en un futuro significar un punto a su favor o en su contra, dependiendo de la sociedad y de su aceptación o rechazo a la constante vigilancia existente en la red.

6.4 Trabajo Futuro

A continuación, se enlistan algunas ideas relacionadas con el trabajo realizado que pueden ser profundizadas posteriormente y que, por el alcance de la investigación, no fueron estudiados:

- Poner en marcha un servicio oculto en la red Tor y analizar el tipo de actividad registrada en él.

- Diseñar y programar un *bot* o *crawler* para visitar automáticamente los diferentes servicios ocultos basados en HTTP cuyo nombre es público, para generar inteligencia de amenazas basándose en las conversaciones existentes en estos.

- Ampliar la infraestructura con diversos nodos de la red para correlacionar una mayor cantidad de información sobre campañas de ataques.

- Fomentar activamente el estudio de la red en el sector académico.

- Estudiar otro tipo de ataques a la seguridad de la información en los que se vulneren los datos cifrados a nivel de capa de aplicación.

- Realizar trabajos de inteligencia y *hunting* para encontrar nodos de salida maliciosos donde se apliquen ataque mostrados en esta tesis y otros diferentes.

Bibliografía

Python for beginners. (2013). *Using pywhois for retrieving WHOIS information*. Recuperado el Noviembre de 2018, de https://www.pythonforbeginners.com/dns/using-pywhois

Acunetix. (2017). *Cyber Threats vs Vulnerabilities vs Risks*. Recuperado el Agosto de 2018, de https://www.acunetix.com/blog/articles/cyber-threats-vulnerabilities-risks

Arch Linux. (2019). *Dnsmasq*. Recuperado el Abril de 2019, de https://wiki.archlinux.org/index.php/Dnsmasq_(Espa%C3%B1ol)

Arma. (2009). *Tor Blog*. Recuperado el 06 de Septiembre de 2019, de https://blog.torproject.org/why-tor-slow-and-what-were-going-do-about-it

Arma. (2010). *Tor blog*. Recuperado el 06 de Septiembre de 2019, de https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea

Avast. (2018). *Pharming*. Recuperado el Agosto de 2018, de https://www.avast.com/es-es/c-pharming

Avast. (2018). *Phishing*. Recuperado el Agosto de 2018, de https://www.avast.com/es-es/c-phishing

Bejtlich, R. (2005). Structured Traffic Analysis. *INsecure Magazine, 4*. Recuperado el Agosto de 2018, de https://www.helpnetsecurity.com/dl/insecure/INSECURE-Mag-4.pdf

Coderwall. (2016). *Looking at HTTP requests using tcpdump*. Recuperado el Abril de 2019, de https://coderwall.com/p/tladjw/looking-at-http-requests-using-tcpdump

Conville. (2019). *A beginner's guide to Tor relay hardening*. Recuperado el Septiembre de 2019, de https://www.sccs.swarthmore.edu/users/16/mmcconv1/tor-hardening.html

DCode. (s.f.). *Luhn Number Checksum*. Recuperado el Abril de 2019

Debian Wiki. (2018). *Unattended Upgrades*. Recuperado el Septiembre de 2018, de https://wiki.debian.org/UnattendedUpgrades

Devops & Python. (2017). *Register Python script as a Linux system service*. Recuperado el Enero de 2019, de http://devopspy.com/linux/python-script-linux-systemd-service/

DIE. (2018). *Ngrep – Linux man page*. Recuperado el Noviembre de 2018, de https://linux.die.net/man/8/ngrep

DIE. (2018). *Tshark – Linux man page*. Recuperado el Octubre de 2018, de https://linux.die.net/man/1/tshark

Dingledine, R., & Mathewson, N. (2018). *Tor Protocol Specification*. Recuperado el Septiembre 2018, de https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

Electronic Frontier Foundation. (2018). *What is Tor?* Recuperado el Septiembre de 2018, de https://www.eff.org/torchallenge/what-is-tor.html

Electronic Frontier Foundation. (2019). *How HTTPS and Tor Work Together to Protect Your Anonymity and Privacy*. Recuperado el 08 de Septiembre de 2019

Electronic Frontier Foundation. (2019). *HTTPS Everywhere*. Recuperado el 08 de Septiembre de 2019, de https://www.eff.org/https-everywhere

FIRST. (2018). *Common Vulnerability Scoring System v3.0: Specification Document*. Recuperado el Agosto de 2018, de https://www.first.org/cvss/v3.0/specification-document

Google Public DNS. (2019). *Security Benefits*. Recuperado el Septiembre 08 de 2019, de https://developers.google.com/speed/public-dns/docs/security?csw=1#add_entropy

IBM Knowledge Center. (2018). *DNS query types*. Recuperado el Octubre de 2018, de https://www.ibm.com/support/knowledgecenter/en/SS5MD2_7.4.0.1/com.ibm.itcamt.doc/ism/dita/rg/concept/ISM_Ref_DNS_guideline_dnsquery.html

ICANN. (2019). *ICANN WHOIS*. Recuperado el Abril de 2019, de https://whois.icann.org/es

Jordan. (2015). *Jordan-Wright*. Recuperado el 28 de Agosto de 2019, de https://jordan-wright.com/blog/2015/05/14/how-tor-works-part-three-the-consensus/

Kaspersky. (2013). *¿Qué es un ataque Man-In-The-Middle?* Recuperado el Agosto de 2018, de https://latam.kaspersky.com/blog/que-es-un-ataque-man-in-the-middle/469/

Kaspersky. (2018). *¿Qué es el spear phishing?* Recuperado el Agosto de 2018, de https://latam.kaspersky.com/resource-center/definitions/spear-phishing

MalwareBytes. (2017). *Ransomware*. Recuperado el Septiembre de 2018, de https://es.malwarebytes.com/ransomware/

Manners, D. (2012). *The User Agent Field: Analyzing and Detecting the Abnormal or Malicious in your Organization*. Recuperado el Agosto de 2018, de https://www.sans.org/reading-room/whitepapers/malicious/user-agent-field-analyzing-detecting-abnormal-malicious-organization-33874

