

Agosto 2021



**Descubrimiento de infraestructuras de
Hosting de dominios onion**

Alfredo Abarca

Indice

Índice.....	2
Descripción.....	3
Estado del Arte.....	4
Motivación.....	5
Hipótesis.....	7
Tesis.....	7
Desarrollo.....	8
Requerimientos Básicos.....	9
Preparación del Ambiente.....	9
Tor Browser.....	10
Onion Search.....	10
ProxyChains4.....	11
NMAP.....	11
Splash.....	12
Docker.....	13
Page-Compare.....	16
Privoxy.....	17
Curl.....	18
Nmap-to-CSV	18
Recopilando Información.....	19
Scraping de los sitios Onion.....	24
Análisis de la información.....	31
¿Cómo relacionar todos los Dominios Onion...?	42
¿Cómo se relacionan todos estos dominios?.....	47
¿Qué tipo de contenido observamos en estos dominios?.....	48
Conclusiones.....	49
Anexos.....	50
Referencias.....	53

1. Descripción

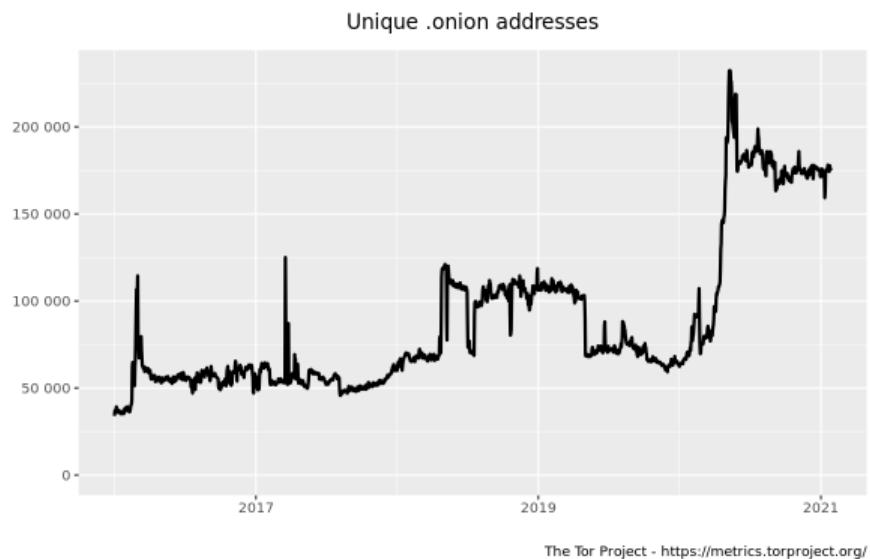
Tomando como punto de partida la información proporcionada por el proyecto Tor Project acerca de los llamados “Tor-Friendly”, todas las investigaciones realizadas por el equipo de onionscan y el Dark Web Map se solicita:

- Obtención de todos los dominios onion posibles haciendo uso de la información proporcionada durante el módulo 5 del Master para la creación de un set de dominios onion en una fecha concreta.
- Diferenciar los distintos tipos de servicios desplegados en los dominios onion (http, https, ftp, smtp, otros).
- Realizar una actualización de los reportes sobre infraestructuras que albergan dominios onion que realizo el personal de onionscan en 2016.
- Investigar como relacionar todos los dominios onion obtenidos con las infraestructuras utilizadas para su despliegue.
- En base a una infraestructura que alberga una cierta cantidad de dominios onion, investigar cómo es posible desanonomizar toda la infraestructura en base a la vulneración de uno de los dominios onion que están hosteados en la infraestructura.
- Creación de mapa que relacione todos los dominios onion obtenidos en las infraestructuras que los albergan y si estos dominios onion contienen temática delictiva.

2. Estado del arte

2.1 Antecedentes de la investigación

Durante los últimos años se ha el aumento de sitios publicados en lo que se conoce como la red “Tor” se ha incrementado significativamente, ya que a través del sitio **metrics.torproject.org** podemos observar una clara tendencia al incremento año con año. Sin embargo, algo de notar es que durante el 2020 el número de estos sitios prácticamente se duplicó de un número alrededor de los 80,000 sitios a prácticamente rondar entre los 160,000 y 170,000 sitios como lo muestran las últimas estadísticas.



Y no es de sorprender tampoco el hecho de que los temas relacionados de la actual pandemia han contribuido al crecimiento desmedido de la publicación de estos sitios en la red oculta de internet, y es que a la par, la actividad en línea se ha aumentado también para realizar actividades cuestionables en ésta red.

Hay que considerar que la red de TOR ha sido empleada principalmente para llevar a cabo actividades no legales que de otra forma no podrían realizarse en la llamada internet superficial debido a que ésta red proporciona un cierto nivel de temporalidad efímera y anonimato a sus usuarios, complicando las actividades de los cuerpos policiacos o de procuración de justicia para la captura de las personas que están detrás del teclado.

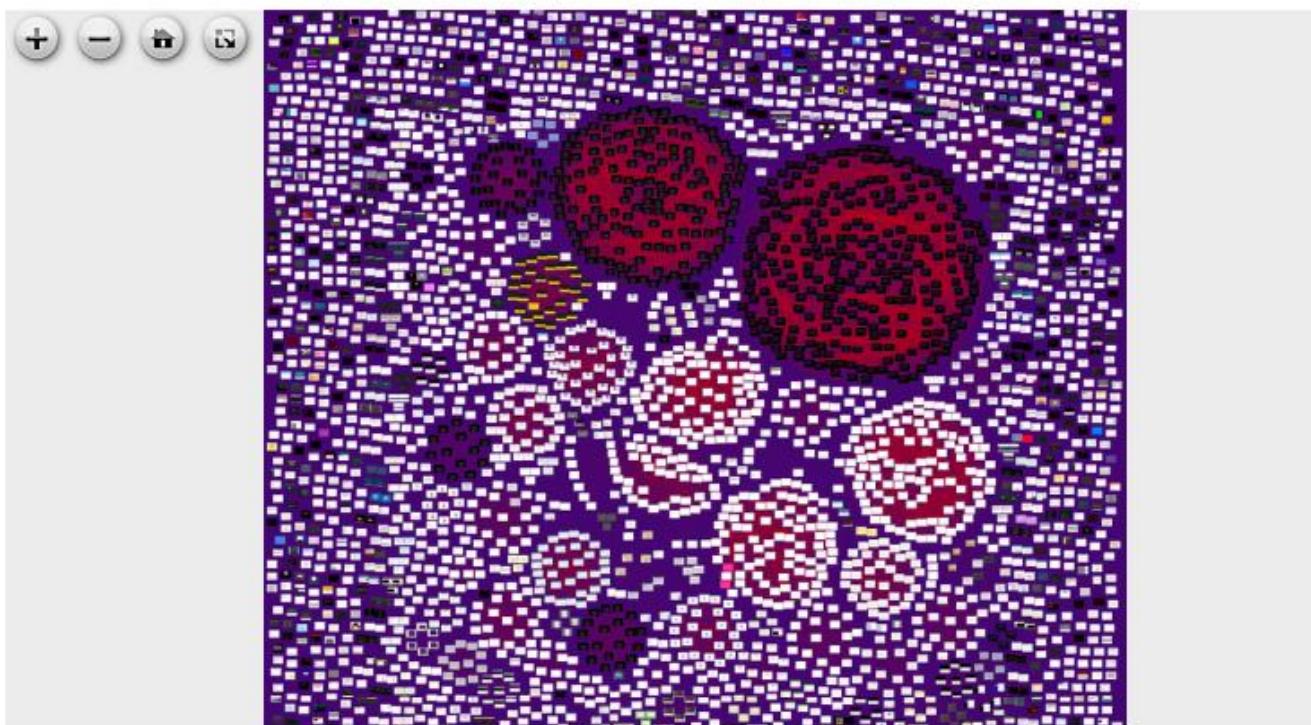
Es por eso que en los últimos años han surgido diversos esfuerzos para que por un lado se pueda realizar un mapeo o dibujo por llamarlo de alguna forma de todos los servicios e infraestructura detrás de los mismos de la red “Tor”, teniendo en ocasiones un acercamiento bastante aproximado sin poder aún realizarlo en su totalidad ya sea porque estos servicios tienen una temporalidad de vida muy corta y adicional a ello por que como es claro, se generan nuevos servicios cada día.

Uno de los proyectos más reconocidos en el trabajo o esfuerzo de realizar un mapeo de estos servicios es el que está bajo el nombre de “**Dark Web Map**” de la empresa Hyperion Gray (<https://www.hyperiongray.com/dark-web-map>), la cual en 2019 realizó un ejercicio de mostrar de manera gráfica en una red de nodos la relación de 3.7 k sitios dentro de la red de TOR pero con la limitante de sólo mostrar la página de inicio relacionada a cada dominio del cual realizaron el descubrimiento.

El mapa en cuestión se muestra en la siguiente imagen a gran escala, sin embargo, si se visita el sitio se puede visualizar con gran detalle el contenido de cada uno de los sitios que ellos lograron relacionar.

Map

Click and drag to pan. Use scroll wheel or +/- buttons to zoom.



Ref: Mapa de Dark Web, por Hyperion Gray (<https://www.hyperiongray.com/dark-web-map/>)

Durante el 2020 la evolución de este proyecto ha ido en función de mostrar las imágenes del mapa hasta cierto punto “censuradas” ya que el contenido que se podía observar en la primera versión de su mapa podría mostrar contenido no apto o con actos de violencia. Por tal motivo durante el 2020 las mejoras al proyecto iban en función de evitar mostrar este contenido, así como las direcciones hacia los sitios para evitar fomentar que por curiosidad o intencionalmente se navegara con más facilidad a sitios de los cuales por desgracia se encuentran en línea. Ya retomaremos parte de este proyecto más adelante cuando entremos en los temas técnicos de la investigación.

Sin embargo, cabe mencionar que el alcance de éste proyecto, es la de relacionar los sitios web publicados de 3.7 k dominios *.onion con base al contenido de la página de inicio publicada y no está dentro de su alcance otros posibles servicios publicados en esta red como pudieran ser: Servidores de correo, FTP, SSH, IRC, etc... por lo que, si bien es un buen punto de inicio, necesitábamos ubicar otros proyectos que considerarán examinar los otros tipos de servicios publicados en la red de TOR.

A este respecto existe otro proyecto bajo el nombre de **Onionscan** (<https://onionscan.org/>) que no es otra cosa más que una herramienta publicada bajo licencia libre que permite de manera similar a lo realizado por Hyperion Gray realizar un mapa de investigación y análisis de los nodos y servicios que se encuentran en la Dark Web.

Este proyecto a diferencia del mencionado anteriormente de “Dark Web Map” deja ver un poco más allá de sólo la página de inicio de los dominios y su relación, si no, que también permite identificar las tecnologías y servicios comúnmente empleados por los servidores en la red de TOR para publicar otros servicios como FTP, SSH, Servidores de correo entre muchos otros. El alcance de éste análisis comprende 5000 servidores analizados en mayo de 2016, arrojando los siguientes resultados:

- La tecnología más empleada para publicar un servicio web son los servidores basados en Apache encontrados en al menos la mitad de la muestra analizada, seguida por nginx en un 30% y el resto entre otras tecnologías.
- La plataforma de gestión de contenido es Wordpress misma que se pudo identificar en un aproximado de un 2% de la muestra. Otras tecnologías empleadas son Drupal, Media Wiki, Joomla, twiki, entre otras.

Más información: <https://mascherari.press/onionscan-report-may-2016-technologies-used-by-onion-services/>

Existen otros proyectos y desarrollos que han hecho interesantes esfuerzos para realizar un descubrimiento o crawling de los sitios e infraestructura de la Dark web, sin embargo, lo que ubico como más alineados al propósito de ésta práctica, son básicamente los 2 mencionados anteriormente y en los cuales se basará el desarrollo de esta investigación.

Así mismo es importante mencionar algunas de las operaciones que se han realizado de forma reciente a través de la red TOR y que han concluido en la desactivación de redes criminales y de tecnología que empleaban este medio para fines ilícitos como los que cito a continuación:

- **Desconexión del sitio Dark Market (13/01/2021)**

A principios de este año, salió una noticia relacionada a la desactivación de uno de los sitios más grandes en cuanto a comercio ilegal se refiere, bajo el nombre de **Dark Market**, mismo que, de acuerdo con datos de Europol, tenía cerca de 500 Millones de usuarios y poco más de 2,400 vendedores ofertando todo tipo de productos, mismos que van desde medicamentos, dinero falso, detalles de tarjetas de crédito, malware, SIMs anónimas, entre otro tipo de productos ilegales.

En los medios de noticias, se informa que esta operación pudo ser posible por medio de la colaboración de varios países y a la ubicación de actividades irregulares en un servicio de hosting, lo que pudo lograr la ubicación de 20 servidores en donde se alojaba este servicio.

<https://www.eluniversal.com.mx/techbit/desmantelan-el-mercado-ilegal-mas-grande-en-la-dark-web>

- **Operación DISRUPTOR (23/09/2020)**

A través de la operación bajo el nombre clave DISRUPTOR, Estados Unidos en colaboración con la Europol y otros 8 países lograron cerrar el sitio de tráfico de drogas bajo el nombre de Wall Street Market, en la cual fue posible el arresto de 179 personas entre las que se encontraban los administradores de este sitio y otros 2 vendedores que eran identificados como los más populares dentro del mismo.

Gran parte del éxito de esta operación se basó en que dentro de la evidencia confiscada se pudo tener acceso a uno de los servidores que actuaba como respaldo de este sitio, identificando con ello a varios usuarios y traficantes que hacían uso del mismo.

En este caso la Europol emitió un mensaje a los usuarios de la Dark Web en el cual se indica que la red oscura ya no lo es más, puesto que pueden encontrar a las personas que están detrás de servidores y sitios que se encuentran alojados en esa red.

"La edad de oro del mercado de la Dark Web ha terminado. Operaciones como estas destacan la capacidad de las fuerzas del orden para contrarrestar el cifrado y el anonimato de los mercados de la Dark Web."

<https://www.fbi.gov/news/stories/operation-disruptor-jcode-shuts-down-darknet-drug-vendor-092220>

3. Motivación

La principal motivación para realizar el desarrollo de este proyecto es la de profundizar más allá del alcance de los estudios y análisis mencionados anteriormente y con ello aportar mayor visibilidad en éste ecosistema aún por explorar en su totalidad y que sin duda también puede ser útil para plantear otra posibilidad de identificar infraestructuras que soportan sitios con actividad ilegal y lograr con ello la posible desarticulación de la misma.

Como analista de Ciberinteligencia, es claro que la Deep y Dark web son sitios que son una fuente muy valiosa al momento de realizar cualquier investigación y el hecho de poder tener un plano claro del tipo de sitios y contenido al cual se tiene acceso es indispensable al momento de enfrentarse investigaciones relacionadas con actividades ilícitas.

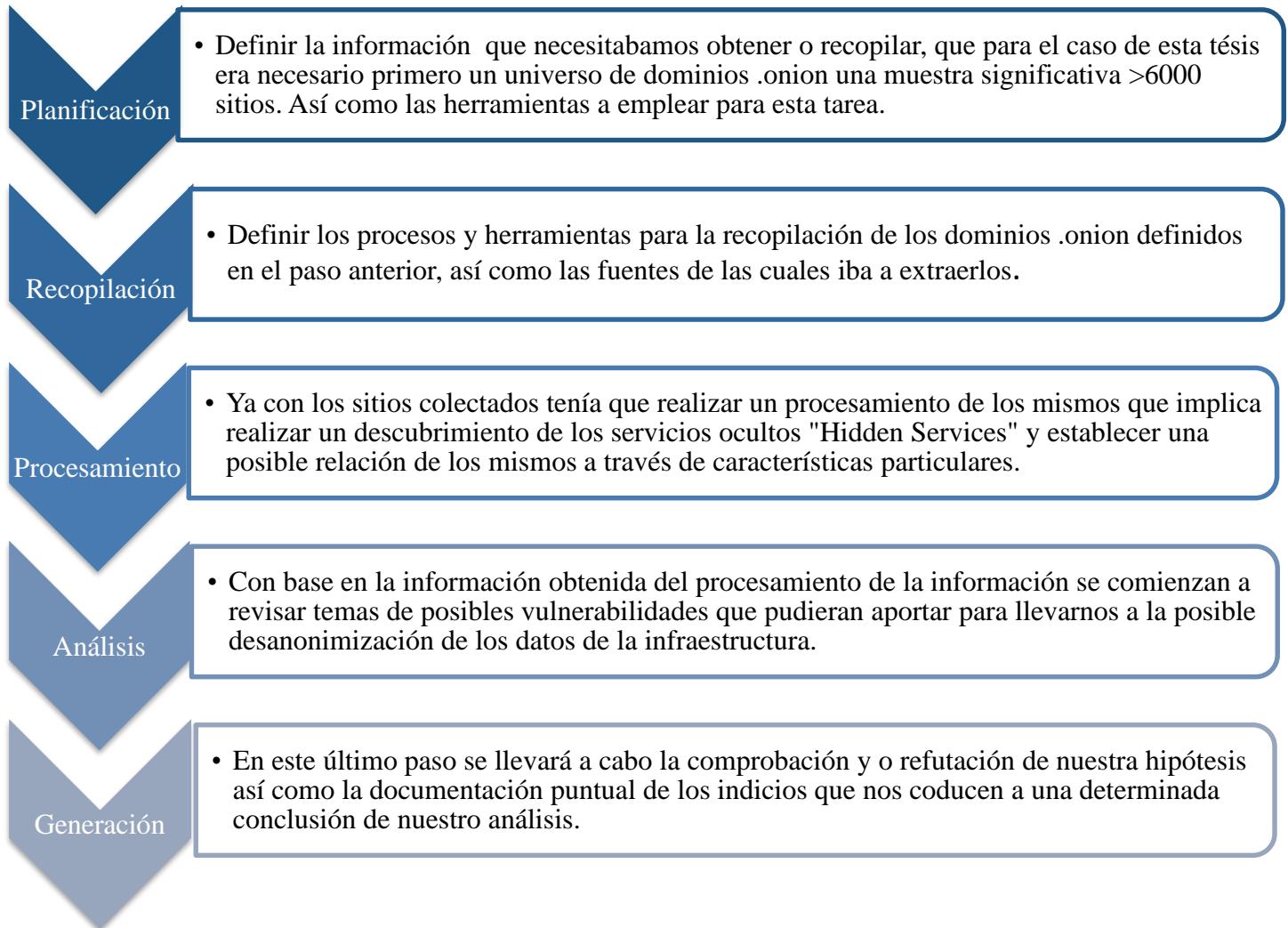
4. Hipótesis

¿Es posible que a través del descubrimiento y vulneración de servicios publicados en la red de TOR se pueda desanonomizar la infraestructura detrás del mismo?