McConville, M. (2018). *A beginner's guide to Tor relay hardening.* Recuperado el Septiembre de 2018, de https://www.sccs.swarthmore.edu/users/16/mmcconv1/tor-hardening.html

Muthukadan, B. (2018). *Selenium with Python*. Recuperado el Noviembre de 2018, de https://selenium-python.readthedocs.io/locating-elements.html

Netresec. (2019). *Network Miner*. Recuperado el Marzo de 2019, de https://www.netresec.com/?page=networkminer

Oberheide, J. (2008). *Dpkt Tutorial #2: Parsing a PCAP File*. Recuperado el Octubre de 2018, de https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/

OWASP. (2018). *Penetration testing methodologies*. Recuperado el Agosto de 2018, de https://www.owasp.org/index.php/Penetration_testing_methodologies

PacketSyndicate. (2018). *Changing IP Addresses in PCAP Files for Sharing*. Recuperado el Febrero de 2019, de https://packetsyndicate.com/change-ip-addresses-in-pcap-files-for-sharing

PacketTotal. (2019). *About*. Recuperado el Febrero de 2019, de https://packettotal.com/about.html

PCI Security Standards. (2015). *Penetration Testing Guidance*. Recuperado el Agosto de 2018, de https://www.pcisecuritystandards.org/documents/Penetration_Testing_Guidance_March_2015.pdf

Pcredz. (2019). *Pcredz*. Recuperado el Marzo de 2019, de https://github.com/lgandx/PCredz

Plotly. (2018). *Getting Started with Plotly for Python in Python*. Recuperado el Octubre de 2018, de https://plot.ly/python/getting-started

Python Hosted. (2018). *Python-GeoIP*. Recuperado el Octubre de 2018, de https://pythonhosted.org/python-geoip/

Rosenberg, M., & Dance, G. (2018). Así funcionaba la recolección de datos de Cambridge Analytica. *The New York Times*. Recuperado el Agosto de 2018, de https://www.nytimes.com/es/2018/04/10/facebook-cambridge-analytica/

S. Davie, B., & Peterson, L. (2003). *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers.

SANS Penetration Testing. (2017). *SANS Poster: Building a better Pen Tester*. Recuperado el Agosto de 2018, de https://pen-testing.sans.org/blog/2017/12/12/sans-poster-building-a-better-pen-tester-pdf-download?msc=ptslider5

SANS Technology Institute. (2019). *TCP/IP and tcpdump Pocket Reference Guide*. Recuperado el Abril de 2019, de https://www.sans.org/security-resources/tcpip.pdf

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software.* No Starch Press.

Stallings, W. (2000). *Comunicaciones y Redes de computadores.* Prentice Hall.

Stallings, W. (2006). *Cryptography and Network Security.* Prentice Hall.

Suricata. (2019). *About*. Recuperado el Febrero de 2019, de https://suricata-ids.org/about/

Tanenbaum, A. S., & Wetherall, D. (2012). *Redes de computadoras.* Pearson.

Tarlogic. (2016). *¿Qué son los Watering Hole Attacks?* Recuperado el Septiembre de 2018, de https://www.tarlogic.com/blog/los-watering-hole-attacks/

Tcpdump. (2012). *Manpage of TCPDUMP*. Recuperado el Septiembre de 2018, de http://www.tcpdump.org/manpages/tcpdump.1.htm

Tcpreplay. (2018). *Tcpreplay*. Recuperado el Febrero de 2019, de http://tcpreplay.synfin.net/tcpreplay-edit.html

The Editorial Board. (2014). Edward Snowden, Whistle-Blower. *The New York Times*. Recuperado el Septiembre de 2018, de https://www.nytimes.com/2014/01/02/opinion/edward-snowden-whistle-blower.html

The TCP/IP Guide. (2005). *DNS Message Header and Question Section Format*. Recuperado el Noviembre de 2018, de http://www.tcpipguide.com/free/t_DNSMessageHeaderandQuestionSectionFormat.htm

The Tor Projec Blog. (2013). *The lifecycle of a new relay*. Recuperado el Septiembre de 2018, de https://blog.torproject.org/lifecycle-new-relay

The Tor Project. (2018). *About Tor*. Recuperado el Septiembre de 2018, de https://2019.www.torproject.org/about/torusers.html.e

The Tor Project. (2018). *Exit Policy policy,policy*. Recuperado el Septiembre de 2018, de https://www.torproject.org/docs/tor-manual.html.en#ExitPolicy

The Tor Project. (2018). *Tor: Overview*. Recuperado el Septiembre de 2018, de https://2019.www.torproject.org/about/overview.html.en

The Tor Project Metrics. (2018). *Tor Hidden Service Descriptors*. Recuperado el Agosto de 2018, de https://metrics.torproject.org/collector.html#type-hidden-service-descriptor

The Tor Project Metrics. (2019). *Servers*. Recuperado el Mayo de 2019, de https://metrics.torproject.org/networksize.html

The Tor Project Wiki. (2018). *Good Bad ISP*. Recuperado el Septiembre de 2018, de https://trac.torproject.org/projects/tor/wiki/doc/GoodBadISPs

The Tor Project Wiki. (2018). *Tor Relay Guide: Debian/Ubuntu*. Recuperado el Septiembre de 2018, de https://trac.torproject.org/projects/tor/wiki/TorRelayGuide/DebianUbuntu

The Tor Project Wiki. (2018). *Tor Relay Guide: Technical Setup*. Recuperado el Septiembre de 2018, de https://trac.torproject.org/projects/tor/wiki/TorRelayGuide#Parttwo:technicalsetup

Tor Nyx. (2018). *Tor Nyx*. Recuperado el Octubre de 2018, de https://nyx.torproject.org

Venafi. (08 de Septiembre de 2018). *What are SSL Stripping Attacks?* Obtenido de https://www.venafi.com/blog/what-are-ssl-stripping-attacks

VirusTotal. (2019). *How it Works*. Obtenido de https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works