R= Es viable que a través del descubrimiento de las tecnologías y explotando las vulnerabilidades asociadas a las mismas, se pueda desanonomizar parte o la totalidad de la infraestructura que soporta un servicio determinado.

5. Tesis

Para llevar a cabo el desarrollo de la presente tesis decidí ajustar los pasos de la misma adecuados a los procesos básicos del ciclo de inteligencia como se muestra a continuación:



6. Desarrollo

A continuación, se describen todos los pasos que técnicamente tuve que realizar para llevar a cabo el desarrollo de los pasos descritos en la página anterior:

6.1 Requerimientos Básicos

Para los procesos de recopilación y procesamiento de información fue necesario generar una máquina virtual de **Ubuntu 20.04** en su versión de escritorio y con todas las actualizaciones al momento de la elaboración de este documento.

Para la descarga el ISO de instalación de este software

<https://releases.ubuntu.com/20.04/>

Tras instalar el sistema operativo recomiendo aplicar todas las actualizaciones tanto de las utilerías como del mismo sistema a través de la ejecución de los siguientes comandos:

```
#sudo apt-get update  
#sudo apt-get upgrade
```

Adicional a la actualización de las utilerías necesitamos también instalar los siguientes componentes del sistema para poder realizar la instalación de otras herramientas posteriormente.

```
#sudo apt-get install python3-pip  
#sudo apt install git
```

6.1.1 PREPARACIÓN DEL AMBIENTE - (ETAPA DE PLANEACIÓN)

Tras haber instalado el sistema operativo anteriormente mencionado, una de las primeras utilerías que tendremos que instalar es el servicio de TOR para poder conectarnos a los servicios de dicha red. Para realizarlo tendremos que teclear el siguiente comando en una ventana de terminal:

```
#sudo apt-get install tor
```

Con el comando anterior y si lo ejecutó de forma adecuada deberemos tener instalado el servicio de tor en nuestro equipo, para validar que efectivamente lo tenemos instalado, podemos correr el siguiente comando:

```
#tor --version
```

Con el cual deberemos ver la versión del servicio que tenemos instalado.

```
digger@tordigger:~$ tor --version
Tor version 0.4.2.7.
digger@tordigger:~$
```

Tras haberlo instalado y verificar la versión del mismo. Aún no iniciare el servicio puesto que faltan aún componentes que descargar de la red, de tal manera que si lo dejamos conectado a la red de TOR pudieran no descargarse y/o instalarse de forma correcta.

➤ TORBROWSER

Otra de las herramientas que necesitaremos para poder trabajar con este escenario es el navegador de TOR, y no necesariamente por el browser, si no por los componentes y librerías empleadas por otras herramientas que se instalarán más adelante.

Para instalar ésta herramienta será necesario ejecutar los siguientes comandos:

```
#sudo add-apt-repository ppa:micahflee/ppa
#sudo apt-get update
#sudo apt install torbrowser-launcher
```

➤ ONIONSEARCH

La siguiente herramienta a descargar se llama **OnionSearch** que es la herramienta que nos ayudará en el proceso de descubrimiento y recolección de los dominios con extensión *.onion sin mayor complejidad o realizarlo de forma manual.

Para descargar la herramienta y realizar la instalación se deberán teclear los siguientes comandos:

```
#git clone https://github.com/megadose/OnionSearch.git
#cd OnionSearch/
#sudo python3 setup.py install
```

Tras ejecutar los comandos y si no nos indica algún error durante la instalación podremos ejecutar el siguiente comando con los siguientes resultados:

```
digger@diggervm2:~/OnionSearch$ onionsearch
usage: onionsearch [-h] [--proxy PROXY] [--output OUTPUT]
                   [--continuous_write CONTINUOUS_WRITE] [--limit LIMIT]
                   [--engines [ENGINES [ENGINES ...]]]
                   [--exclude [EXCLUDE [EXCLUDE ...]]]
                   [--fields [FIELDS [FIELDS ...]]]
                   [--field_delimiter FIELD_DELIMITER] [--mp_units MP_UNITS]
                   search
```

Con lo anterior corroboramos que nuestra herramienta está correctamente instalada, y podremos continuar con la siguiente herramienta que nos permitirá realizar los escaneos posteriores a la recopilación de información de todos los dominios .onion que veremos mas adelante.

➤ PROXYCHAINS4

La utilería de ProxyChains4 nos permitirá enrutar todo el tráfico de las utilerías que ejecutemos en linea de comandos a través de la red TOR, con lo cual podremos emplear herramientas como **nmap** para llevar a cabo los escaneos posteriores destinados al descubrimiento de los servicios ocultos o “hidden services” de cada uno de los dominios .onion que recabemos a través de la herramienta de **OnionSearch**.

Para instalar esta herramienta basta con ejecutar el siguiente comando:

```
#sudo apt-get install proxychains4
```

Si el comando anterior no marca ningún error deberemos poder teclear el comando proxychains4 mostrándonos algo de la siguiente forma:

```
digger@diggervm2:~$ proxychains4
Usage: proxychains4 -q -f config_file program_name [arguments]
      -q makes proxychains quiet - this overrides the config setting
      -f allows one to manually specify a configfile to use
          for example : proxychains telnet somehost.com
More help in README file
```

➤ NMAP

La siguiente utilería a instalar es nmap, ya que esta es la que nos ayudará a realizar el escaneo de puertos en cada uno de los dominios .onion que logremos recabar, para realizar la instalación basta con teclear el siguiente comando:

```
#sudo apt-get install nmap
```

Al igual que las otras herramientas, si lo instalamos correctamente deberemos ver un texto similar al ejecutar el comando:

```
digger@tordigger:~/OnionSearch$ nmap
Nmap 7.80 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
```

Si aparece este texto, es que todo hasta el momento se ha configurado correctamente, podremos continuar con la instalación de otras herramientas, necesarias para este escenario.

➤ SPLASH

Splash es una herramienta que nos permitirá extraer el contenido web (HTML) de una página determinada así como un screenshot de la misma, esta utilería la ocuparemos para poder comparar las similitudes de las páginas que existen en los dominios *.onion que recabemos y a partir de ahí, comenzar a establecer posibles relaciones con base a contenido de los sitios.

Splash en su versión mas reciente se encuentra disponible en un contenedor de docker, por lo que deberemos instalar en primer lugar el manejador de docker para posteriormente instalar ésta herramienta.

Para realizar la instalación de docker en el equipo virtual, ejecutamos los siguientes comandos:

```
#sudo apt-get install docker.io
```

Posterior a la instalación de docker, deberemos iniciarizar dicho servicio a través del siguiente comando:

```
#sudo systemctl enable --now docker
```

Para verificar que versión de docker acabamos de instalar ejecutamos el siguiente comando:

```
#docker --version
```

Y deberemos ver algo similar a lo siguiente:

```
digger@tordigger:~/OnionSearch$ docker --version
Docker version 19.03.8, build afacb8b7f0
digger@tordigger:~/OnionSearch$
```

De igual manera podemos ejecutar esta otra instrucción para verificar que el servicio se encuentra activo y ejecutándose de forma correcta.

```
#sudo docker run hello-world
```

```
digger@tordigger:~/OnionSearch$ sudo docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

Si hasta este punto ha ido todo bien, podemos ahora si tratar de instalar la utilería de splash en el contenedor de docker recién instalado.

Para instalar el contenedor de splash ejecutamos la siguiente línea de comandos:

```
#sudo docker pull scrapinghub/splash
```

Aquí tardará algunos minutos en lo que se descarga y configura nuestro contenedor, por lo que deberemos esperar a que el proceso termine en su totalidad.

```
digger@tordigger:~/OnionSearch$ sudo docker pull scrapinghub/splash
Using default tag: latest
latest: Pulling from scrapinghub/splash
7595c8c21622: Pull complete
d13af8ca898f: Pull complete
70799171ddba: Pull complete
b6c12202c5ef: Pull complete
60a588d4f36e: Pull complete
74efcc44bb0a: Pull complete
630a03095961: Pull complete
6ece3b427ef5: Pull complete
a1fce8353093: Pull complete
3933ca5f4768: Retrying in 1 second
b0097a197686: Retrying in 1 second
eb38ff0231d8: Retrying in 1 second
3d065f0b97ea: Waiting
bce087e2552e: Waiting
```

Aquí tardará algunos minutos en lo que se descarga y configura nuestro contenedor, por lo que deberemos esperar a que el proceso termine en su totalidad.

Si todo funcionó correctamente se deberá ejecutar la siguiente linea de comandos para verificar que el contenedor está correctamente instalado y funcionando correctamente.

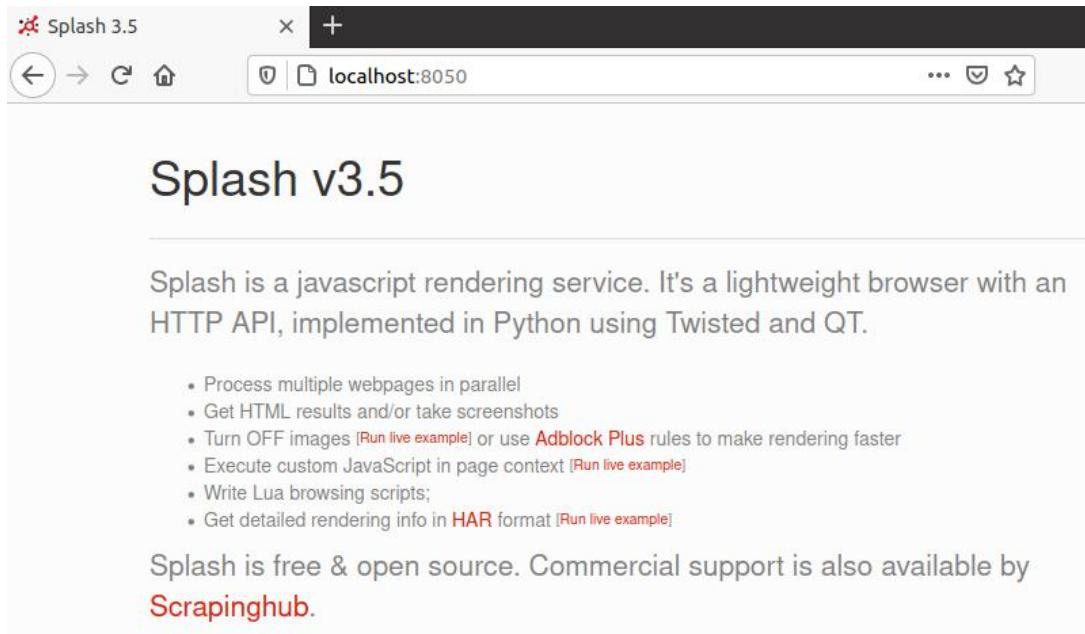
```
#sudo docker run -it -p 8050:8050 --rm scrapinghub/splash
```

NOTA: Es probable que en un primer ejercicio nos marque error dado a que no se ha configurado bien en una primera ejecución, por lo que si marca error en una pimer ejecución, simplemente vuelva a ejecutar el comando, hasta obtener el resultado como el que se muestra:

```
digger@tordigger:~$ sudo docker run -it -p 8050:8050 --rm scrapinghub/splash
2021-01-30 12:09:13+0000 [-] Log opened.
2021-01-30 12:09:13.565497 [-] Xvfb is started: ['Xvfb', ':2084675407', '-screen', '0', '1024x768x24', '-nolisten', 'tcp']
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-splash'
2021-01-30 12:09:13.678378 [-] Splash version: 3.5
2021-01-30 12:09:13.709842 [-] Qt 5.14.1, PyQt 5.14.2, WebKit 602.1, Chromium 7
7.0.3865.129, sip 4.19.22, Twisted 19.7.0, Lua 5.2
2021-01-30 12:09:13.709975 [-] Python 3.6.9 (default, Jul 17 2020, 12:50:27) [GCC 8.4.0]
2021-01-30 12:09:13.710266 [-] Open files limit: 1048576
2021-01-30 12:09:13.710351 [-] Can't bump open files limit
2021-01-30 12:09:13.724411 [-] proxy profiles support is enabled, proxy profiles path: /etc/splash/proxy-profiles
2021-01-30 12:09:13.724735 [-] memory cache: enabled, private mode: enabled, js cross-domain access: disabled
2021-01-30 12:09:13.814913 [-] verbosity=1, slots=20, argument_cache_max_entries=500, max-timeout=90.0
2021-01-30 12:09:13.815264 [-] Web UI: enabled, Lua: enabled (sandbox: enabled), Webkit: enabled, Chromium: enabled
2021-01-30 12:09:13.815636 [-] Site starting on 8050
2021-01-30 12:09:13.815695 [-] Starting factory <twisted.web.server.Site object at 0x7fb2182825c0>
2021-01-30 12:09:13.816018 [-] Server listening on http://0.0.0.0:8050
```

Si obtenemos este mensaje el contenedor de docker nos indica que ya se encuentra escuchando peticiones en el puerto 8050 de nuestro equipo local.

Para verificarlo, basta con abrir un navegador y poner en el navegador <http://localhost:8050> nos deberá desplegar la siguiente página web:



The screenshot shows the Splash 3.5 web interface. At the top, there's a header bar with icons for refresh, back, forward, and search, followed by the text "localhost:8050". Below the header is a main content area with a title "Splash v3.5". A descriptive paragraph follows: "Splash is a javascript rendering service. It's a lightweight browser with an HTTP API, implemented in Python using Twisted and QT." To the right of this text is a large blue watermark-like logo. Below the paragraph is a bulleted list of features:

- Process multiple webpages in parallel
- Get HTML results and/or take screenshots
- Turn OFF images [Run live example] or use **Adblock Plus** rules to make rendering faster
- Execute custom JavaScript in page context [Run live example]
- Write Lua browsing scripts;
- Get detailed rendering info in **HAR** format [Run live example]

At the bottom of the content area, there's a note: "Splash is free & open source. Commercial support is also available by [Scrapinghub](#)".

NOTA: Si hemos llegado a este punto quiere decir que hemos instalado todo correctamente, aunque aún hay algunas configuraciones adicionales por hacer, pero ya falta poco para que nuestro entorno quede completamente listo para comenzar con la recopilación de información.

El siguiente paso es configurar nuestro contenedor para que el tráfico de navegación pueda salir a través de la red de TOR, para lo cual necesitamos ejecutar los siguientes comandos para crear un perfil de proxy para desviar todo el tráfico por el puerto 9050 de nuestro cliente de TOR.

```
#mkdir /etc/splash  
#mkdir/etc/splash/proxy-profiles  
#cd /etc/splash/proxy-profiles
```

Una vez generado esa ruta de directorios abrir un editor de textos, como por ejemplo VI y agregar el siguiente contenido en el archivo bajo el nombre **default.ini**

```
[proxy]  
  
host=127.0.0.1  
port=9050  
type=SOCKS5
```

Ya que hemos generado este archivo, deberemos ejecutar la siguiente linea de comandos para volver a iniciar el contenedor de docker pero redireccionando la salida por medio de la red de TOR.

```
#sudo docker run -p 8050:8050 -v /etc/splash/proxy-profiles:/etc/splash/proxy-profiles --net="host"  
scrapinghub/splash --proxy-profiles-path=/etc/splash/proxy-profiles --max-timeout 300
```

Si no marca ningún error al momento de ejecutar puede intentar nuevamente abrir un navegador e intentar hacer el scraping de un sitio de la red de tor, por ejemplo, podría tratar con el siguiente: <http://2gow4dgzxcjyzb3u.onion/>

Al ingresar este valor en la caja de búsqueda de splash, debería ver una imagen similar a la que se muestra a continuación.

Con lo cual podemos verificar que nuestra instancia de Splash ya puede resolver los dominios de tipo *.onion.

NOTA: Es importante que considere que los dominios de la red tor son volátiles, por lo que el sitio que puse como ejemplo pudiera no estar activo, por lo que se recomienda igual probar con alguno.



➤ PAGE-COMPARE

Otra utilería que instalaremos para este escenario es la que se llama page-compare, esta utilería lo que nos permitirá es comparar las similitudes de código HTML de las páginas que han sido scrapeadas o analizadas por medio de la utilería de **SPLASH**, misma que instalamos en el paso anterior.

Para poder instalar la utilería de page-compare, es necesario ejecutar los siguientes comandos en una ventana de terminal.

```
#git clone https://github.com/TeamHG-Memex/page-compare.git  
#cd page-compare  
#pip3 install lxml
```

Si todo ha ido bien, solo resta ejecutar el siguiente comando para validar que la utilería se ejecutó de forma correcta y no hace falta ninguna dependencia.

```
#python3 scrape.py
```

Deberá mostrar un mensaje como el siguiente:

```
tigger@tordigger:~/page-compare$ python3 scrape.py
Usage: scrape.py <splash url> <sites JSON> <run number> <output path>
tigger@tordigger:~/page-compare$
```

➤ PRIVOXY

Otra de las herramientas que emplee para el proceso de recopilación de información fue la conocida como Privoxy, esta herramienta me permitirá redireccionar el tráfico http, https, socks por medio de un puerto interno hacia la red de TOR, para poder automatizar parte del procesamiento de información acerca de las tecnologías empleadas para hospedar las páginas que se visitan en la red de TOR, para realizar su instalación es necesario ejecutar la siguiente linea de comandos.

```
# sudo apt-get install privoxy
```

Al ejecutar la siguiente linea de comando, sólo instalamos el servicio, hace falta aún configurar el tipo de tráfico que estará enruteando para lo cual será necesario editar el archivo **/etc/privoxy/config**. Para indicar que el protocolo socks5 se forwardeará al puerto 9050, que es el mismo que nuestro servicio de TOR.

Simplemente necesitamos descomentar la siguiente linea:

```
#      HTTP parent looks like this:
#
#      forward-socks4    /          socks-gw.example.com:1080 .
#
# To chain Privoxy and Tor, both running on the same system, you
# would use something like:
#
#      forward-socks5t   /          127.0.0.1:9050 .
#
# Note that if you got Tor through one of the bundles, you may
# have to change the port from 9050 to 9150 (or even another
# one). For details, please check the documentation on the Tor
# website.
#
# The public Tor network can't be used to reach your local
```

Tras haber realizado este cambio, basta con que reiniciemos el servicio para que tome esta nueva configuración, mismo que podemos realizar con el siguiente comando:

```
#sudo /etc/init.d/privoxy restart
```

Ahora bien, para asegurar que el tráfico o solicitudes que nosotros realicemos desde una ventana de terminal se vayan por ese servicio, tendremos que especificar en una variable de entorno, que nuestro proxy será el servicio de privoxy, lo cual podemos realizar a través de los siguientes comandos:

```
#export http_proxy="127.0.0.1:8118"
#export https_proxy="127.0.0.1:8118"
```

Y ya con lo anterior podremos automatizar parte importante del proceso de recopilación de información referente a la versión de SW de los manejadores que se encuentran detrás de las páginas web

➤ CURL

Otra de las herramientas necesarias para la actividad del proceso de recopilación de información es la herramienta de CURL, esta herramienta me permite hacer peticiones web desde la linea de comandos, misma que me da una salida en formato de texto de la respuesta del servidor.

Para realizar la instalación sólo es necesario ejecutar el siguiente comando:

```
#sudo apt-get install curl
```

Para validar que se encuentra correctamente instalada, podemos ejecutar el siguiente comando, para verificar la versión del curl que acabamos de instalar.

```
#curl -V
```

Deberemos ver una salida de información similar a la que se muestra a continuación:

```
digger@diggervm2:~$ curl -V
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib
nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s
    rtmp rtsp scp sftp smb smbs smtp smtpts telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile
    libz NTLM NTLM_WB PSL SPNEGO SSL TLS-SRP UnixSockets
```

➤ Nmap-to-CSV

Esta utilería va a transformar la salida de nmap en XML a CSV, esto nos va a ayudar a procesar fácilmente la información que obtengamos de los escaneos de puertos hacia una hoja de CSV, para instalar esta utilería solamente tenemos que ejecutar el siguiente comando.

```
#git clone https://github.com/laconicwolf/Nmap-Scan-to-CSV.git
```

Con este comando habremos descargado el programa que nos ayudará a convertir la salida en XML de los resultados del escaneo a formato CSV.

Ahora si, con éste último paso tenemos configurado todo el entorno para comenzar con el proceso de recopilación de la información.

6.1.2 RECOPILANDO INFORMACIÓN – (ETAPA DE RECOPILACIÓN)

Una vez preparado el entorno, vamos encaminandonos a uno de los primeros puntos que nos deja como reto esta práctica, que es la creación de un repositorio de dominios .onion para una fecha determinada, para lo cual emplearemos una de las herramientas que instalamos en la fase anterior **OnionSearch**.

OnionSearch, nos permite realizar una búsqueda de una palabra o una frase en todos estos buscadores de la red de TOR:

- ahmia
- darksearchio
- onionland
- notevil
- darksearchengine
- phobos
- onionsearchserver
- torgle
- onionsearchengine
- tordex
- tor66
- tormax
- haystack
- multivac
- evosearch
- deeplink

Los resultados nos los devolverá de la misma forma que si hicieramos una consulta en un buscador similar a Google con la opción de guardarlos en un archivo de texto para su posterior procesamiento.

Así que para efectos de recopilar direcciones ONION de diferentes temáticas lo primero que tenemos que ejecutar es el siguiente comando.

```
#onionsearch "término_de_búsqueda"
```

Cómo término de búsqueda puede ser cualquier palabra que se busque de forma similar a lo que haríamos en el buscador de Google, en el siguiente ejemplo vamos a buscar lo relacionado a **Bitcoin** por ejemplo:

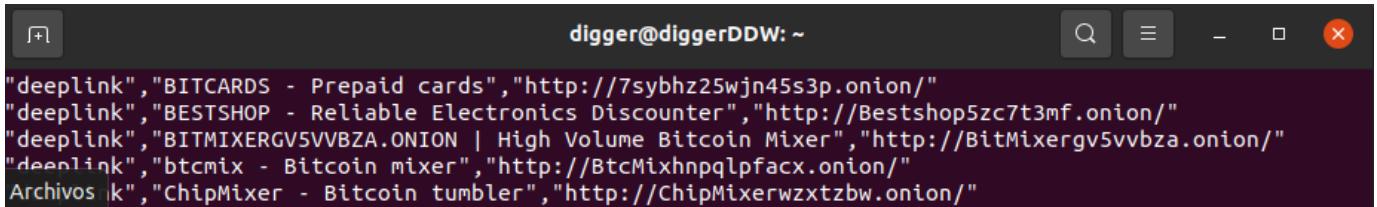
```
digger@diggerDDW:~$ onionsearch bitcoin
search.py started with 3 processing units...
    Dark Search Enginer (#2): 100% | [██████████] 2/2 [00:03<00:00, 1.78s/it]
        OnionLand (#1): 3% | [██] 3/100 [00:04<02:13, 1.37s/it]
Error: unable to connect1): 5% | [███] 5/100 [00:09<02:58, 1.87s/it]
        Ahmia (#0): 100% | [██████████] 1/1 [00:35<00:00, 35.84s/it]
        OnionLand (#1): 18% | [███] 18/100 [00:50<03:50, 2.81s/it]
        Torgle 1 (#0): 100% | [██████████] 30/30 [00:42<00:00, 1.43s/it]
    Onion Search Engine (#0): 4% | [██] 4/100 [00:20<08:17, 5.18s/it]
Error: unable to connect2): 49% | [██████████] 49/100 [00:57<01:00, 1.19s/it]
    Onion Search Server (#2): 100% | [██████████] 100/100 [01:56<00:00, 1.17s/it]
        Torgle (#2): 100% | [██████████] 1/1 [00:04<00:00, 4.39s/it]
        Tormax (#2): 0% | [██████████] 0/1 [00:03<?, ?it/s]
Error: unable to connect): 18% | [███] 18/100 [00:20<01:28, 1.08s/it]
    Onion Search Engine (#0): 13% | [██] 13/100 [01:25<09:30, 6.56s/it]
Error: unable to connect): 38% | [███] 38/100 [00:48<01:32, 1.49s/it]
        Tordex (#1): 38% | [███] 38/100 [00:56<01:32, 1.49s/it]
Error: unable to connect:
        Haystack (#2): 56% | [██████████] 28/50 [00:31<00:24, 1.13s/it]
        DeepLink (#2): 100% | [██████████] 1/1 [00:01<00:00, 1.47s/it]
        Tor66 (#1): 0% | [██████████] 0/30 [00:00<?, ?it/s]
        Tor66 (#1): 40% | [███] 12/30 [01:03<01:42, 5.71s/it]
```

Como se puede observar y de acuerdo a lo mencionado, la aplicación de OnionSearch comenzará a buscar el término que nosotros le indicamos en los diversos buscadores de la red de TOR con lo cual nos enviará un resultado con el número de URLs encontradas y que pudieran estar relacionadas con ese término buscado.

```
Report:
Execution time: 0:05:50.071128 seconds
Results per engine:
ahmia: 1000
darksearchio: 0
onionland: 360
notevil: 0
darksearchenginer: 14
phobos: 0
onionsearchserver: 988
torgle: 20
torgle1: 300
onionsearchengine: 0
tordex: 0
tor66: 600
tormax: 0
haystack: 540
multivac: 0
evosearch: 863
deeplink: 102
Total: 4787 links written to output_bitcoin_20210130104621.txt
```

Al finalizar el proceso de búsqueda, OnionSearch nos mostrará un breve resumen como el que se muestra en la imagen anterior, con el número de respuestas arrojadas por cada servidor, así como el número de URLs total recopiladas, así como el nombre del **archivo CSV** en el cual nos almacenó esta información, para este ejemplo el nombre del archivo es el **output_bitcoin_20210130104621.txt**.

Al abrir este archivo podemos observar que se trata de un archivo estructurado por comas (CSV) a pesar de que la extensión del mismo sea TXT.



```
digger@diggerDDW: ~
"deeplink","BITCARDS - Prepaid cards","http://7sybhz25wjn45s3p.onion/"
"deeplink","BESTSHOP - Reliable Electronics Discounters","http://Bestshop5zc7t3mf.onion/"
"deeplink","BITMIXERGV5VVBZA.ONION | High Volume Bitcoin Mixer","http://BitMixergv5vvba.onion/"
"deeplink","btcmix - Bitcoin mixer","http://BtcMixhnpqlpfacx.onion/"
Archivos [k],"ChipMixer - Bitcoin tumbler","http://ChipMixerwzxtzbw.onion/"
```

La estructura de este archivo es de la siguiente forma:

Nombre del Buscador	Descripción del Sitio	Url
---------------------	-----------------------	-----

Adicionalmente es importante mencionar que **OnionScan** nos generará un archivo de texto por cada término de búsqueda que ingresemos, de tal forma, que si ingresamos 5 términos distintos tendremos un archivo distinto correspondiente a cada término de búsqueda.

Así que para la tarea de recopilación de URLs me arme un breve listado de alrededor de 60 términos de búsqueda para recopilar alrededor de **1.9 Millones de urls** para éste ejercicio de recopilación.

Para lo cual elaboré un script muy sencillo en bash, con el siguiente contenido:

```
digger@diggerDDW:~/TFM/OnionSearch$ cat RunAll2.sh
#!/bin/bash

onionsearch bank --continuous_write True
onionsearch financial --continuous_write True
onionsearch atm --continuous_write True
onionsearch cashout --continuous_write True
onionsearch ammo --continuous_write True
onionsearch gun --continuous_write True
onionsearch hitman --continuous_write True
onionsearch bitcoin --continuous_write True
onionsearch litecoin --continuous_write True
onionsearch leak --continuous_write True
onionsearch hack --continuous_write True
onionsearch hire --continuous_write True
onionsearch "credit card" --continuous_write True
onionsearch exploit --continuous_write True
onionsearch "zero day" --continuous_write True
onionsearch passport --continuous_write True
onionsearch pasaporte --continuous_write True
onionsearch cvv2 --continuous_write True
onionsearch cvv --continuous_write True
onionsearch mobile --continuous_write True
```

Lo único que realizará este script es ir ejecutando la búsqueda en todos los buscadores de cada uno de los términos especificados en cada ejecución, se realizará de forma secuencial de tal forma que cada comando de búsqueda tendrá que esperar a que se ejecute el anterior, con todo lo anterior nos aseguramos de tener miles de resultados en cada búsqueda almacenadas en cada archivo.

Anexo:

Lista de términos de búsqueda empleados para esta investigación:

Lista de términos empleados para la búsqueda con OnionScan		
Visa	ID	ATM
Weed	Service	Politics
Paypal	Home	Mexico
Carding	Hole	Latinoamerica
Chemicals	Search	Racial
Mastercard	Forum	Dinero
Shop	Dir	Hack
Road	List	Services
Onion	Login	Hire
Criminal	Database	BitCoin
Cocaine	Hispano	Cryptos
Money	Spain	Passport
Sex	Turkish	Pasaporte
Room	Darkweb	Identity
Stolen	Hidden	Credentials
Dark	Library	Government
Collection	Links	Ammo
Tor	Chat	Organ
Casino	Porn	Traffic
Mexican	Gun	Btc

Ahora bien, ya tenemos un paso, estoy indagando sobre diversos términos de búsqueda de mi interés, pero aún no he podido analizar la información, qué para ésta altura, ya son varios miles de líneas y en diferentes archivos, así que el siguiente paso a dar, es comenzar a procesar ésta información para continuar la recopilación en 2 sentidos:

- Los nombres de los hosts de los enlaces *.onion para poder realizar los escaneos de puertos y descubrimiento de servicios ocultos.
- Las URLs de los sitios web obtenidos para comenzar a extraer pantallas y los códigos HTML de las páginas que se despliegan en cada uno de ellos.



Concentrando la información:

Parte importante de recopilar la información es ahora la de tener un archivo unificado con todas las URLs, descripciones de sitios que me permita trabajar con este archivo para ir obteniendo los siguientes inputs para los siguientes pasos para continuar con la recopilación de información.

Para poder concentrar toda la información en un solo archivo, ejecuto el siguiente comando desde el directorio en donde se almacenaron todos los archivos productos de la búsqueda de **OnionSearch**.

```
#for file in `ls -tr *.txt`; do cat $file;done > all_sites.list
```

A través de esta línea de comando lo que realicé fue conjuntar todo el contenido en un solo archivo de texto con la extensión *.list, misma que no es relevante, sólo la hice de esta forma para poderla ubicar de forma rápida al listar el contenido del directorio, sobre todo si tengo muchos archivos como es en mi caso.

Al concluir este comando, vemos que al final del día tengo un solo archivo con todos los resultados de búsqueda.

```
digger@diggerDDW:~/TFM/OnionSearch$ for file in `ls -tr *.txt`; do cat $file;done > all_sites.list
digger@diggerDDW:~/TFM/OnionSearch$ cat all_sites.list | wc -l
1949796
```

Para mi ejercicio de recopilación pude conjuntar un total de **1.94 Millones de registros**, aunque todavía falta depurar esta información, ya que es probable que tengamos entre estos resultados de muchas búsquedas con términos de cierta similitud, pero esto lo puedo realizar al extraer tanto las URLs como los nombres de hosts.

```
digger@diggerDDW:~/TFM/OnionSearch$ head -n 5 all_sites.list
"ahmia","Computer program - The Hidden Wiki","http://zqktlw14fecvo6ri.onion/wiki/Computer_program"
"ahmia","Hacking Services - Computer- Social Media","http://ly75dbzixy7hlp663j32xo4dtoikm6bx53jvivq
kpo6jwppptx3sad.onion/index.php?route=product/product&path=101_107&product_id=160"
"ahmia","Debian -- Computer vendors that pre-install Debian","http://sejnfjrq6szgca7v.onion/distrib/p
re-installed"
"ahmia","Selling Modified and Weaponized Computer Virus - BlackHats Lounge","http://32orihrbhrpk5x6o.
onion?product=selling-modified-and-weaponized-computer-virus"
"ahmia","Does using Tor Browser protect other applications on my computer? | Tor Project | Support","
http://4bf1p2c4tnynnbes.onion/tbb/tbb-13/"
```

Ya con este archivo, lo primero que realizaré es extraer los nombres de los hosts para continuar a realizar un escaneo de descubrimiento de puertos para cada host encontrado. Para lo cual emplearé el siguiente conjunto de comandos en Ubuntu.

```
# cat all_sites.list | sed -r 's/", "/"/"/g' | awk -F'"/' '{print $2"*$,$3}' | grep "http" | cut -d "/" -f 3 | sed -r 's/www//g' | sed -r 's/\//g' | grep ".onion$" | uniq > all_hosts_name.list
```

Con el comando anterior, logramos reducir el listado de todas las urls y hosts obtenidos en casi la mitad y extrayendo sólo la parte del hostname del sitio ***.onion**.

```
digger@diggerDDW:~/TFM/OnionSearch$ cat all_sites.list | sed -r 's/", "/"/"/g' | awk -F'"/' '{print $2"*$,$3}' | grep "http" | cut -d "/" -f 3 | sed -r 's/www//g' | sed -r 's/\//g' | grep ".onion$" | uniq > all_hosts_name.list
digger@diggerDDW:~/TFM/OnionSearch$ head -n 5 all_hosts_name.list
zqktlw14fecvo6ri.onion
ly75dbzixy7hlp663j32xo4dtoikm6bx53jvivqkpo6jwppptx3sad.onion
sejnfjrq6szgca7v.onion
4bf1p2c4tnynnbes.onion
quantum2l7xnxtb.onion
```

Y de los cerca de 1.94 Millones de registros por todas las URLs extraídas el número final de hosts quedó en 1.1 Millones de hostnames, mismos que ya podremos comenzar a escanear por medio de NMAP, mismo que instalamos previamente.

Para comenzar a escanear los puertos básicos de todo este listado, emplee el siguiente comando de Linux para realizar el escaneo a través de los hosts por medio de la red de TOR.