VirusTotal Developers. (2019). *Public vs Private API*. Recuperado el Enero de 2019, de https://www.virustotal.com/en/documentation/public-api

Wolf, G. (2017). Independencia en el Ciberespacio. *SG, 55*. Recuperado el Septiembre de 2018, de https://sg.com.mx/revista/55/independencia-el-ciberespacio

Wolf, G. (2018). Funcionamiento de una Red Anonimizadora: La red Tor. *SG, 56*. Recuperado el Septiembre de 2018, de https://sg.com.mx/revista/56/red-anonimizada-tor

Wolf, G. (2018). Privacidad y Anonimato en Redes: ¿y la sociedad? *SG, 57*. Recuperado el Septiembre de 2018, de https://sg.com.mx/revista/57/privacidad-anonimato-redes

Xplico Wiki. (2018). *Xplico Interface*. Recuperado el Abril de 2019, de http://wiki.xplico.org/doku.php?id=web_interface

Yuan, L. (2019). Así funciona una fábrica de censura en China. *The New York Times*. Recuperado el Enero de 2019, de https://www.nytimes.com/es/2019/01/05/censura-internet-china/

Anexo A. Glosario

Agente de usuario: Es una cabecera de los mensajes usados en el protocolo HTTP que permite identificar el sistema operativo y software usados en las peticiones de los clientes.

Analizador de protocolos: Herramienta que permite analizar los encabezados y contenido de los diferentes protocolos de red pertenecientes al modelo TCP/IP. Comúnmente son conocidos cono *sniffers*. Son de especial utilidad para hallar errores en las comunicaciones y requieren privilegios administrativos en el equipo donde se ejecutan, ya que configuran la interfaz de red en modo monitor para poder procesar todos los paquetes de la red.

Archivo hosts: Archivo de configuración existente en los sistemas operativos Windows y los tipos Unix que sirve para indicar traducciones entre nombres de dominio y direcciones IP. Los dispositivos revisan el contenido de este archivo previo a realizar consultas DNS a un servidor DNS.

API: *Application Programming Interface*, es un conjunto de herramientas publicado por los dueños de determinado software para que éste pueda comunicarse con productos y servicios de terceros, sin la necesidad de que conozcan su funcionamiento.

Base 64: Es un sistema de numeración que utiliza 64 símbolos para codificar los datos. Es ampliamente utilizado para codificar datos en binario para que estos sean almacenados o transferidos por la red.

Debian: Es un sistema operativo de distribución libre y código abierto que usa principalmente el kernel de *Linux* para funcionar.

Bases de datos GeoLite2: Son bases de datos gratuitas que relacionan direcciones IP a ubicaciones geográficas. Existen otras bases de datos llamadas *GeoIP2* que son de paga y suelen ser más precisas ya que se actualizan constantemente y no de forma mensual como con las bases *GeoLite*.

Hardening: El *hardening* se refiere a un conjunto de configuraciones aplicadas a un sistema informático para que sea más seguro, reduciendo las vulnerabilidades existentes en él y, de esta forma, reduciendo la superficie de ataque.

IANA: *Internet Assigned Numbers Authority*, es la entidad responsable de asignar las direcciones IP y nombres de dominio, entre otros, en internet a nivel global.

IDS: *Intrussion Detection System* es un Sistema informático cuya función es analizar el tráfico de una red y encontrar actividad sospechosa o maliciosa a través de firmas o comportamiento anómalo, y alertar dicha actividad.

Libpcap: Es una biblioteca para captura de paquetes utilizada en sistemas Unix, aunque existe un símil en Windows conocido como *Winpcap*. Estas bibliotecas son usadas por programas orientados a la captura y análisis de tráfico de red como *Wireshark, tshark, tcpdump,* entre otros. El formato de los archivos generados con estas bibliotecas es *pcap*.

Sistema de búsquedas de la red Tor: Aplicación web desarrollada por el proyecto Tor que permite realizar búsquedas simples ya avanzadas de nodos de la red, a través de su dirección IP, banderas, su alias, etcétera. https://atlas.torproject.org/

Sqlite: Es un motor de bases de datos SQL pequeño y rápido orientado a interactuar con datos de aplicaciones pequeñas que no requieren la administración directa de un humano y que no tienen grandes volúmenes de datos, como las de teléfonos celulares y tabletas. Implementa todas las funcionalidades básicas del lenguaje SQL.

Systemd: Es un gestor de servicios utilizado recientemente en sistemas operativos Linux que se ejecuta con el identificador de proceso 1 y, por lo tanto, tiene como funcionalidad arrancar y gestionar el resto del Sistema operativo.

Táctica: Según el *Framework* de amenazas de *Mitre ATTK&CK*, una táctica es ***el qué*** que los atacantes tratan de lograr, como por ejemplo la recolección y exfiltración de información sensible, la elevación de privilegios en un sistema, entre otros.

Técnica: Según el *Framework* de amenazas de *Mitre ATTK&CK*, una técnica es un comportamiento de un atacante específico para alcanzar un objetivo, es decir, es ***el cómo*** un atacante logra su objetivo. Un ejemplo de una técnica es el envío de correos de phishing dirigido.

Top-level domain: Dentro de la base de datos jerárquica que forma el sistema de nombres de dominio (DNS), un *Top-level domain* o *TLD* es un dominio configurado en las zonas raíz de internet y que sirven para indicar la categoría del nombre de dominio completo. Al ver un nombre de dominio, es fácil identificar un TLD, ya que es el que se encuentra al final del nombre, después del último punto. Por ejemplo, los TLD "*mx*" y "*fr*" hacen referencia a dominios asignados a México y Francia respectivamente. Asimismo, el TLS "*edu*" indica un dominio educativo y "*gov*" uno gubernamental.

Pandas: Es una biblioteca de código abierto desarrollada para el lenguaje de programación Python que tiene como objetivo facilitar el uso de las estructuras de datos y facilitar el análisis de datos.

Plotly: Es una biblioteca orientada para lenguajes de programación como Python y *R,* sirve para la visualización de datos, mediante la generación de todo tipo de gráficas.