```
# proxychains4 nmap -Pn -sV -v -p 22,21,80,443,25,6667,11009,4050,55080 -oX scan_results_full.xml -iL
all_hosts_name.list
```

Lo que el comando anterior, realizará es leer cada una de las entradas del archivo que generé bajo el nombre de **all_hosts_name.list** y por cada entrada de esa lista, nmap realizará un escaneo de los puertos 22,21,80,443,25,6667,11009,4050 y 55080, que son los puertos comúnmente utilizados por los nodos de la red TOR. Para la publicación de los hidden services más comunes.

Los resultados de este escaneo, se almacenarán en un archivo bajo el nombre de **scan_results_full.xml**. Para este proceso se requirieron varios días de ejecución del NMAP debido a la cantidad de hosts, puertos a escanear y la velocidad de los circuitos de la misma red. Así que retomaremos este archivo más tarde para su procesamiento y análisis.

```
digger@diggerDDW:~/TFM/OnionSearch$ proxychains4 nmap -Pn -sV -v -p 22,21,80,443,25,6667,11009,4050,5080 -oX scan_results_full.xml -iL all_hosts_name.list
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-30 17:20 CST
NSE: Loaded 45 scripts for scanning.
Initiating Parallel DNS resolution of 4096 hosts. at 17:20
Completed Parallel DNS resolution of 4096 hosts. at 17:21, 21.55s elapsed
Initiating Connect Scan at 17:21
Scanning 113 hosts [9 ports/host]
[proxychains] Strict chain ... 127.0.0.1:9050 ... ly75dbzixy7hlp663j32xo4dtoiikm6bx53jvivqkpo6jw
ppptx3sad.onion:443 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:9050 ... quantum2l7xnxbtb.onion:443
```

El proceso de escaneo de puertos y servicios en la red de Tor es un proceso lento, mismo que me llevó varios días para obtener apenas el 10% de mi muestra inicial, es decir, apenas poco mas de 10k hosts escaneados.

Posterior al escaneo y entrega de resultados de NMAP lo siguiente que realicé es transformar esta información en formato CSV para hacer mas fácil el tema de la interpretación y análisis.

SCRAPING DE LOS SITIOS ONION

De una forma similar a como realizamos la extracción de los nombres de los hosts en el punto anterior para el escaneo de puertos con NMAP. Otra de las actividades importantes a realizar es la de comenzar a relacionar los sitios onion encontrados en la fase de recopilación a través de varios parámetros ubicados como pueden ser: El contenido del sitio web, servicios publicados, versiones de tecnologías, certificados digitales, etc...

Para este paso requerí de emplear el mismo archivo bajo el nombre de **all_sites.list** pero ahora para obtener las URLs de los sitios onions identificados, para lo cual tuve que ejecutar el siguiente comando.

```
# cat all_sites.list | sed -r 's/'//'"!'"'g' /awk -F'!' '{print $3}' | sed -r 's/\//g' | uniq > all_urls.list
```

Con ese comando podremos obtener todas las URLs de los sitios onions obtenidos de la búsqueda que realizamos con OnionScan, en este caso a diferencia de los resultados que obtuvimos con la generación del archivo de los nombres de los hosts no podremos depurar mucho ya que los resultados en este caso son variados y también lo son las URLs.

```

digger@diggerDDW:~/TFM/OnionSearch$ cat all_sites.list | sed -r 's/", "/!"/g' | awk -F'!'"'{print $3}' | uniq > all_urls.list
digger@diggerDDW:~/TFM/OnionSearch$ cat all_urls.list | wc -l
1870182
digger@diggerDDW:~/TFM/OnionSearch$ head -n 5 all_urls.list
"http://zqktlw14fecvo6ri.onion/wiki/Computer_program"
"http://ly75dbzixy7hlp663j32xo4dtoikm6bxbs3jvivqkpo6jwppptx3sad.onion/index.php
?route=product/product&path=101_107&product_id=160"
"http://sejnfjrq6szgca7v.onion/distrib/pre-installed"
"http://32orihrbhpk5x6o.onion?product=selling-modified-and-weaponized-computer-
virus"
"http://4bf1p2c4tnynnbes.onion/tbb/tbb-13/"

```

En este caso pudimos reducir un poco el listado de datos obtenidos de 1.9 Mill a sólo 1.8 Mill de direcciones distintas de tipo onion. Hasta este punto tenemos gran parte del trabajo realizado, sin embargo aún necesitamos realizar un formato de JSON de este listado para comenzar con el procesamiento de la misma.

Para esa actividad realicé el siguiente script de Python para comenzar a realizar el crawling automatizado de esta gran cantidad de sitios.

===== Create_Json_Crawling.py =====

```

#!/usr/bin/env python

#Esta linea obtendra el total de lineas (url) recopiladas, este dato nos servira como bandera para
saber en que momento debemos cerrar el arreglo de JSON.

count = len(open("all_urls.list").readlines())
f_in = open ('all_urls.list')
f_out = open ('all_urls.json','w')
index_flag = 1

f_out.write("{}")
for line in f_in.readlines():
    label = "\"Sitio_A" + str(index_flag) + ":""

    if len(line)>5:
        line = line.replace('\r', '').replace('\n', '')
        line = '""' + line + '""'

    if line.find(".onion") != -1:

        if (index_flag == count ): #Es el ultimo registro del archivo.
            f_out.write(label + """+ line + '\n')
        else:
            if not line.strip():
                print("()")
            elif not line in ["\n",'r\n']:

```

```

f_out.write(label +""+ line + ",\n")
index_flag +=1
else:
    index_flag +=1
else:
    index_flag +=1

f_out.write("}")
f_in.close()
f_out.close()

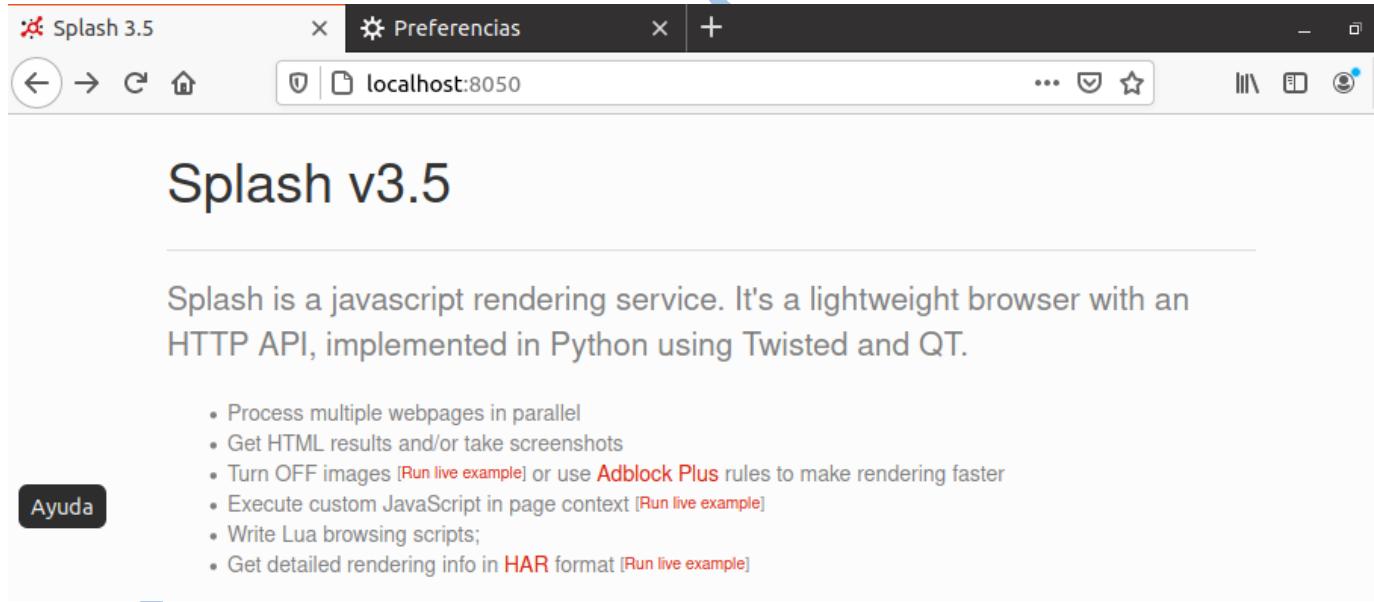
```

A través del archivo anterior convertimos el archivo all_urls.list a un listado en formato tipo Json que nos permitirá procesar cada una de éstas URLs con la aplicación de SPLASH, misma que instalamos previamente.

Para validar que nuestro servicio se encuentra activo, recuerden que podemos abrir una ventana del navegador y poner la siguiente dirección web:

<http://127.0.0.1:8050>

Deberemos ver la siguiente ventana:



Si observamos esa pantalla quiere decir que nuestro servicio se encuentra activo y listo para procesar la información que nosotros le ingestemos a través del archivo json que acabo de generar.

Si no está disponible ese servicio, pueden intentar levantarla o ejecutarlo a través del siguiente comando:

```
#sudo docker run -p 8050:8050 -v /etc/splash/proxy-profiles:/etc/splash/proxy-profiles --net="host"  
scrapinghub/splash --proxy-profiles-path=/etc/splash/proxy-profiles --max-timeout 500
```

Si todo funciona correctamente deberemos comenzar a procesar la información que acabo de procesar en el archivo Json a través del archivo de Python.

Para lo siguiente necesitamos ubicar el directorio en donde nosotros instalamos la aplicación de Page-Compare.

Para mi caso, yo lo instalé en **/home/digger/page-compare** y en esa carpeta deberemos copiar el archivo **all_urls.json** recién generado y ya que lo tenemos en esa misma carpeta ejecutaremos el siguiente comando para comenzar a realizar un crawling de todos los sitios de la lista.

```
#python3 scrape.py http://127.0.0.1:8050 all_urls.json 1 data
```

Una vez ejutado este comando de igual forma tomará un gran tiempo para su ejecución, ya que lo que el comando anterior realiza es obtener un screenshot de la página que se muestra en cada sitio *.onion, así como el código html de la misma para posteriormente realizar una comparación.

```
digger@diggerDDW:~/TFM/page-compare$ python3 scrape.py http://127.0.0.1:8050 all_urls.json 1 data  
Requesting Sitio_A1 (http://zqktlwi4fecvo6ri.onion/wiki/Computer_program).  
2021-01-31 01:44:33.351325 [events] {"path": "/render.json", "rendertime": 6.764827728271484, "maxrss": 220264, "load": [0.29, 0.18, 0.17], "fds": 57, "active": 0, "qsize": 0, "_id": 139904601850768, "method": "GET", "timestamp": 1612057473, "user-agent": "python-requests/2.22.0", "args": {"html": "1", "png": "1", "width": "400", "height": "300", "timeout": "10", "images": "0", "url": "http://zqktlwi4fecvo6ri.onion/wiki/Computer_program"}, "uid": 139904601850768}, "status_code": 200, "client_ip": "127.0.0.1"}  
2021-01-31 01:44:33.351451 [-] "127.0.0.1" - - [31/Jan/2021:01:44:32 +0000] "GET /render.json?html=1&png=1&width=400&height=300&timeout=10&images=0&url=http%3A%2F%2Fzqktlwi4fecvo6ri.onion%2Fwiki%2FComputer_program HTTP/1.1" 200 224124 "-" "python-requests/2.22.0"
```

NOTA: Durante este proceso el equipo pudiera quedarse pasmado debido a que la cantidad de peticiones que realiza a la red de TOR y también el proceso de rendering de la página pueden llegar a consumir muchos recursos de cómputo, lo mejor será asignar tanto memoria ram como procesador al equipo en el cual se realice esta actividad.

Cabe mencionar que toda la información resultante (imagen y código html) se almacenarán en la subcarpeta de “Data” dentro del directorio page-compare dado que fue ahí desde donde ejecutamos el archivo...

Tras varios días después...

Posterior a la recolección de información de tipo crawling de los sitios web que se obtuvieron por medio de este proceso misma que me llevó varios días, logré recopilar alrededor de 57,000 sitios scrapeados para poder realizar el siguiente paso de procesamiento de los datos, que es la de generar las relaciones de cada sitio, sin embargo, me encontré con una restricción de la herramienta **compare-tags-all.py** para procesar

gran cantidad de información contenida en el directorio de data, ya que posterior a los 1400 sitios comparados, la solución comienza a fallar y con ello el proceso se detiene, sin generar algún log.

Derivado de lo anterior, tuve que generar un script de python para generar carpetas para cada organizar 1400 sitios por carpeta y evitar así que la aplicación de compare-tags-all.py se detenga y no genere nada de información.

El inconveniente de realizar esto es que tendremos imágenes separadas de un gran universo de lo que se detecta de los sitios de la Dark Web como lo veremos mas adelante.

Este script se tiene que ejecutar dentro de la misma carpeta de “**data**” dentro de la ruta en la cual instalamos la utilería de “**page-compare**”.

===== Create_bulk.py =====

```
import shutil,os

f = open("list.txt",'r')
dir=1
y=0
foldername= "Carpeta_" + str(dir)
os.mkdir(foldername)
for x in f:
    file = x.replace("\n","")
    if y <= 2800:
        y+=1
        shutil.copy(file,foldername)
        print(x)
    else:
        y=0
        dir+=1
        foldername= "Carpeta_" + str(dir)
        os.mkdir(foldername)
        shutil.copy(file,foldername)
```

Como ya lo he explicado, lo único que hace este script es organizar carpetas con 1400 sitios cada una para realizar el procesamiento por bloques.

Posterior a organizar todos los resultados por carpetas es necesario comenzar con el procesamiento de las similitudes de los mismos por contenidos para comenzar a ubicar posibles relaciones de los mismos con alguna infraestructura que pudiera encontrarse detrás de los mismos. Para lo cual usaré la herramienta de **compare-tags-all.py** en cada carpeta generada y con ello obtener un archivo de tipo JSON con las posibles similitudes entre cada sitio.

Sin embargo como eran demasiadas carpetas la labor de realizarlo 1x1 es demasiado tediosa, para lo cual realice el siguiente script

```
===== Compare_Sites.sh =====
```

```
#!/bin/bash
for directory in `find data/mydata/ -maxdepth 1 -type d | grep Carpeta`
do
    readarray -d / -t DirName <<< "$directory"
    array_lenght=${#DirName[@]}
    array_lenght=`expr $array_lenght - 1`
    python3 compare-tags-all.py $directory
    Fname=${DirName[$array_lenght]}
    Fname=${Fname%$'\n'}
    mv compare-tags.json "compare-tags_${Fname}.json"
done
```

Este proceso, en mi caso tardó algunas horas, por la cantidad de información que alcancé a recopilar, aproximadamente tardó cerca de 10 horas, sin embargo al finalizar se obtiene un archivo .json por cada uno de los directorios procesados por este script.

Lo que resta por realizar con cada json es crear su representación gráfica (mapa de relación) para poder visualizar la relación entre sitios.

Para lo cual, de igual forma me apoyé del siguiente script en bash, dentro de la carpeta de “**page-compare**”