VPN: *Virtual Private Network* o Red Privada virtual es una tecnología que permite acceder a redes locales de forma remota y segura, ya que el tráfico entre el origen y la red destino se encuentra cifrado. Suele ser un método usado para evitar las restricciones y bloqueos existentes en internet.

Anexo B. Programas

Creación de nuevas capturas de tráfico sin HTTPS

Script en Bash cuyo objetivo fue filtrar nuevamente el tráfico capturado para eliminar todo lo relativo al puerto 443, es decir, al protocolo HTTPS.

```
#!/bin/sh

#Generación de nuevos archivos eliminando todo el tráfico https
for dump in ./tor_traffic/*; do
      new_dump="./no_https_traffic/new_"${dump:20}
      tcpdump -r $dump -w $new_dump \
            "tcp and not (tcp port 443 or dst host <VPS_IP>)"
done
```

Extracción de agentes de usuario

Script en Bash que, por cada archivo capturado, realiza un filtro para revisar el puerto 80 y obtener todas las cadenas con la subcadena '*User-Agent*', para obtener un listado de agentes de usuario existentes en todo el tráfico.

```
#!/bin/sh

rm /opt/devices/requests.txt && touch /opt/devices/requests.txt

#Para cada archivo, buscar la cadena 'User-Agent' en todo el tráfico
dirigido al puerto 80
# Redirigir las cabeceras y el payload a un archivo de texto para
procesarlo posteriormente
for dump in no_https_traffic/*; do
      ngrep -q -I "$dump" -i user-agent port 80 >>
/opt/devices/requests.txt
done
```

Extracción de peticiones DNS

Script en *Bash* que utiliza el programa *tshark* para extraer todas las peticiones DNS de los archivos *pcap* y, con ayuda de herramientas de los sistemas operativos tipos Unix, generar un listado ordenado y de fácil lectura.

```
#!/bin/sh

#Generación de nuevos archivos de tráfico con peticiones y respuestas DNS
for dump in ./tor_traffic/*; do
      new_dump="./dns_traffic/dns_"${dump:20}
      tcpdump -r $dump -w $new_dump \
            "udp port 53"
done

#Creación de archivo vacío para contener los nombres solicitados
echo "" > /opt/dns_queries_A.txt

#Para cada archivo nuevo buscar las peticiones DNS,
#guardándolas en minúsculas en el archivo previamente generado
for dump in dns_traffic/*; do
            tshark -r "$dump" -T fields -e dns.qry.name \
                  -e dns.a \
                  -Y "dns.qry.type == 1" \
            | grep -F '.' \
            | awk '{print tolower($0)}' \
            | sort -u >> /opt/dns_queries_A.txt
done
```

Creación de tabla SQLite

Creación de una base de datos con SQLite simple que consta únicamente de una tabla, la cual es usada por otros scripts para almacenar información relativa al análisis de las direcciones IP.

```
#!/bin/sh
sqlite3 tor_real_ips.db <<EOF

DROP TABLE IF EXISTS IPS;
CREATE TABLE IPS(
      id INTEGER,
      ip TEXT,
      tor_relay INTEGER,
      country TEXT,
      UNIQUE(ip),
      CONSTRAINT ips_pk PRIMARY KEY(id) );

EOF
```

Visualización de datos (TLDs y SLDs)

Programa escrito en Python que para generar gráficas de nombres de dominio con *plotly*.

```
#!/usr/bin/python
import sys
import optparse
import plotly.offline as ploff
import plotly.graph_objs as go

def printError(msg, exit=True):
"""Imprime en el error estandar """
    sys.stderr.write('Error:\t%s\n' % msg)
    if exit:
        sys.exit(1)

def checkOptions(opts):
"""Revisa que se introduzca el archivo con listado de nombres de dominio
"""
    if opts.ifile is None:
        printError('You must indicate the file to read')

def getDomains(ifile):
"""Obtiene una lista con todos los dominios"""
    with open(ifile,'r') as i:
        domains = [d.split()[0] for d in i.readlines()]
    return domains

def extractCount(domains, extract_tld, extract_sld, extract_whole):
"""Obtiene 3 diccionarios que incluyen la cuenta de los TLD, SLD y
dominios completos"""
    tldomains = {}
    sldomains = {}
    whole = {}

    for d in domains:
        if '.' not in d: continue
        subdomains = d.split('.')

        if extract_whole:
            if d not in whole:
                whole[d] = 1
            else:
                whole[d] += 1

        if extract_tld:
            dom = subdomains[-1]
            if dom not in tldomains:
                tldomains[dom] = 1
            else:
                tldomains[dom] += 1
```

```
        if extract_sld:
            dom = '%s.%s' % (subdomains[-2],subdomains[-1])
            if dom not in sldomains:
                sldomains[dom] = 1
            else:
                sldomains[dom] += 1

    return tldomains,sldomains,whole

def printDict(dictionary):
"""Imprime un diccionario de forma legible"""
    for key, value in sorted(dictionary.iteritems(), key=lambda (k,v):
(v,k),reverse=True):
        print "%s: %s" % (key, value)

def addOptions():
"""Permite indicar las opciones de línea de comandos para personalizar la
ejecución del programa"""
    parser = optparse.OptionParser()
    parser.add_option('-t','--extract-tld',
                    dest='extract_tld',
                    default=False,
                    action='store_true',
                    help='Extract Top-Level-Domains count from file')
    parser.add_option('-s','--extract-sld',
                    dest='extract_sld',
                    default=False,
                    action='store_true',
                    help='Extract Second-Level-Domains count from file')
    parser.add_option('-w','--extract-whole',
                    dest='extract_whole',
                    default=False,
                    action='store_true',
                    help='Extract the whole domains count from file')
    parser.add_option('-i','--input-file',
                    dest='ifile',
                    default=None,
                    help='Input file with dns queries')
    parser.add_option('-o','--output-file',
                    dest='ofile',
                    default='graph.html',
                    help='Output html file containing the resulting
graph')
    parser.add_option('-g','--graph',
                    dest='graph',
                    default=None,
                    help='Top n domains to graph')
    parser.add_option('-n','--include-others',
```

```
                    dest='others',
                    default=False,
                    action='store_true',
                    help='Add to graph an extra bar including the count
of domains not belonging to the top')
    parser.add_option('-p','--print',
                    dest='prnt',
                    default=False,
                    action='store_true',
                    help='Prints in standard output the results of the
counting')
    opts, args = parser.parse_args()
    return opts

def graphDomains(domainDict, totalDomains, top, others, fileName):
"""Crea con plotly una gráfica de barras con la cuenta de dominios"""
    sortedDict = sorted(domainDict.iteritems(), key=lambda (k,v):
(v,k),reverse=True)[:top]
    if others:
        s = 0
        for k,v in sortedDict:
            s += v
        sortedDict.append(('others',totalDomains-s))

    x = [k for k,v in sortedDict]
    y = [v for k,v in sortedDict]

    data = [go.Bar(x=x,y=y)]
    ploff.plot(data,filename=fileName)

if __name__ == '__main__':
"""Ejecución principal"""
    opts = addOptions()
    checkOptions(opts)
    domains = getDomains(opts.ifile)
    totalDomains = len(domains)
    tld, sld, whole = extractCount(domains, opts.extract_tld,
opts.extract_sld, opts.extract_whole)
    if opts.prnt:
        if tld != {}:
            printDict(tld)
        if sld != {}:
            printDict(sld)
        if whole != {}:
            printDict(whole)
    if opts.graph:
        if tld != {}:
```

```
            graphDomains(tld, totalDomains, int(opts.graph), opts.others,
opts.ofile)
        if sld != {}:
            graphDomains(sld, totalDomains, int(opts.graph), opts.others,
opts.ofile)
        if whole != {}:
            graphDomains(whole, totalDomains, int(opts.graph),
opts.others, opts.ofile)
```