```
===== Make_Images.sh =====
```

```
#!/bin/bash
for file in `ls *Carpeta*.json`
do
    readarray -d _ -t FileName <<< "$file"
    array_lenght=${#FileName[@]}
    array_lenght=`expr $array_lenght - 1`
    Fnum=${FileName[$array_lenght]}
    Fnum=${Fnum%$'\n'}
    readarray -d . -t num <<< "$Fnum"
    num=${num%$'\n'}
    python3 graph.py "compare-tags_Carpeta_${num}.json" data/ > "grafica_Carpeta_${num}.dot"
    neato -O -Tpng "grafica_Carpeta_${num}.dot"
done
```

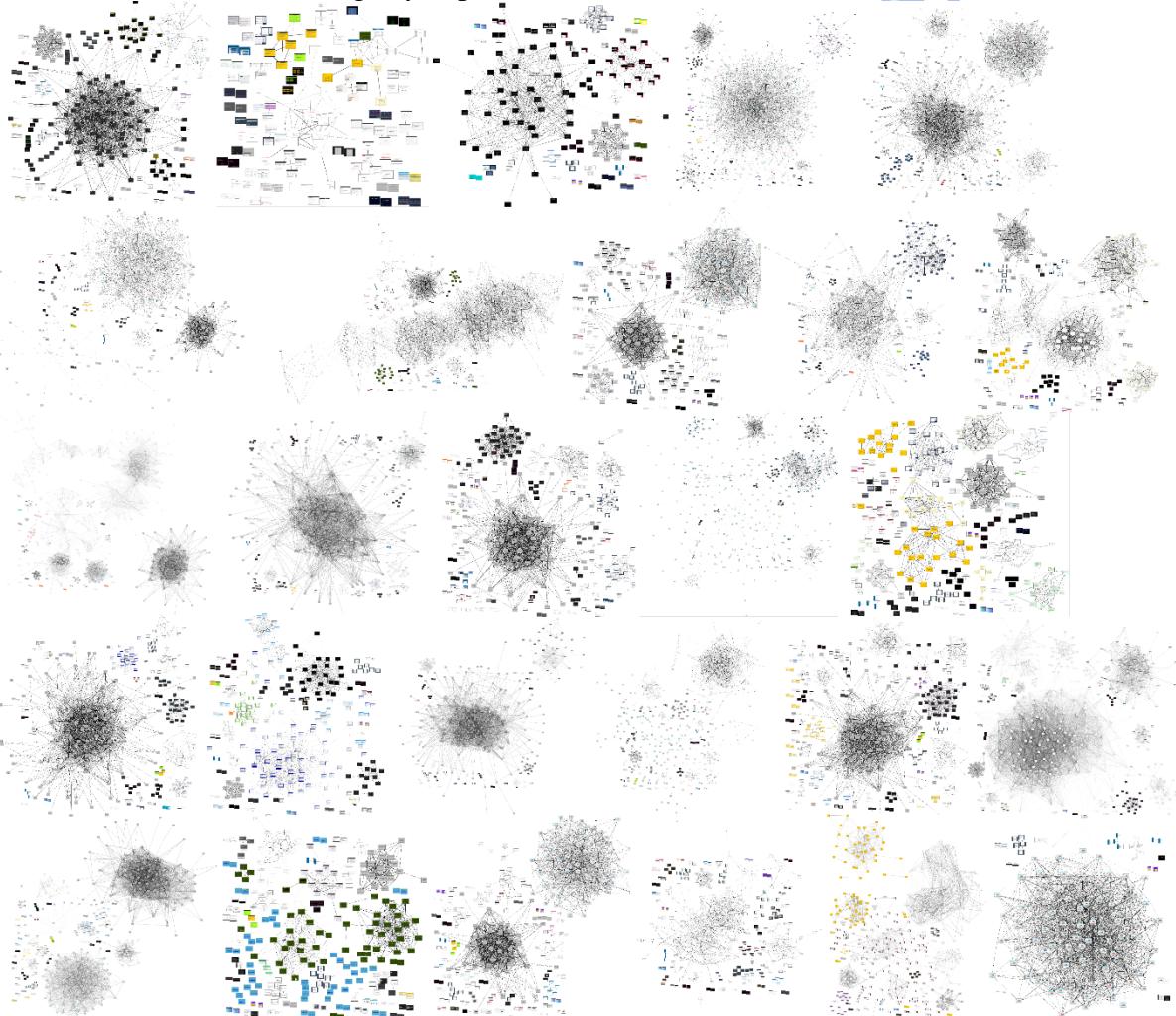
De igual forma este script generará una serie de imágenes (1 por carpeta) con la relación en forma de diagrama de nodos por cada conjunto de carpetas, en mi caso y por la cantidad de información recopilada, me generó **27 imágenes** (1 por carpeta) recordemos que cada carpeta tiene un máximo de 1400 sitios, por lo que nuestro muestreo es bastante completo.

Posteriormente entraré en detalle de lo que se observó como parte de la fase de análisis de toda esta información recabada y procesada.

Sin embargo lo que puedo mostrar es una especie de collage con todas éstas imágenes para darnos una idea de como se puede ir visualizando este universo.

A pesar de que no se distingue muy bien, se pueden ver las relaciones entre todos los sitios de acuerdo por el contenido de la página que exponen hay vecindades bastante pobladas y otras mas discretas y aisladas.

Mas adelante detallaré estos hallazgos y la posible relación con infraestructura.



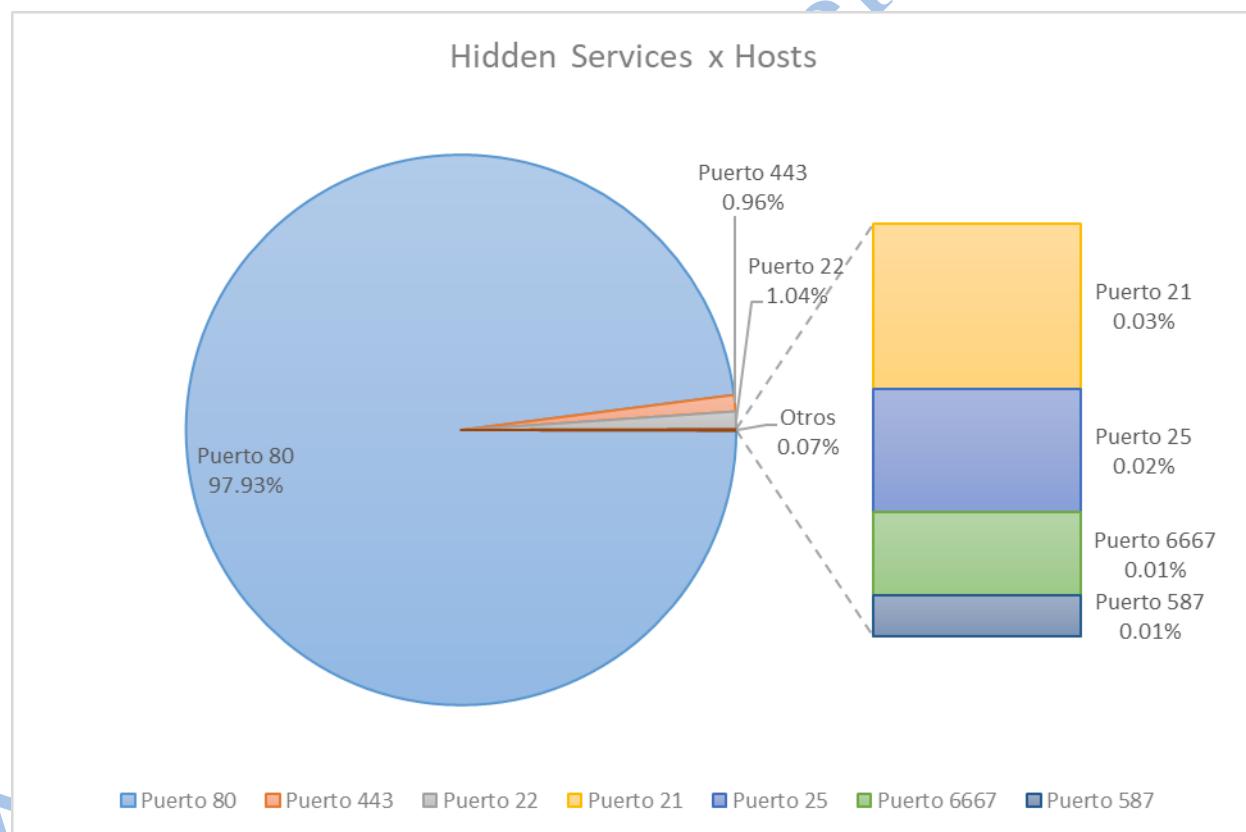
6.1.3 ANALISIS DE LA INFORMACIÓN – (ETAPA DE ANÁLISIS)

Finalmente con la información tanto de los escaneos de puertos como el comparativo de los contenidos de las urls recopiladas tengo la materia prima para comenzar a realizar el análisis de la misma.

Debido a que por el tamaño de la información y por el hecho de no contar con diversos equipos para recopilar información fue necesario acortar la muestra a 57,200 sitios de los cuales fue posible obtener la información relacionable del punto anterior y de ese mismo muestreo ha sido posible recopilar información de los servicios en un 35% de ese muestreo, es decir, se tiene información de los servicios que residen en alrededor de 17,259 servidores o dominios onion.

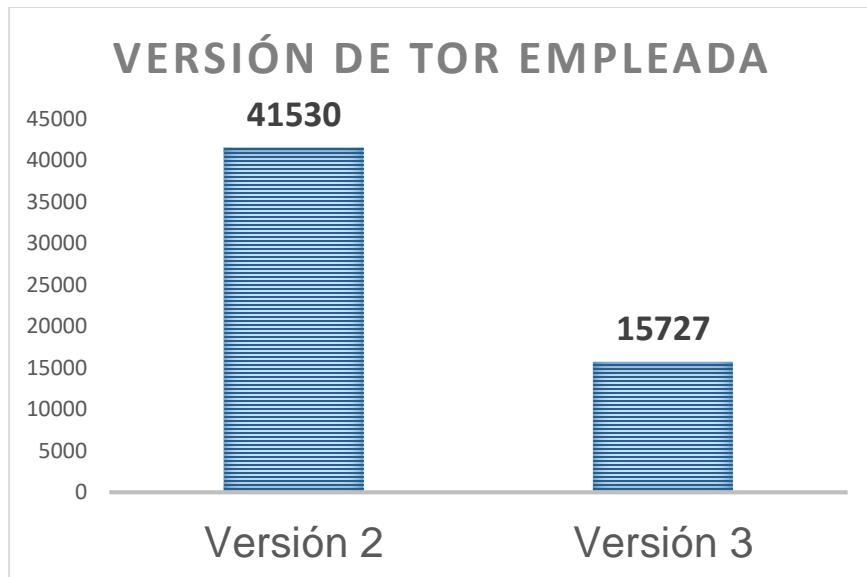
Por lo que ese será nuestro punto de partida para realizar este análisis.

En una primer revisión se observan los siguientes puertos abiertos y por ende el tipo de Hidden Service asociado a cada host de los que se pudo obtener información. Es de destacar que más del 98% está asociado a proveer contenido web (HTTP/HTTPS) , mientras que otros servicios como FTP, SSH, SMTP, IRC, etc.. ocupan menos del 2% del total de la muestra analizada, tal y como se observa en el gráfico siguiente:

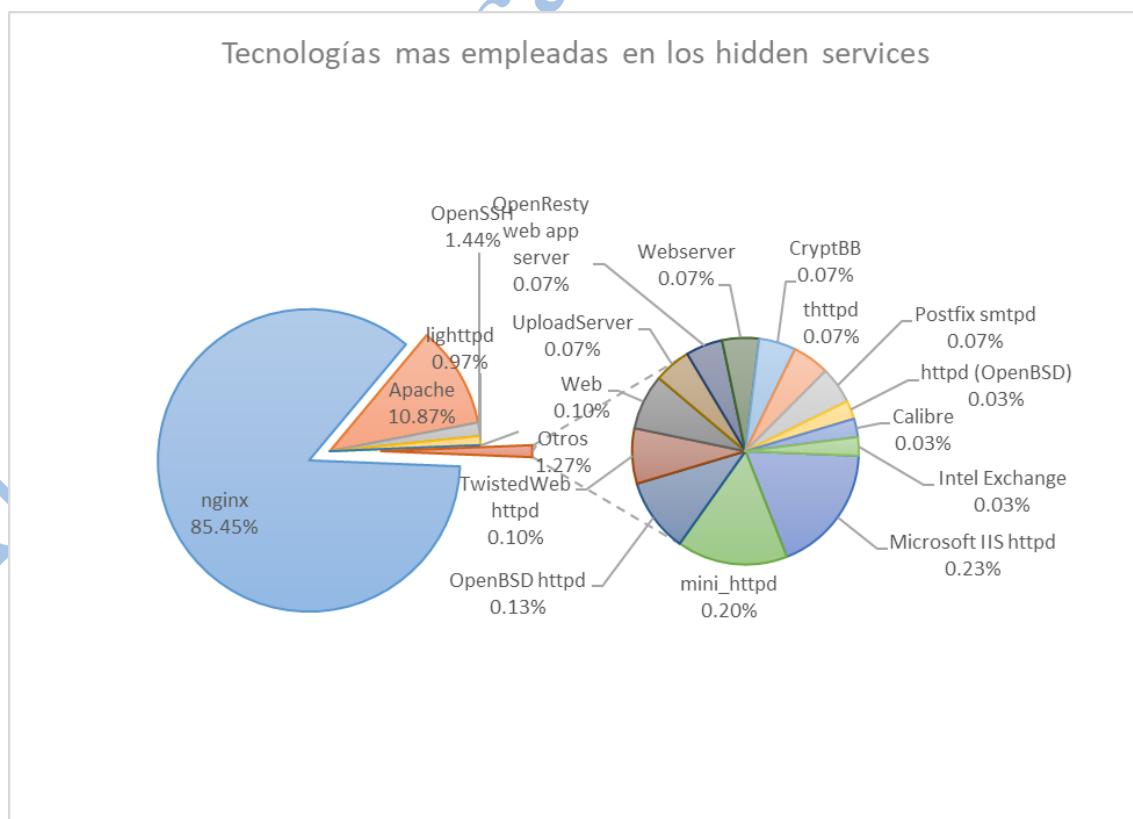


Puerto	80	443	22	21	25	6667	587
Hosts	14683	144	156	4	3	2	1
Servicio	HTTP	HTTPS	SSH	FTP	SMTP	IRC	SMTP

Así mismo otro dato relevante es la versión de sitio TOR empleada por los hosts observados en la muestra, al menos un 25% emplean la Versión 3 de TOR, mientras que predomina aún la versión 2 con un 75% de los sitios analizados.



En cuanto a las tecnologías empleadas para dar soporte a los Hidden Services, se encuentra que poco más del 85% de los servicios que nos proporcionaron este dato utilizan alguna versión de Nginx, seguido por Apache en un 10.87%, seguido por un discreto 1.44% de SSH, el resto de la muestra se disuelve en otras tecnologías.



Alfredo

In 2021

Servicio/Tecnología y Versión	Hosts
22	43
OpenSSH	43
25	1
Postfix smtpd	1
80	2919
nginx	2395
Apache httpd	215
nginx/1.16.1	82
Apache	79
lighttpd	26
nginx/1.18.0 (Ubuntu)	13
nginx/1.14.1	10
nginx/1.14.0 (Ubuntu)	6
nginx/1.14.2	5
nginx/1.16.0	5
mini_httpd	4
OpenBSD httpd	4
Apache/2.4.29 (Ubuntu)	3
nginx/1.17.6	3
Apache/2.4.10 (Debian)	3
Apache/2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.5	3
nginx/1.10.3 (Ubuntu)	3
Apache/2.4.7 (Ubuntu)	3
nginx/1.6.2	3
nginx/1.10.3	3
Web	3
nginx/1.14.0	2
lighttpd/1.4.45	2
TwistedWeb/17.1.0	2
CryptBB	2
Microsoft IIS httpd	2
UploadServer	2
Microsoft-IIS/5.0	2
Apache/2.4.43 (Win64) OpenSSL/1.1.1g	2
Apache/2.4.38 (Debian)	2
nginx/1.17.3	2
Webserver	2
nginx/1.10.2	2
mini_httpd/1.23 28Dec2015	2
thttpd	1
nginx/1.2.1	1
Apache/2.4.18 (Ubuntu)	1
Apache/2.4.25 (Debian)	1
Apache/2.4.34 (Win32) OpenSSL/1.0.2o PHP/5.6.38	1
Apache/2.4.41 (Ubuntu)	1
TwistedWeb httpd	1
lighttpd/1.4.53	1
nginx/1.12.1	1
Microsoft HTTPAPI httpd	1
MS-IIS/8.0	1
Apache/2.4.43 (Unix) OpenSSL/1.1.1g PHP/7.2.31 mod_perl/2.0.8-dev Perl/v5.16.3	1
OpenResty web app server	1
Apache/2.4	1
thttpd/2.27 19Oct2015	1
Apache/2.4.46 (Debian)	1
calibre 3.39.1	1
Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9 PHP/5.4.16	1
httpd (OpenBSD)	1
nginx/1.19.0	1
Intel Exchange	1
nginx/1.19.2	1
443	25
nginx	12
Apache	6
nginx/1.14.0	2
nginx/1.10.2	2
Microsoft-IIS/7.5	1
openresty/1.17.8.2	1
Apache httpd	1
587	1
Postfix smtpd	1

Fig: Detalle de versiones de tecnología descubierta por tipo de servicio.

Para el siguiente punto a concluir en cuanto a los puntos del planteamiento inicial, la siguiente incógnita a resolver es la de: ¿Cómo partiendo de una vulnerabilidad, es posible desanimizar la infraestructura en la que se encuentran instalados los hidden services?.

Para lo cual fue necesario rescatar el resultado del escaneo realizado con NMAP a los servidores descubiertos en la etapa de recopilación de información, este resultado como mencioné anteriormente, se almacenó en un archivo bajo el nombre de **scan_results_full.xml**, entonces para comenzar a trabajar con esta información, fue necesario pasarlo a CSV, a través de la herramienta “**Nmap-to-CSV**” descargada e instalada anteriormente.

Para lo cual sólo fue necesario ubicarme en el directorio en donde instalé la herramienta y posteriormente copiar el archivo XML del resultado del nmap en esa misma carpeta, para ejecutar el siguiente comando:

```
#python3 nmap_xml_parser.py -f scan_results_full.xml -csv Scan_Table.csv
```

Tal como lo muestra la siguiente pantalla me generará un archivo CSV con los resultados del escaneo, así como los puertos abiertos en cada host.

```
digger@diggerDDW:~/TFM/Nmap-Scan-to-CSV$ python3 nmap_xml_parser.py -f scan_results_full.xml -csv Scan_Table.csv
[+] The file Scan_Table.csv does not exist. New file created!
digger@diggerDDW:~/TFM/Nmap-Scan-to-CSV$ head -n 5 Scan_Table.csv
IP,Host,OS,Proto,Port,Service,Product,Service FP,NSE Script ID,NSE Script Output,Notes
224.0.0.1,z...ri.onion,,tcp,80,http,nginx,,,,
224.0.0.2,s...7v.onion,,tcp,80,http,Apache httpd,,,
224.0.0.4,b...pe.onion,,tcp,80,http,Apache httpd,,,
224.0.0.5,r...lz.onion,,tcp,80,http,lighttpd,,,
digger@diggerDDW:~/TFM/Nmap-Scan-to-CSV$
```

De este archivo la información relevante que nos interesa es la de comenzar a ubicar versiones de gestores de contenido y servidores web, tales como: Apache, Nginx, lighttpd, etc..

Así que para comenzar a procesarlo ejecuté los siguientes comandos para comenzar a realizar extractos de información valiosa que me pudiera dar pie a encontrar alguna posible versión vulnerable:

Por ejemplo, con este comando, pude obtener todas las direcciones *.onion que cuentan con servidor Apache instalado, esto me servirá para poder realizar la búsqueda de una mala configuración, misma que en caso de encontrarse, me indicaría el sistema operativo instalado y en caso de que se encontrara el gestor de contenido como complemento.