HTTP scrapping

Automatización en Python de peticiones HTTP en el sistema de búsquedas de la red Tor, mediante un sistema para interpretar código *javascript*, para poder determinar si una dirección IP es o no un nodo de Tor.

```
#!/usr/bin/python
import sys
import optparse
import plotly.offline as ploff
import plotly.graph_objs as go
from sqlite3 import connect
from time import sleep
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from geoip import geolite2
driver = webdriver.PhantomJS()
driver.implicitly_wait(10)


def printError(msg, exit=True):
"""Imprime en el error estándar"""
    sys.stderr.write('Error:\t%s\n' % msg)
    if exit:
        sys.exit(1)

def checkOptions(opts):
"""Revisa que se introduzcan las opciones necesarias"""
    if opts.ifile is None:
        printError('You must indicate the connections file to read')
    if opts.dbfile is None:
        printError('You must indicate the database file to read')


def printDict(dictionary):
"""Imprime un diccionario de forma legible"""
    for key, value in sorted(dictionary.iteritems(), key=lambda (k,v):
(v,k),reverse=True):
        print "%s: %s" % (key, value)


def addOptions():
"""Agrega las opciones para personalizar ejecución"""
    parser = optparse.OptionParser()
    parser.add_option('-i','--input-file',
                      dest='ifile',
                      help='File with list of connections')
    parser.add_option('-d','--db-file',
                      dest='dbfile',
                      help='Previously created SQLite database file')
    parser.add_option('-s','--sleep',
                      dest='sleep',
                      default='2',
```

```
                    help='Sleep time in seconds between requests')
    parser.add_option('-c','--commit',
                    dest='commit',
                    default='10',
                    help='Number of insertions before a commit is done in
database')
    opts, args = parser.parse_args()
    return opts


def isTorRelay(ip):
"""Revisa mediante web scrapping si una IP es un nodo de Tor"""
    url = 'https://metrics.torproject.org/rs.html#search/' + ip
    driver.get(url)
    try:
        box = driver.find_element_by_class_name('results_box')
        if 'No Tor relays matched your query' in box.text:
            return 0
        else:
            for i in range(3):
                box = driver.find_element_by_class_name('tip')
            return 1
    except NoSuchElementException as e:
        try:
            if "Unable to find element with class name 'results_box'" in
str(e):
                box = driver.find_element_by_class_name('tip')
                return 1
            elif "Unable to find element with class name 'tip'" in
str(e):
                box = driver.find_element_by_class_name('results_box')
                if 'No Tor relays matched your query' in box.text:
                    return 0
            return 2
        except Exception as e:
            return 2
    except Exception as e:
        return 2

def ips_list(ips_file):
"""Obtiene un listado de todas las direcciones IP a revisar"""
    with open(ips_file,'r') as ips:
        ips = [l.split(':')[0] for l in ips.readlines() if '*' not in l]
        return ips


def isInDB(dbcursor, ip):
```

```
"""Revisa si la dirección IP ya ha sido validada"""
    query = """SELECT * FROM IPS WHERE ip='%s' """ % (ip)
    dbcursor.execute(query)
    rows = dbcursor.fetchall()
    if rows == []:
        return False
    return True


def makeRequest(dbcursor, ip):
"""Hace petición y con base en los resultados agrega a la base de
datos"""
    isRelay = isTorRelay(ip)
    match = geolite2.lookup(ip)
    if match is not None:
        country = match.country if match.country is not None else
'UNKNOWN'
    else:
        country = 'UNKNOWN'
    print isRelay,country,'\n'
    query = """INSERT OR IGNORE INTO IPS(ip,tor_relay,country) VALUES
('%s',%s,'%s');""" % (ip,isRelay,country)
    dbcursor.execute(query)
    return True


if __name__ == '__main__':
"""Ejecución principal que controla el tiempo de espera entre peticiones
y los commits a la BD"""
    opts = addOptions()
    checkOptions(opts)
    con = connect(opts.dbfile)
    with con:
        cur = con.cursor()
        ips = ips_list(opts.ifile)
        n_commit = 0
        for ip in ips:
            print ip
            if not isInDB(cur, ip):
                sleep(float(opts.sleep))
                if makeRequest(cur, ip):
                    n_commit += 1
            if n_commit == int(opts.commit):
                n_commit = 0
                con.commit()
```

Procesamiento de PCAP y visualización de flujos

Programa en Python que analiza las conexiones vistas en un conjunto de archivos *pcap* para poder generar gráficas sobre el uso de la red, considerando un análisis de flujos de estos.

```
#!/usr/bin/python
import sys
import optparse
import plotly.offline as ploff
import plotly.graph_objs as go
from dpkt.pcap import Reader
from dpkt.ethernet import Ethernet
from socket import inet_ntoa

flow_dict = {}

#Flujos configurados en la politica de salida
torrc_ports = (20,21,23,25,43,53,79,80,81,88,110,143,194,220,
            389,443,464,465,531,543,544,554,563,587,636,706,
            749,873,902,903,904,981,989,990,991,992,993,994,
            995,1194,1220,1293,1500,1533,1677,1723,1755,1863,
            2082,2083,2086,2087,2095,2096,2102,2103,2104,3128,
            3389,3690,4321,4643,5050,5190,5222,5223,5228,5900,
            6660,6661,6662,6663,6664,6665,6666,6667,6668,6669,
            6679,6697,8000,8008,8074,8080,8082,8087,8088,8232,
            8233,8443,8888,9418,9999,10000,11371,19294,50002,64738)

class Flow:
"""Clase para definir un flujo"""
    def __init__(self, srcip, dstip):
        self.srcip = srcip
        self.dstip = dstip

    def __str__(self):
        return '%s -> %s' % (self.srcip,self.dstipt)

    def __eq__(self, other):
        return self.__dict__ == other.__dict__

    def __hash__(self):
        return hash((self.srcip,  self.dstip))


def printError(msg, exit=True):
"""Imprime en el error estandar"""
    sys.stderr.write('Error:\t%s\n' % msg)
    if exit:
        sys.exit(1)
```

```
def checkOptions(opts):
"""Revisa que se indique el archivo de entrada y la direccion IP origen a
analizar"""
    if opts.ifile is None:
        printError('You must indicate the file to read')
    if opts.srcip is None:
        printError('You must indicate the source IP address that count
will apply to')


def getFiles(ifile):
"""Obtiene el listado de todos los archivos PCAP a procesar"""
    with open(ifile,'r') as i:
        files = [d.split()[0] for d in i.readlines()]
        return files


def printDict(dictionary):
"""Imprime un diccionario de forma legible"""
    print '\n\n'
    for key, value in sorted(dictionary.iteritems(), key=lambda (k,v):
(len(v),k),reverse=True):
        print "%s: %s" % (key, len(value))


def addOptions():
"""Agrega las posibles opciones para personalizar la ejecución"""
    parser = optparse.OptionParser()
    parser.add_option('-i','--input-file',
                    dest='ifile',
                    default=None,
                    help='Input file with list of traffic files')
    parser.add_option('-o','--output-file',
                    dest='ofile',
                    default='graph.html',
                    help='Output html file containing graph')
    parser.add_option('-g','--graph',
                    dest='graph',
                    default=None,
                    help='Top n ports to graph')
    parser.add_option('-s','--src-ip',
                    dest='srcip',
                    default=None,
                    help='The IP address that this count will apply to')
    parser.add_option('-p','--print',
                    dest='prnt',
                    default=False,
```

```
                    action='store_true',
                    help='Prints in standard output the results of the
counting')
    opts, args = parser.parse_args()
    return opts


def graphFlows(flowDict, top, fileName):
"""Crea una gráfica de pastel con el porcentaje de flujos por puerto
destino"""
    sortedDict = sorted(flowDict.iteritems(), key=lambda (k,v):
(len(v),k),reverse=True)
    graphDict = sortedDict[:top]
    othersDict = sortedDict[top:]
    s = 0
    for k,v in othersDict:
        s += len(v)

    x = ['p'+str(k) for k,v in graphDict] + ['other ports']
    y = [len(v) for k,v in graphDict] + [s]


    data = [go.Pie(labels=x,values=y,textinfo="label")]
    ploff.plot(data,filename=fileName)


def addToDict(packet, dport):
"""Agrega al diccionario de flujos por puerto destino"""
    if dport in flow_dict:
        flow_dict[dport].add(packet)
        return True
    flow_dict[dport] = {packet}
    return True


def processPcap(pcapFile, srcip):
"""Lee todos los paquetes de un determinado archivo pcap"""
    pcap = Reader(open(pcapFile,'r'))
    flows_list = []
    c1 = 0
    c2 = 0
    for timestamp, buff in pcap:
        c1 += 1
        if c1 == 100000 :
            c1 = 0
            c2 += 100000
            print '\t\t\t\t\t\t%s packets' % c2
```

```
        eth = Ethernet(buff)
        ip_pkt = eth.data
        tcp_sgmt = ip_pkt.data
        sip = inet_ntoa(ip_pkt.src)
        dip = inet_ntoa(ip_pkt.dst)
        sport = tcp_sgmt.sport
        dport = tcp_sgmt.dport
        if sip == srcip and dport in torrc_ports:
            f = Flow(sip,dip)
            addToDict(f, dport)
        else:
            continue

if __name__ == '__main__':
    opts = addOptions()
    checkOptions(opts)
    trafficFiles = getFiles(opts.ifile)
    for i in range(len(trafficFiles)):
        print '\t\t\t\t\t%s) Proccesing file %s' % (i+1, trafficFiles[i])
        packets = processPcap(trafficFiles[i], opts.srcip)
    if opts.prnt:
        printDict(flow_dict)
    if opts.graph:
        graphFlows(flow_dict, int(opts.graph), opts.ofile)
```

Visualización de conexiones hacia la red (World map)