```
#cat Scan_Table.csv | cut -d "," -f 2,7 | grep Apache
```

Acá un ejemplo de la salida proporcionada por este comando, si se suprime la columna 7, nos dejaría una lista sólo de sitios *.onion

```
digger@diggerDDW:/TFM/Nmap-Scan-to-CSV$ cat Scan_Table.csv | cut -d "," -f 2,7 | grep Apache
se      7v.onion,Apache httpd
bn      pe.onion,Apache httpd
ha      5q.onion,Apache httpd
7p      ha.onion,Apache httpd
f7      ne.onion,Apache httpd
jq      gk.onion,Apache httpd
e4      yt.onion,Apache
cw      zf.onion,Apache/2.4.7 (Ubuntu)
py      rd.onion,Apache/2.4.46 (Debian)
vy      lf.onion,Apache
ug      az.onion,Apache
pa      kp.onion,Apache
pa      kp.onion,Apache
rc      dq.onion,Apache httpd
55      wz.onion,Apache httpd
ph      yr.onion,Apache httpd
7l      ii.onion,Apache httpd
7l      ii.onion,Apache httpd
sk      mb.onion,Apache httpd
```

Estos resultados los almaceno en un archivo bajo el nombre de **Sites_With_Apache.list** para poder procesarlo con la búsqueda del archivo mal configurado y con ello comenzar a obtener mas información.

===== GetAll_SS.sh =====

```
#!/bin/bash

for page in `cat Sites_With_Apache.list`
do
filename="${page}.html"
touch $filename
curl "${page}/server-status" > $filename
done
```

Este script lo que realiza es que a través del listado obtenido anteriormente va recorriendo el listado y a través del comando CURL, va buscando el archivo **server-status**, mismo que en caso de estar expuesto podría revelar mas información valiosa que nos pudiera llevar o bien a una versión vulnerable de su manejador o inclusive hasta la IP del servidor mismo, revelando así su posible ubicación, nombres de usuario, etc...

Para el ejercicio que realicé no fue posible ubicar con los resultados del escaneo algo de información que nos pudiera dar este indicio, aunque si versiones potencialmente vulnerables, mismas que de ser explotadas podría contribuir a revelar información o hasta tomar control del equipo.

Así mismo, va generando un archivo con el nombre del sitio *.onion para que posteriormente buscando la evidencia o información que tengan estos archivos podamos realizar en una sola búsqueda aquellos equipos que hacen disclosure de información sensible a través de esta mala configuración.

```

 GetAll_SS.sh
grl          wjl.onion.html
grl          pda.onion.html
grl          ois.onion.html
gui          rre.onion.html
gvi          x3k.onion.html
hai          75q.onion.html
hai          qzq.onion.html
hai          e2b.onion.html
hai          h42n5o7gbzrewxee3'
ndi          iwk.onion.html
ng           6db.onion.html
hi           6km.onion.html
hi           yzr.onion.html
hi           2ed.onion.html
hl           7wr.onion.html
hoi          gku.onion.html

```

Y ya para buscar la información sensible o algo que nos pudiera dar algún indicio relevante para el tema de desanonomizar, podemos ejercutar alguna de las siguientes búsquedas en ésta carpeta.

```
#grep -rnw '.' -e "<address>*" | uniq | grep address
```

Busca en todos los archivos contenidos de la carpeta, la etiqueta de <address> misma que nos podría indicar la dirección IP real del equipo, la version del manejador de contenido, Sistema operativo, tal com se puede ver en la siguiente imagen:

```

./resistanhbucoyeb.onion.html:719:<address>Apache/2.4.38 (Debian) Server at resistanhbucoyeb.onion Port
80</address>
./2gzxfta4w657xuz7.onion.html:8:<address>Apache/2.4.29 (FreeBSD) Server at 2gzxfta4w657xuz7.onion Port
80</address>
./upv3wvf6sikfiluy.onion.html:8:<address>Apache/2.4.29 (Ubuntu) Server at upv3wvf6sikfiluy.onion Port 8
0</address>
./drk4ijmcv5pq677z.onion.html:659:<address>Apache/2.4.10 (Debian) Server at drk4ijmcv5pq677z.onion Port
80</address>
digger@diggerDDW:~/TFM/Get_All_Server_Status$ grep -rnw '.' -e "<address>*" | uniq | grep address

```

La siguiente linea de comandos, nos traerá un listado de aquellos equipos que configuraron correctamente el servidor web de apache y no permiten visualizar el archivo

```
#grep -rnw '.' -e "Forbidden" | uniq
```

Por lo tanto en estos equipos a pesar de tener Apache instalado, no se puede obtener mas información.

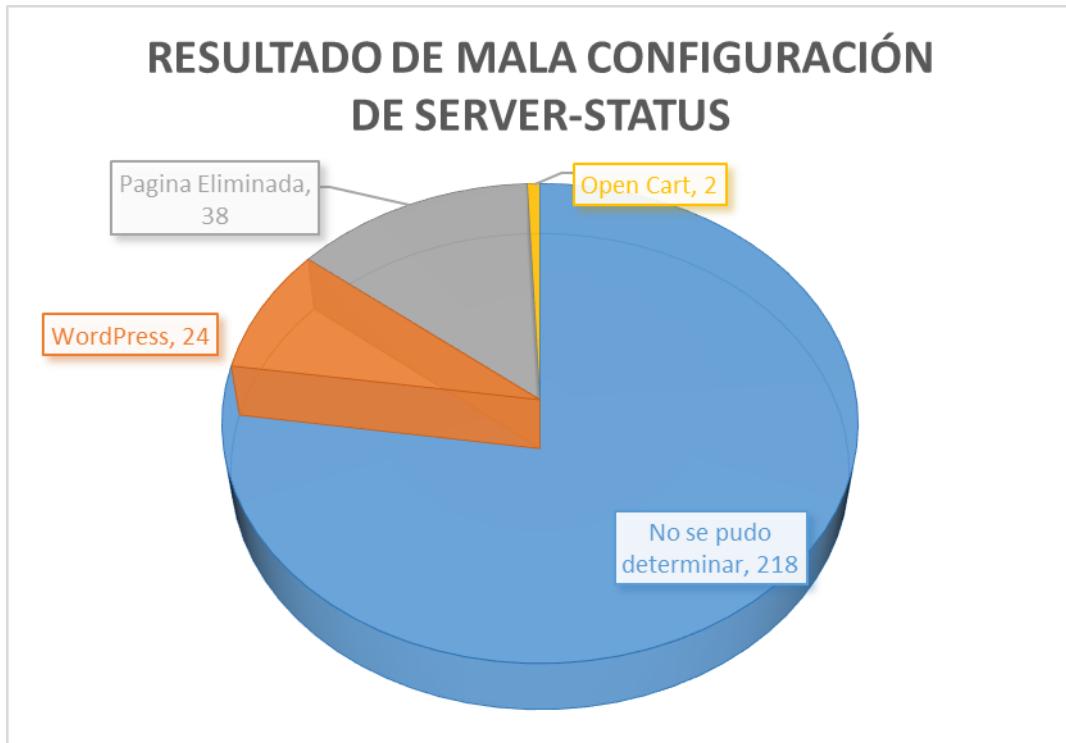
```

./4hohkxjvlt5fjzqv.onion.html:5:<h1>Forbidden</h1>
./xpll5hy2jlze25w2.onion.html:3:<title>403 Forbidden</title>
./xpll5hy2jlze25w2.onion.html:5:<h1>Forbidden</h1>
./dumpscctorpress2sarn7xw.onion.html:3:<title>403 Forbidden</title>
./dumpscctorpress2sarn7xw.onion.html:5:<h1>Forbidden</h1>
digger@diggerDDW:~/TFM/Get_All_Server_Status$ grep -rnw '.' -e "Forbidden" | uniq

```

Y así podríamos seguir haciendo diversos filtros para obtener la información valiosa, misma que me di a la tarea de concentrar en un Excel para resumir los principales, hallazgos.

De todos los equipos que había identificado con servicios de Apache expuestos, esto es lo que encontré a través de la búsqueda de esta mala configuración del archivo server-status.



En resumen, de todos los equipos en los que se analizó esta posible vulnerabilidad dada a una mala configuración se pudo obtener solamente que en 24 equipos se encuentra instalada alguna versión de Wordpress, lo que representa el 9% del universo y muy por detrás se asoma un discreto 1% de equipos con OpenCart instalado.

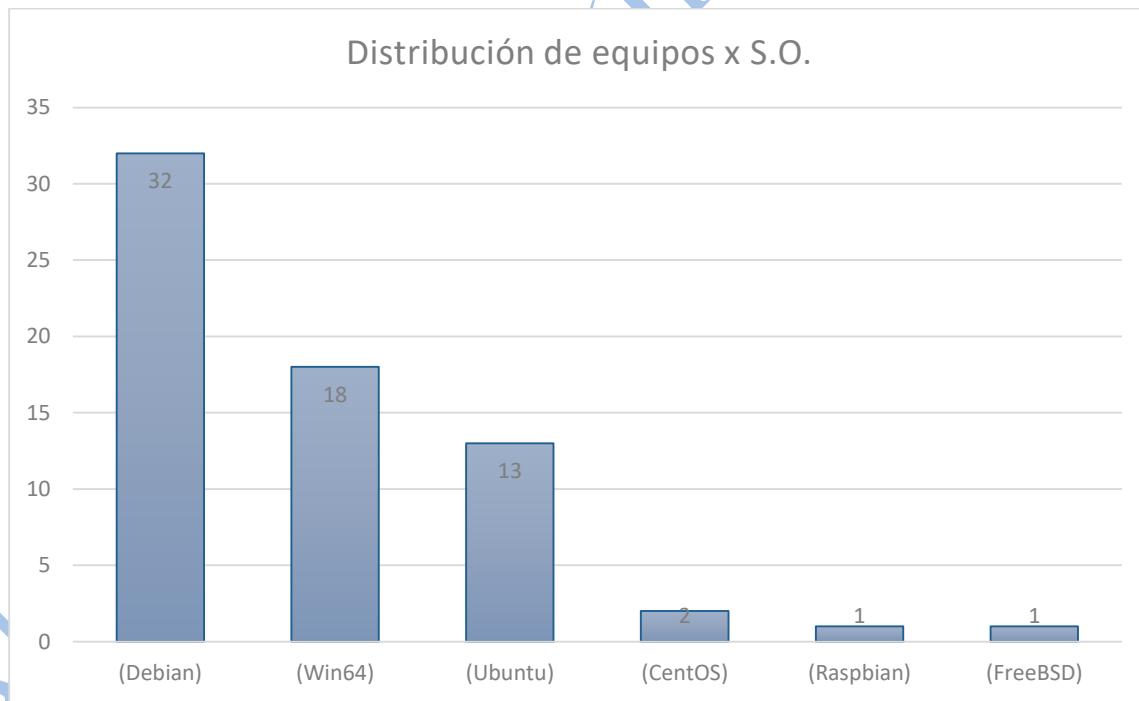
Como sabemos algunas de las versiones de Wordpress han sido explotadas fácilmente para obtener control del equipo, sin embargo el alcance del presente estudio, no contempla validar la explotación de estas vulnerabilidades, si no simplemente mencionar algunas que pudieran dar como resultado el compromiso de la infraestructura que alberga estos hidden services.

Como un ejemplo de los sitios con alguna versión de Wordpress, podemos ejecutar el siguiente comando, nuevamente la carpeta donde se han almacenado todos los archivos con la información recopilada.

```
digger@diggerDDW:~/TFM/Get_All_Server_Status$ grep -rnw '.' -e "wp-*" | cut -d ":" -f 1 | uniq
./m3          6.onion.html
./ub          z.onion.html
./yi          q.onion.html
./gr          s.onion.html
./nq          q.onion.html
./av          5.onion.html
./lg          d.onion.html
./pu          hkllg5bttt2dmiaexs3ggmfpyewc44vt5265uuad.onion.html
./ca          b.onion.html
./dh          g.onion.html
./dr          4.onion.html
./o3          o.onion.html
./4r          a.onion.html
./ha          q.onion.html
./we          7.onion.html
./pc          7.onion.html
./hw          j.onion.html
./ch          q.onion.html
./re          b.onion.html
```

Con esta información el siguiente paso sería indagar por medio de un fingerprinting mas detallado que versión de Wordpress tiene instalado, así como las vulnerabilidades asociadas a la misma.

De igual forma, un dato relevante que se pudo obtener de este análisis es la de que en algunos casos arrojó información del Sistema Operativo que tienen instalados estos equipos.



Tal y como se puede observar, la mayor distribución empleada para equipos con Apache y que se encuentran en la DarkWeb son equipos con Debian, seguidos de equipos con Windows y Ubuntu.

Por lo que de la misma manera podríamos darnos una idea de la superficie de ataque que podría tener una infraestructura determinada.

Para aterrizar un poco mejor lo anteriormente comentado, con la información que pudimos obtener y una actividad de fingreprinting de la versión del S.O. y gestor de contenido web, podríamos fácilmente ubicar el tipo de exploit o vulnerabilidad que nos pueda llevar al control total del equipo y con ello lograr desanonomizar la infraestructura del sitio (s) alojados en un servidor determinado.

De igual forma y aproposito de develar las IPs reales, sobre ese mismo volcado de archivos, podemos ejectar el siguiente comando, y obtener las IPs arrojadas en los mismos.

```
#grep -rnw '.' -e "ip" / uniq
```

```
digger@diggerDDW:~/TFM/Get_All_Server_Status$ grep -rnw '.' -e "ip"
./l...ku.onion.html:63:      "ip":"".
./cWL...f.onion.html:44:</td><td>143.110. 9</td><t
u</td><td nowrap>POST /wordpress/xmlrpc.php HTTP/1.1</td></tr>
./c...f.onion.html:48:</td><td>143.110.2      </td><t
u</td><td nowrap>POST /wordpress/xmlrpc.php HTTP/1.1</td></tr>
./cwu7...nion.html:52:</td><td>143.110.      </td><t
u</td><td nowrap>POST /wordpress/xmlrpc.php HTTP/1.1</td></tr>
```

En la imagen anterior podemos observar como el dominio **c*****zf.onion**, se encuentra alojado en alguna IP de Digital Ocean.

WHOIS IP Lookup Tool

The IPWHOIS Lookup tool finds contact information for the owner of a specified IP address.

Enter a host name or an IP address:

Related Tools: [DNS Traversal](#) [Traceroute](#) [Vector Trace](#) [Ping](#) [WHOIS Lookup](#)

Source: whois.arin.net
IP Address: 143.110...
Name: DIGITALOCEAN-143-110-
Handle: NET-143-110-
Registration Date: 17/01/20
Range: 143.110.0.255
Org: Digitalocean, LLC

Esa IP en algún momento estuvo asignada a un dominio de nombre **h*****vc.com**

SuperTool Beta7

143.110 Reverse Lookup

ptr:143.110 Find Problems

Type	IP Address	Domain Name
PTR	143.110. DigitalOcean, LLC (AS14061)	h.c.com

Mismo que fue registrado en China:

Resultados de la búsqueda de WHOIS

Domain Name: H C.COM

Registry Domain ID: 2083660193_DOMAIN_COM-VRSN

Registrar WHOIS Server: grs-whois.hichina.com

Registrar URL: <http://www.net.cn>

Updated Date: 2020-04-25T13:47:39Z

Creation Date: 2016-12-21T12:38:02Z

Registry Expiry Date: 2022-12-21T12:38:02Z

Registrar: Alibaba Cloud Computing (Beijing) Co., Ltd.

Registrar IANA ID: 420

Registrar Abuse Contact Email: DomainAbuse@service.aliyun.com

Registrar Abuse Contact Phone: +86.95187

Domain Status: clientTransferProhibited <https://icann.org/epp#clientTransferProhibited>

Así que estamos hablando de un servidor web con Apache, que físicamente pudiera estar en un datacenter en EU, con un registro de nombre en China.

Lo anterior también nos podría contribuir a desanonomizar si bien no explotando una vulnerabilidad, si al menos ubicando a los administradores a través de las empresas que contraten para alojar sus servidores web.

De igual forma a lo realizado por el archivo de Apache se podría realizar con los servidores de Nginx que se encuentran en la red y que representan mas del 80% del total de equipos instalados. Tal y como lo describe el siguiente artículo.

<https://blog.detectify.com/2020/11/10/common-nginx-misconfigurations/>

La dinámica sería similar realizando cada una de las pruebas mencionadas en este artículo vs cada uno de los servidores nginx encontrados y tratar de encontrar la mayor información posible. Tal y como se mostró en un ejemplo haciendo un disclosure de información de la IP “real” del servidor.

Ya con versiones de los productos instalados, bastaría con encontrar vulnerabilidades que pudieran ser explotadas ya sea para lograr un tema de divulgación de información, ejecución remota de código o inclusive tomar el control del equipo, para poder tener mayores detalles del servidor y conexiones con otras infraestructuras, para el ejercicio que realicé podría tomar algo como lo siguiente:

Producto	Possible Desanonimización	CVE_Asociado
Apache/2.4.34 (Win32) OpenSSL/1.1.1	Si	CVE-2019-9020 CVE-2019-9021 CVE-2019-9023 CVE-2019-9024
nginx/1.16.1	Si	CVE-2018-12886
nginx/1.16.1	Si	CVE-2018-12886
lighttpd/1.4.53	Si	CVE-2008-1270
nginx/1.16.1	Si	CVE-2018-12886
thttpd	Si	https://www.exploit-db.com/exploits/23306
nginx/1.16.1	Si	CVE-2018-12886
nginx/1.16.1	Si	CVE-2018-12886

Donde tomando todo lo anterior podríamos tener en la mira uno de los dominios asociados con una temática o red de nuestro interés y comenzar a tratar de explotar aquellos servicios que ubicemos como vulnerables.

Como ya se ha dicho, la red TOR proporciona anonimato, mas no seguridad, ya que el instalar una versión vulnerable de un servicio que cuente con una vulnerabilidad en una red como ésta, representa siempre un riesgo importante.

En la siguiente sección explicaré como se podrían relacionar los dominios Onion con la infraestructura detrás de ellos, con lo cual, tomando uno de los nodos de la posible infraestructura, y realizando la explotación correspondiente podría llevar a desmantelar una determinada red de alguna de las temáticas ubicadas dentro de esta red. Tal como ha sucedido recientemente con la tienda de Tarjetas de Crédito Jokerstash, Silk Road, etc...

¿Cómo relacionar todos los dominios Onion obtenidos con las infraestructuras que los albergan?

Derivado de lo que pude leer y documentarme existen varias formas posibles de ubicar que dominios Onion pudieran estar asociados con una infraestructura determinada, de varias maneras:

- Por el certificado web que presentan si están en la modalidad HTTPS

Información sobre esta página - https://ezpo2kkovm3liams.onion/

General Multimedia Permisos Seguridad

Identidad del sitio web

Sitio web: ezpo2kkovm3liams.onion
Propietario: Este sitio web no suministra información sobre su propiedad.
Verificado por: Let's Encrypt
Caduca el: 29 de febrero de 2020

[Ver certificado](#)

Este dato del certificado es algo que se podría validar que exista el mismo hash en diversos sitios Onion, de ser así daría el indicio de que el conjunto de esos sitios con HTTP se encuentran albergados en un mismo sitio o están amparados por una misma entidad certificadora. Algo que sin duda nos da un indicio de que puede estar relacionado con alguna entidad que avale su autenticación.

Nombre de DNS l4c.pw
Nombre de DNS liams.io

Información de clave pública

Algoritmo Elliptic Curve
Tamaño de clave 384
Curva P-384
Valor público 04:B9:F7:4B:85:48:35:DE:C9:96:63:0F:3B:E8:95:31:DE:2B:F8:82:82:7F:04:04:...

Cabe mencionar que el certificado digital, también nos puede dar algo de información valiosa al momento de indagar un sitio *.onion, como por ejemplo otros posibles alias del servidor o aquellos subdominios que contempla el certificado.

**Nombres alternativos
del sujeto**

Nombre de DNS	bitmessage.ch
Nombre de DNS	mail.bitmessage.ch
Nombre de DNS	www.bitmessage.ch

**Información de clave
pública**

Algoritmo	RSA
Tamaño de clave	2048
Exponente	65537
Módulo	8B:B3:DA:CA:0D:D2:9D:AF:C6:AC:5F:CA:B1:B8:17:18:67:AC:B9:38:04:A6:56:6...

- **Por similitudes de puertos abiertos en un servidor, y adicionalmente que las versiones de SW correspondan.**

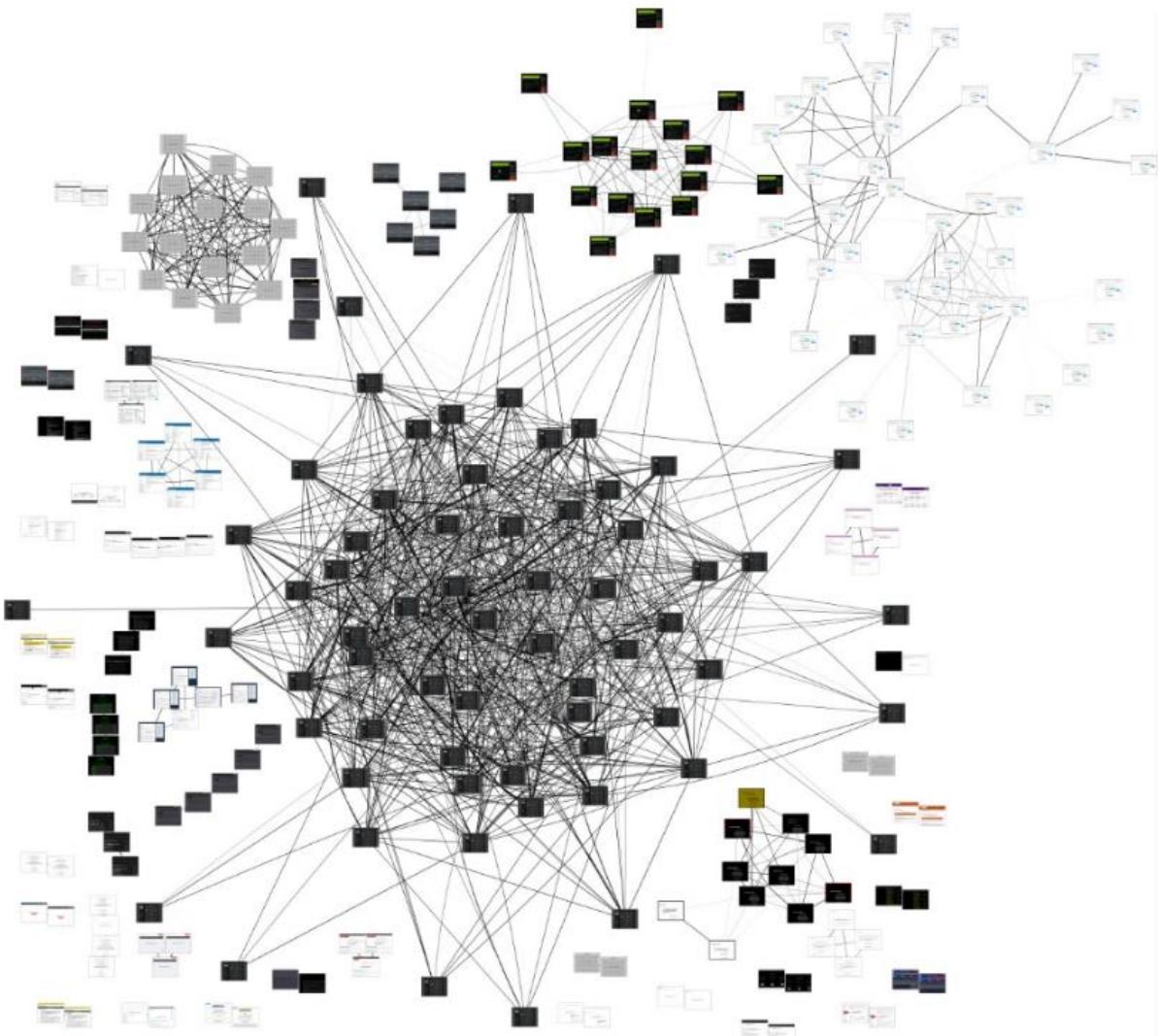
Este tipo de relación puede ser un poco burda e inclusive imprecisa, ya que el hecho de que un servidor tenga los mismos puertos y versiones en diferentes sitios *.onion, pudiera no garantizar al 100% la relación entre ellos, ya que como hemos visto antes, hacerlo de esta forma, nos llevaría a relacionar de forma directa o entretegidos muchos de los servidores que pudieran tener la misma versión de Apache o Ngix, así como sólo tener el puerto 80.

No es del todo desechable esta posibilidad, pero considero que antes de realizar esta relación se pudiera realizar un proceso previo, tal y como se describe en el siguiente punto.

- **Por el contenido de los sitios web que albergan.**

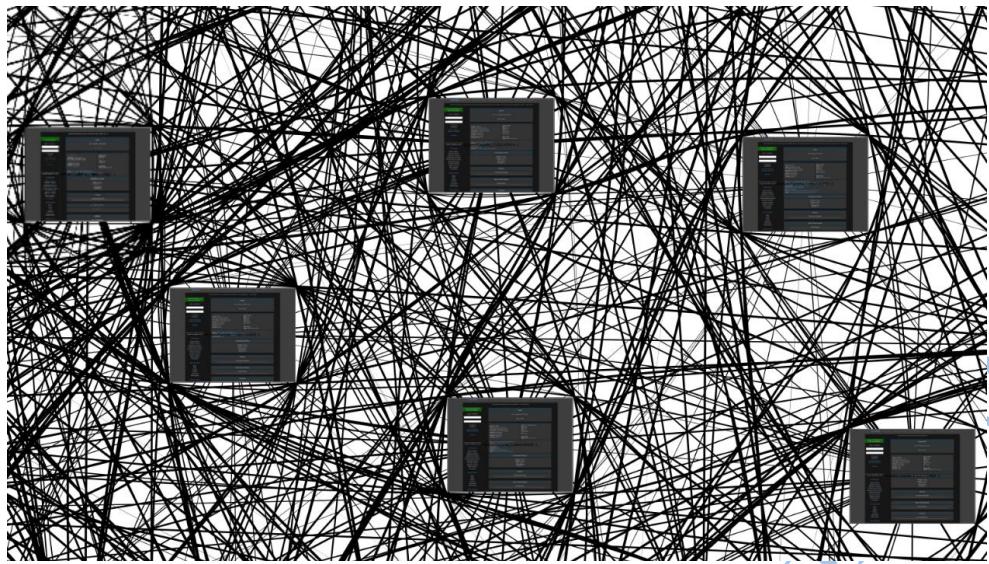
Otra forma de relacionar los sitios web *.onion con las infraestructuras que los albergan es por el contenido que almacenan, hemos visto que a pesar de que existen múltiples nombres de servicios, estos pudieran residir en un o varios servidores del mismo cluster, aunque con diferente nomenclatura de dominio *.onion, esta es la relación que yo pude obtener a través de la comparativa realizada con la aplicación de page-compare que descargamos inicialmente y con la cual me puede generar la representación en un diagrama de relación de entidades, el como se pudieran relacionar unos sitios con otros con base a la similitud del contenido que albergan.