Programa en Python que aprovecha los datos recabados mediante *Web Scrapping* para poder generar, con ayuda de las bases de datos públicas de localización de direcciones IP, un mapa del mundo estilo *chroropleth* de las conexiones hacia la red y los dominios consultados desde la red.

```
#!/usr/bin/python
import plotly.plotly as py
import pandas as pd
import sqlite3
from codes import *

country_dict = {}
con = sqlite3.connect('tor_real_ips.db')

#Lee todas aquellas direcciones que no son nodos de Tor
df = pd.read_sql_query("SELECT * FROM IPS WHERE tor_relay = 0", con)

for code in codes:
      country_dict[codes[code]] = 0

#Aumenta el conteo de IPs por país
for country in df['country3']:
      country_dict[country] += 1

#Crea gráfica cloropleth con los datos del diccionario
df2 = pd.DataFrame(country_dict.items(), columns=['Country','Visits'])
data = [ dict(
        type = 'choropleth',
        locations = df2['Country'],
        z = df2['Visits'],
        text = df2['Country'],
        colorscale = [[0,"rgb(5, 10, 172)"],[0.35,"rgb(40, 60,
190)"],[0.5,"rgb(70, 100, 245)"],\
            [0.6,"rgb(90, 120, 245)"],[0.95,"rgb(106, 137,
247)"],[1,"rgb(249, 249, 249)"]],
        autocolorscale = False,
        reversescale = True,
        marker = dict(
            line = dict (
                color = 'rgb(180,180,180)',
                width = 0.5
            ) ),
        colorbar = dict(
            title = 'Tor use by country - Entry relay'),
      ) ]
```

```
layout = dict(
    title = 'Tor use by country - Entry relay',
    geo = dict(
        showframe = False,
        showcoastlines = False,

        projection = dict(
            type = 'mercator'
        )
    )
)

fig = dict( data=data, layout=layout )
py.iplot( fig, validate=True, filename='tor-entry-map' )
```

Listado de nodos de entrada usados en cliente

Programa en Python que interactua con el servicio *tor* en una computadora para mostrar los nodos de entrada de todos los circuitos que ha armado dentro de la red Tor.

```
from stem import CircStatus
from stem.control import Controller
import socks
import socket
from urllib import urlopen
from datetime import datetime


with Controller.from_port(port = 9051) as controller:
#Se conecta al puerto de administracion de tor
    socks.set_default_proxy(socks.SOCKS5, "127.0.0.1", 9050)
    socket.socket = socks.socksocket
    controller.authenticate()

      #Muestra la direccion IP pública en uso
    my_ip = urlopen('http://ip.42.pl/raw').read()

    entries = ''
    for circ in sorted(controller.get_circuits()):
        if circ.status != CircStatus.BUILT:
            continue

        for i,entry in enumerate(circ.path):
            fingerprint, nickname = entry
            desc = controller.get_network_status(fingerprint,None)
            address = desc.address if desc else 'unknown'
            relay = circ.path[2][1]
            if my_ip in address:
                relay = relay + '**'

        entries = entries + ',' + relay
    time = datetime.utcnow().time() #Muestra momento de impresión
    print time, entries[1:]
```

Interacción con VirusTotal para dominios maliciosos

Interacción con la API de *VirusTotal* para obtener un análisis sobre actividad maliciosa con todos los nombres de dominio consultados por el nodo de salida.

```
#!/usr/bin/python
import sys
import optparse
from pickle import dump, load
from time import time, sleep
from requests import post
from math import ceil
import json
import urllib

def printError(msg, exit=True):
"""Imprime en el error estandar"""
    sys.stderr.write('Error:\t%s\n' % msg)
    if exit:
        sys.exit(1)

def checkOptions(opts):
"""Revisa que se pasen las opciones necesarias"""
    if opts.pklfile is None and opts.initialize is None:
        printError('You must indicate the txt ot pkl file to read')

def printDict(dictionary):
"""Imprime un diccionario de forma legible"""
    for key, value in dictionary:
        print "%s:\t%s" % (key, value)

def addOptions():
"""Agrega opciones para personalizar ejecución"""
    parser = optparse.OptionParser()
    parser.add_option('-i','--pkl-file',
                    dest='pklfile',
                    default=None,
                    help='Input file with dns queries')
    parser.add_option('-o','--output-file',
                    dest='ofile',
                    default='domain_db.pkl',
                    help='Output file when initializing db')
    parser.add_option('-t','--initialize-pickle',
                    dest='initialize',
                    default=None,
                    help='Receives a txt file with the domains to create
the pickle file that will be used')
    parser.add_option('-n','--write-n',
```

```
                    dest='write_n',
                    default=50,
                    help='Updates the pkl file n domains')
    parser.add_option('-v','--verbose',
                    dest='verbose',
                    default=False,
                    action='store_true',
                    help='Prints in standard output the execution
process')
    parser.add_option('-e','--report',
                    dest='report',
                    default=False,
                    action='store_true',
                    help='Prints in standard output the summary of
results')
    opts, args = parser.parse_args()
    return opts

def writeDB(ifile, pklfile, domains):
"""Escribe en la base de datos pkl"""
    try:
        if ifile:
            with open(ifile,'r') as i:
                domains = [[d[:-1],None,None] for d in i.readlines()]
        with open(pklfile,'wb') as o:
            dump(domains, o)
        return True
    except Exception as e:
        printError(e, exit=False)
        return False

def readPkl(ifile, verbose):
"""Lee en la base de datos para determinar dominios a procesar"""
    with open(ifile,'rb') as i:
        domains = load(i)
    if verbose:
        processed = len(filter(lambda d: d[1] is not None, domains))
        print '%s domains read \n%s domains submitted\n' % (len(domains),
processed)
    return domains


def isMalicious(domain, verbose):
"""Determina si es malicioso con base en los resultados de la API"""
    try:
        url = 'https://www.virustotal.com/vtapi/v2/domain/report'
```

```
        parameters = {'domain': domain, 'apikey': '<API KEY>'}
        response = urllib.urlopen('%s?%s' % (url,
urllib.urlencode(parameters))).read()
        json_response = json.loads(response)
        if json_response['response_code'] == 0:
            return (False, 'No information about the domain')
        if json_response['response_code'] == -1:
            return (False, 'Invalid domain')
        else:
            if 'detected_urls' not in json_response:
                return (False, 'URLs not in VirusTotal response')
            for url in json_response['detected_urls']:
                if url['positives'] > 0:
                    return (True, url['url'])
            return (False, 'No malicious URLs detected')
    except Exception as e:
        print e
        return (None, 'Unexpected error')


def printDomain(domain):
"""Imprime dominios e indica si es malicioso"""
    print '%s is %s (%s)' % (domain[0], 'MALICIOUS' if domain[1] else
'clean', domain[2])


def submitDomains(domains, pklfile, write_n, verbose):
"""Envia dominios tomando en cuenta el límite de peticiones por API"""
    write_n = int(write_n)
    writeCounter = 0
    virusTotalCounter = 0
    for d in domains:
        if d[1] is None:
            if virusTotalCounter == 0:
                startTime = time()
            d[1],d[2] = isMalicious(d[0], verbose)
            if verbose:
                printDomain(d)
            virusTotalCounter += 1
            writeCounter += 1

            if writeCounter == write_n:
                if verbose:
                    print '\n\tWriting changes in DB file\n'
                writeCounter = 0
                writeDB(None, pklfile, domains)
```

```
            if virusTotalCounter == 4:
                virusTotalCounter = 0
                elapsedTime = time() - startTime
                if elapsedTime < 60:
                    if verbose:
                        print '\n\tWaiting %s seconds\n' % (int(ceil(60 -
elapsedTime)))
                    sleep(int(ceil(60 - elapsedTime)))

def reportResults(domains):
"""Escribe los resultados de la ejecución"""
    print '\n%s domains loaded' % (len(domains) - 1)
    processed = len(filter(lambda d: d[1] is not None, domains))
    print '\n%s domains submitted' % processed
    malicious = filter(lambda d: d[1] == True, domains)
    print '\n%s domains have at least one malicious URL detected:\n' %
len(malicious)
    for d in malicious:
        print ' - %s' % d[2]

if __name__ == '__main__':
"""Ejecución principal"""
    opts = addOptions()
    checkOptions(opts)
    if opts.initialize:
        writeDB(opts.initialize, opts.ofile, [])
    else:
        domains = readPkl(opts.pklfile, opts.verbose)
        if opts.report:
             reportResults(domains)
        else:
            submitDomains(domains, opts.pklfile, opts.write_n,
opts.verbose)
```