Tal y como podemos observar en algunas de las siguientes imágenes obtenidas a través de este método:



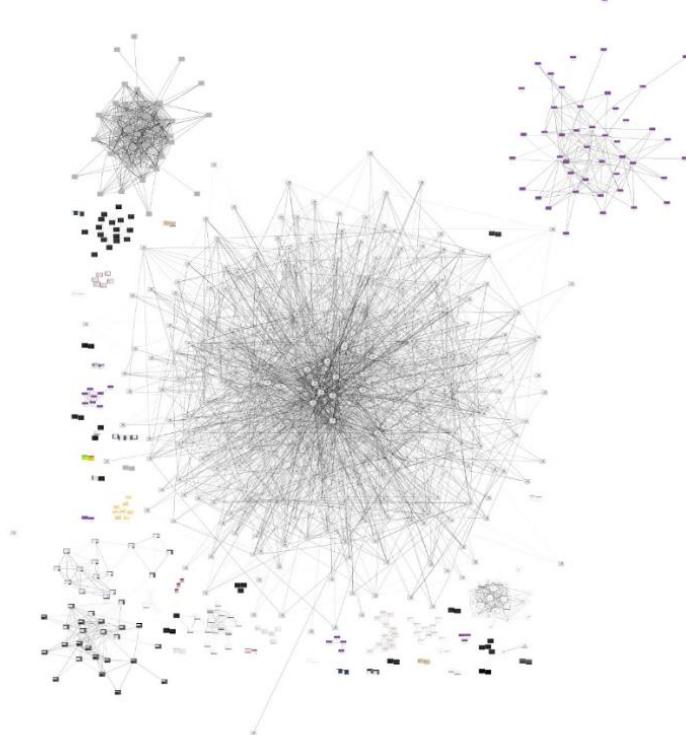
Esta es una imagen de una red de relaciones de sitios *.onion elaborado a partir de las similitudes del contenido de sus páginas principales y con ello comenzar a realizar una relación entre los diversos dominios .onion y el contenido de los sitios que exponen.

Si realizamos el zoom al centro del gráfico anterior, podríamos observar como es que todas las pantallas capturadas en la etapa de procesamiento de información son muy parecidas, aunque cada una de ellas representa un dominio *.onion distinto.

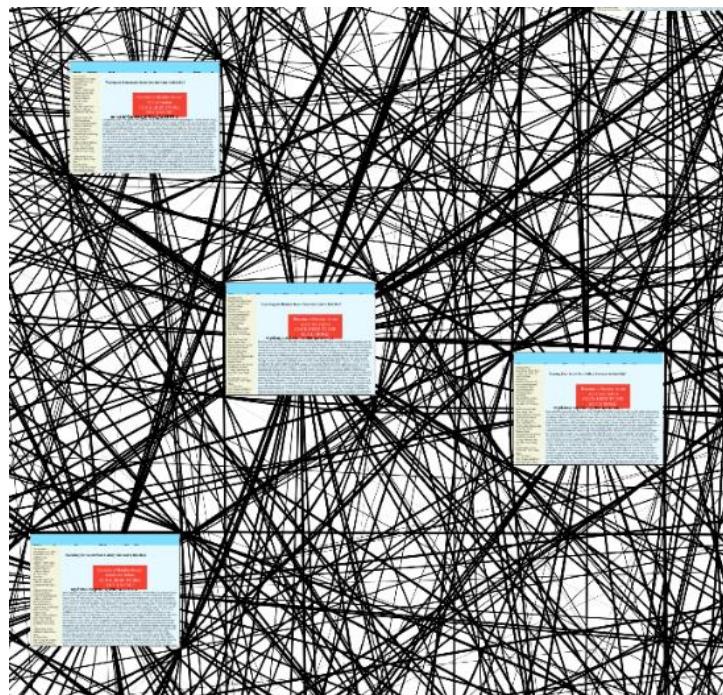


Esta imagen es un extracto del centro de la imagen anterior y aquí claramente se puede observar como la página expuesta en cada nodo (o dominio .onion) es muy similar, por lo que podríamos quizá inferir que se trate de un mismo servidor o conjunto de servidores proporcionando el mismo contenido inclusive en diferentes partes del mundo, pero la relación se dá por el contenido de la página que suministran.

Así podríamos observar estructuras mas complejas como la mostrada a continuación:



Donde se puede observar, de una forma similar una red de relaciones densamente poblada en el centro, contenido muy similar entre esos nodos, y en relación con algunos nodos exteriores que indicaría que el contenido de la página tiene algunas variaciones a las páginas centrales, pero guardan aún algún grado de similitud entre ellas.



Si comenzamos a combinar al menos estos 3 criterios de relación:

- Certificados digitales
- Servicios expuestos
- Similitud de contenido

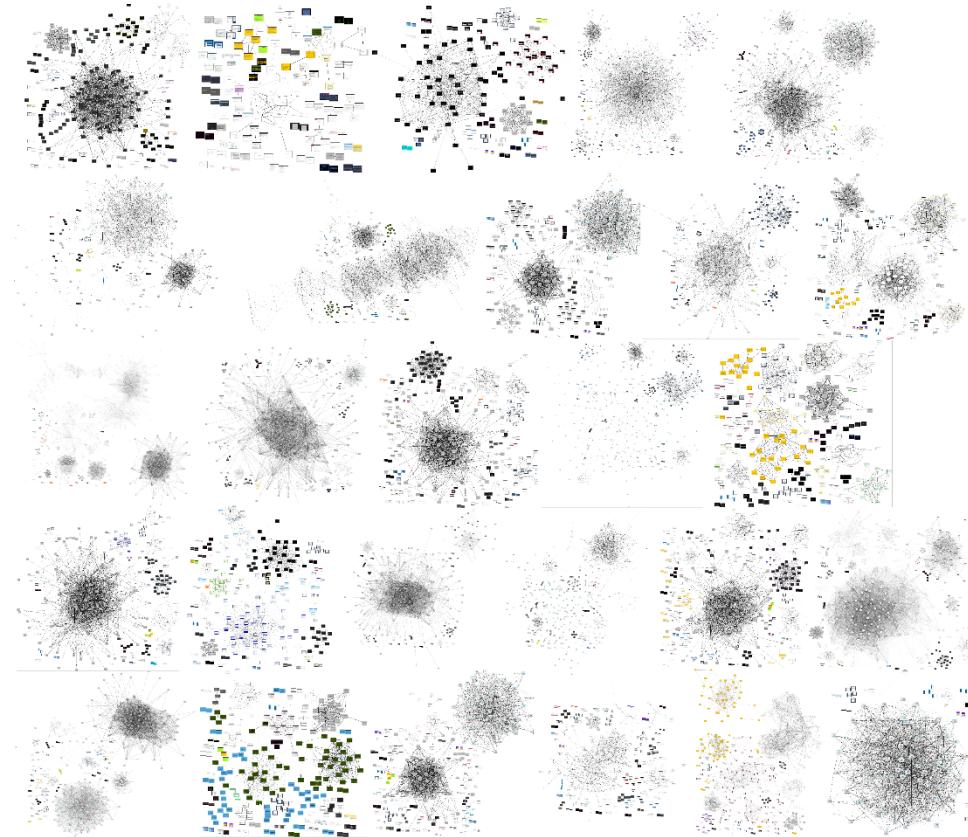
Es probable que la identificación de la infraestructura detrás de los mismos, sea aún mas precisa y asertiva al momento de buscar y/o detectar un posible punto vulnerable.

Ya con todos los elementos anteriormente obtenidos ha sido posible la creación de un mapa el cual es una representación temporal de la relación de los diversos dominios encontrados en la Dark Web, y con ello ubicar el tipo de contenido que proporcionan, lo anterior para darnos una idea simplemente de lo que podríamos encontrar en ella, sin tener que entrar en tanto detalle.

A continuación mostraré 2 mapas uno solamente con la relación de los dominios bajo el procedimiento anteriormente descrito (similitudes entre contenidos de dominios) y posteriormente otro que a manera de densidades, pudiera mostrar el tipo de contenidos que podemos encontrar en la Dark Web.

NOTA IMPORTANTE: El mapa de temáticas no tiene como objetivo incentivar a buscar el detalle de los contenidos, si bien en la Dark Web no todo tipo de contenido es ilícito, se recuerda que, si muchos sitios se encuentran aquí, es por que precisamente requieren una capa de anonimato por que distribuyen contenido que incumple con legislaciones en varios países, por lo que insto a no visitarlos, ni por curiosidad.

¿ Cómo se relacionan los diversos dominios encontrados en la Dark Web, derivado de este ejercicio?

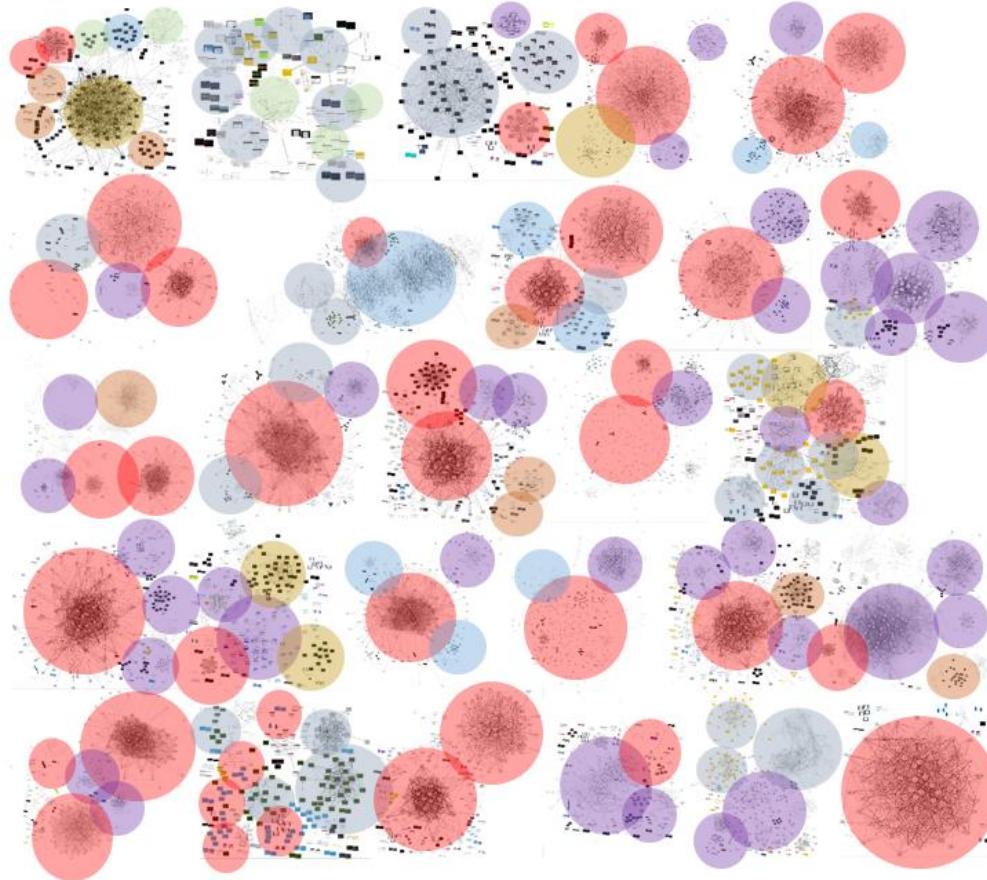


El mapa que se encuentra a la izquierda es la representación de todos los dominios recolectados para efectos de este trabajo.

Como ya he dicho anteriormente logré relacionar al menos alrededor de 25 mil dominios debido a la cantidad y tiempo dispuesto para realizarlo sin embargo es importante mencionar que en todos los recuadros que lo componen se pueden observar concentraciones importantes de infraestructuras para un dominio determinado.

En la sección de anexos se podrán observar estas imágenes con un poco mas de ampliación para observar mejor las relaciones de todos los dominios, como resultado de lo recopilado. El presente es una representación de todos estos dominios juntos en la galaxia de la DW interpretados como sólo un muestreo de ella.

¿Qué tipo de contenido observamos en los dominios recopilados?



El mapa superior muestra una clasificación de muy alto nivel en donde trato de englobar las temáticas de los sitios recopilados para este ejercicio, como se puede ver es el mismo mapa presentado en la página anterior, pero en esta ocasión con una clasificación de estilo densidad para mostrar el impacto de las infraestructuras y cantidad de sitios relacionados de una temática determinada. Cabe mencionar que todos los sitios de una determinada categoría tienen algún grado de relación ya bien sea con otros sitios de la misma temática o de temática distinta. Mismo que no puede ser observable de forma simple en este mapa por la complejidad de las mismas.

7. Conclusiones

Con base en todo lo anterior, podemos inferir que si es posible desanonomizar una infraestructura que alberge uno o varios de los dominios de la red de TOR, ya sea bien por una mala configuración y a través de la explotación de alguna vulnerabilidad.

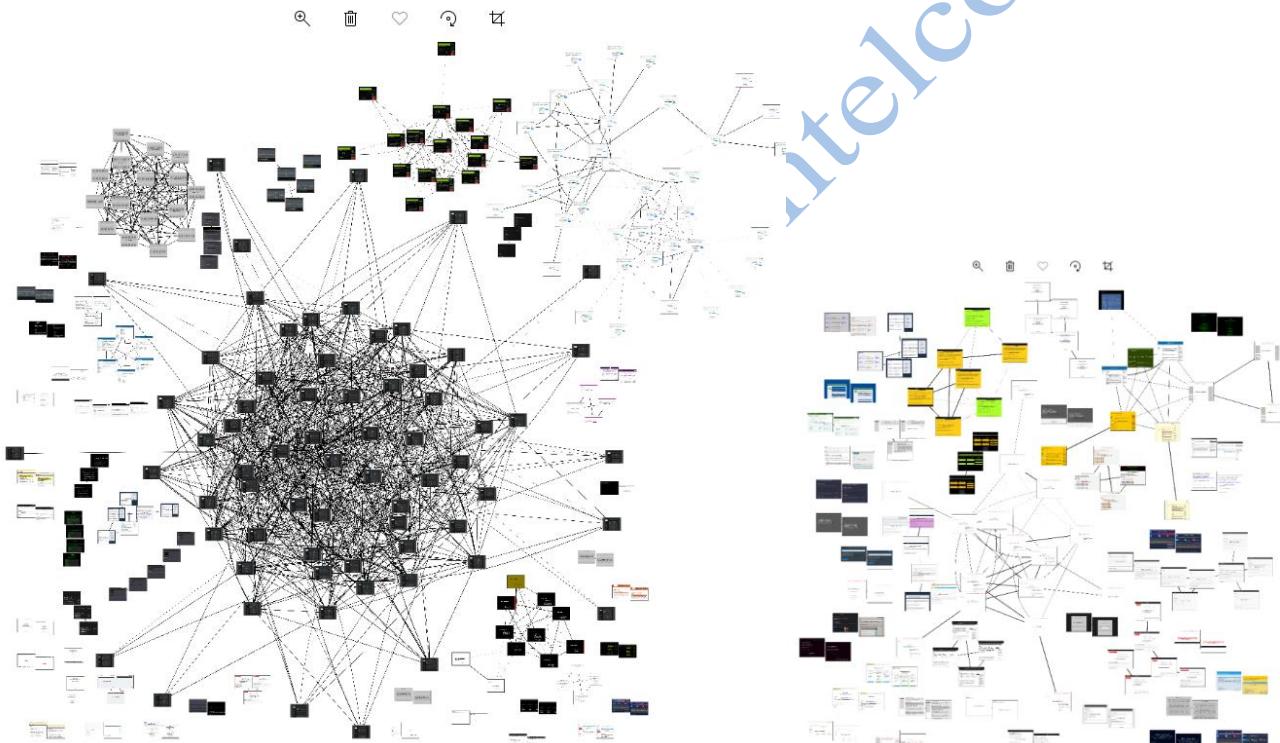
Existen otros métodos para realizarlo y que no están comprendidos en el alcance de esta investigación como lo es la instalación de nodos intermedios o de salida para monitorear tráfico y peticiones de dominios y que son realizados por empresas gubernamentales.

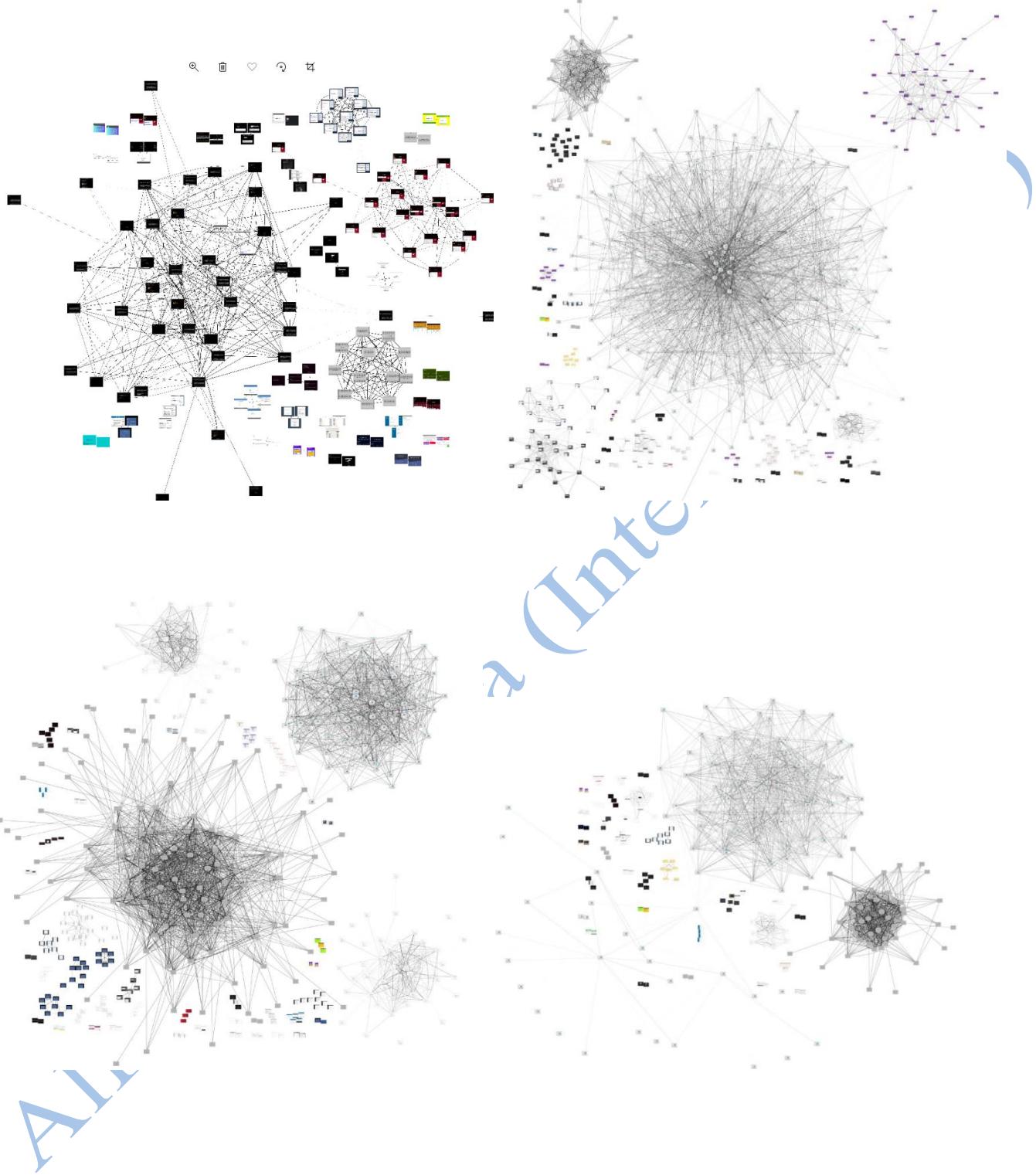
Una de las principales barreras que enfretamos al momento de realizar éste tipo de ejercicios es justamente la de lo efímero que son algunos servicios y/o servidores en la red TOR, inclusive hay quienes cambian el nombre del Hidden Service para ubicarlos nuevamente. Lo cual, requiere que el analista o equipos de analistas inviertan mucho tiempo y seguimiento muy puntual a esta actividad.

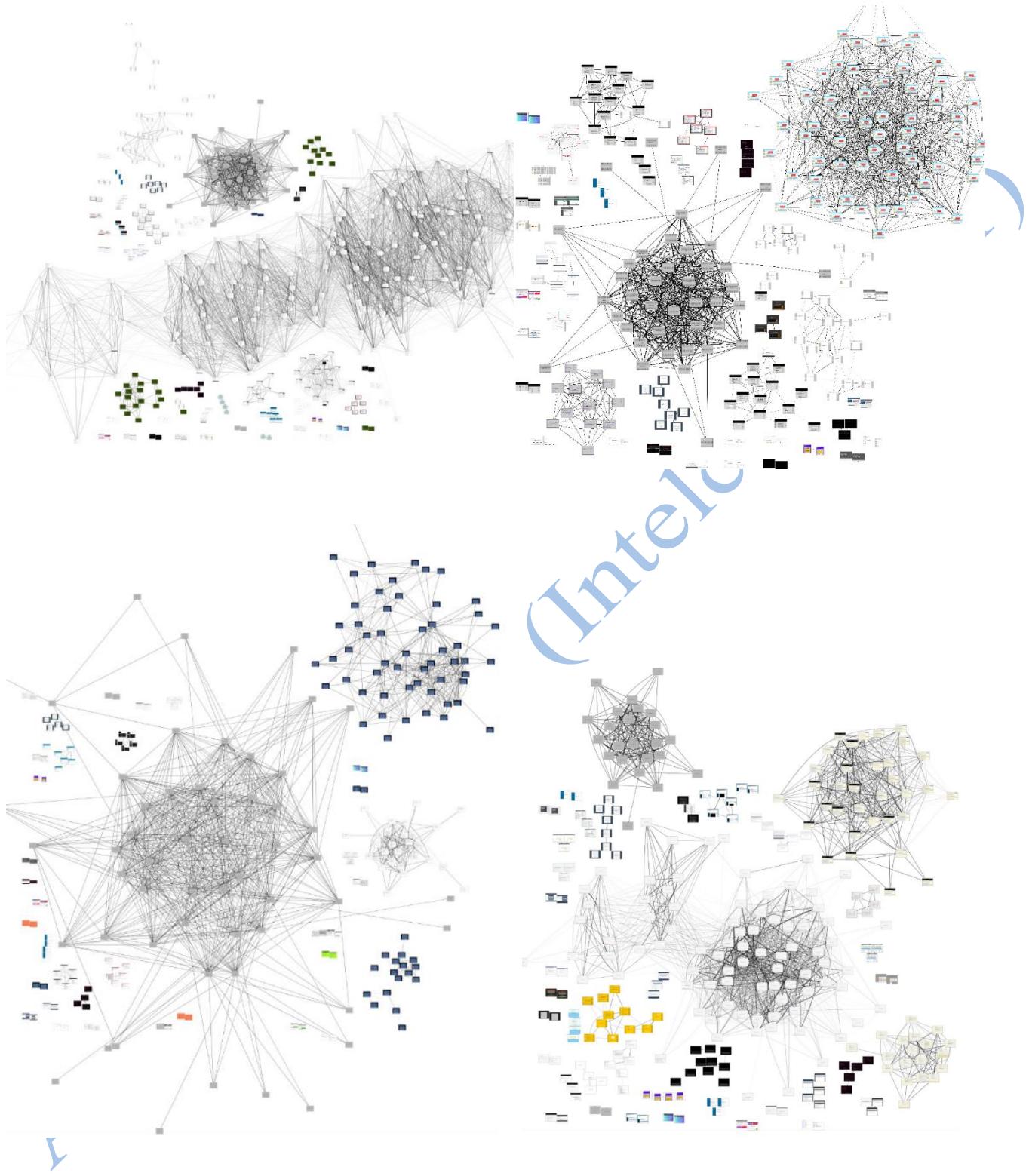
8. Anexos

A continuación se detallan cada uno de los mapas que componen la galaxia mostrada de manera macro en el desarrollo del mapa de relaciones y que por el tamaño de la muestra no fue posible mostrarlo por secciones en los incisos anteriores.

Sin embargo se anexan para referencia de una vista mas amplia de los temas de relación que existen entre ellos.







9. Referencias

1.- Reportes de Onion Scan 2016

<https://mascherari.press/onionscan-report-may-2016-technologies-used-by-onion-services/>

2.- Dark Web Map

<https://www.hyperiongray.com/dark-web-map/>

3.- Content and popularity analysis of Tor Hidden Services

<https://arxiv.org/pdf/1308.6768.pdf>

4.- Exposing the Public IPs of TOR Services Through SSL Certificates

<https://www.netsparker.com/blog/web-security/exposing-public-ips-tor-services-through-ssl-certificates/>

5.- Darkweb + Python: discover, analyze and extract information from hidden services

<https://dev.to/jmortega/darkweb-python-discover-analyze-and-extract-information-from-hidden-services-55p9>

6.- Page Compare Crawler Tool

<https://github.com/TeamHG-Memex/page-compare>

7.- Splash server HTTP API

<https://splash.readthedocs.io/en/stable/api.html>

8.- Apache Status Module

https://www.datadoghq.com/blog/collect-apache-performance-metrics/#:~:text=Apache%20web%20server%20exposes%20metrics,mod_status%20in%20your%20configuration%20file.

9.- Common Nginx misconfigurations that leave your web server open to attack

<https://blog.detectify.com/2020/11/10/common-nginx-misconfigurations/>

10.- Onion Search GitHub Repo

<https://github.com/megadose/OnionSearch>

11.- Metrics TOR Project

<https://metrics.torproject.org/>