Análisis de origen de dominio (TLD y Whois)

Análisis de dominios considerando el TLD de país y los registros WHOIS de todos los nombres de dominio consultados por el nodo de salida, para poder generar un mapa de estilo *chroropleth*.

```
#!/usr/bin/python
from codes import codes
from whois import whois
from sqlite3 import connect
from time import sleep
import optparse
import sys


def printError(msg,exit=True):
"""Imprime en el error estandar"""
    sys.stderr.write('Error:\t%s\n'%msg)
    if exit:
        sys.exit(1)


def getSld(domain):
"""Obtiene el second-level domain de un dominio"""
    split_domain = domain.split('.')
    return split_domain[-2] + '.' + split_domain[-1]


def checkOptions(opts):
"""Revisa las opciones necesarias"""
    if opts.ifile is None:
        printError('You must indicate the connections file to read')
    if opts.dbfile is None:
        printError('You must indicate the database file to read')


def addOptions():
"""Agrega opciones para personalizar ejecución de programa"""
    parser=optparse.OptionParser()
    parser.add_option('-i','--input-file',
        dest='ifile',
        help='File with list of domain names')
    parser.add_option('-d','--db-file',
        dest='dbfile',
        help='Previously created SQLite database file')
    parser.add_option('-s','--sleep',
        dest='sleep',
        default='2',
        help='Sleep time in seconds between whois requests')
    parser.add_option('-c','--commit',
        dest='commit',
        default='50',
```

```
        help='Number of insertions before a commit is done in database')
    opts,args=parser.parse_args()
    return opts



def readDomains(input_file):
"""Crea listado de dominios"""
    with open(input_file,'r') as domainFile:
        domains = [i[:-1] for i in domainFile.readlines() if '.' in i]
    return domains

def isInDB(dbcursor,domain):
"""Revisa si determinado dominio ya se revisó anteriormente"""
    query="""SELECT * FROM DOMAINS WHERE full_domain='%s';""" % (domain)
    dbcursor.execute(query)
    rows=dbcursor.fetchall()
    if rows==[]:
        return False
    return True


def fillDb(domains, dbfile, query_sleep, commit):
"""Llena la base de datos con los datos de todos los dominios"""
    con = connect(dbfile)
    with con:
        cur = con.cursor()
        n_commit = 0
        counter = 0
        for domain in domains:
            counter += 1
            print counter, n_commit, domain
            if isInDB(cur, domain):
                continue
            try:
                country = domain.split('.')[-1].upper()
                if country in codes:
                    query = """INSERT OR IGNORE INTO
DOMAINS(full_domain,sld_domain,country2,country3) VALUES
('%s','%s','%s','%s');""" %
(domain,getSld(domain),country,codes[country])
                    cur.execute(query)
                else:
                    sleep(float(query_sleep))
                    whois_res = whois(getSld(domain))
                    country = whois_res['country']
```

```
                    query = """INSERT OR IGNORE INTO
DOMAINS(full_domain,sld_domain,country2,country3) VALUES
('%s','%s','%s','%s');""" %
(domain,getSld(domain),country,codes[country])
                    cur.execute(query)
                n_commit += 1
            except Exception as e:
                query = """INSERT OR IGNORE INTO
DOMAINS(full_domain,sld_domain,country2,country3) VALUES
('%s','%s','UNKNOWN','UNKNOWN');""" % (domain,getSld(domain))
                cur.execute(query)
                n_commit += 1
            finally:
                if n_commit == int(commit):
                    n_commit = 0
                    con.commit()



if __name__ == '__main__':
"""Ejecución principal del programa"""
    opts = addOptions()
    checkOptions(opts)
    domains = readDomains(opts.ifile)
    fillDb(domains, opts.dbfile, opts.sleep, opts.commit)
```