

UNIVERSIDAD CATÓLICA DE MURCIA

CAMPUS INTERNACIONAL DE CIBERSEGURIDAD

MÁSTER EN CIBERINTELIGENCIA — 12^a Edición



**Análisis técnico de infraestructuras .onion
vinculadas a RaaS y APTs en la Dark Web**

Tesis presentada por

Judit López Jiménez

*Tesis presentada al **Campus Internacional de Ciberseguridad**
en cumplimiento parcial de los requisitos para la obtención del título de
Máster en Ciberinteligencia (12^a Edición)*

28 de agosto de 2025

Abstract

Este Trabajo de Fin de Máster se centra en el análisis técnico de los dominios `.onion` utilizados en campañas de ransomware, con especial atención a su rol dentro del ecosistema Ransomware-as-a-Service (RaaS) y su posible vinculación con grupos de Amenazas Persistentes Avanzadas (APT). El estudio aborda el fenómeno desde una perspectiva operativa y estructural, considerando tanto la arquitectura de la red Tor (The Onion Router) como las características técnicas expuestas por los servicios ocultos.

A través de una metodología híbrida que combina técnicas de inteligencia de fuentes abiertas (OSINT), fingerprinting de servicios, análisis de encabezados HTTP, favicons y certificados TLS, se construyó un dataset representativo de infraestructuras activas relacionadas con actividades maliciosas. Esta aproximación permitió identificar patrones de comportamiento, configuraciones replicadas y elementos reutilizados, lo que sugiere la existencia de ecosistemas técnicos compartidos o gestionados de forma centralizada, posiblemente mediante paneles de administración comunes o entornos de despliegue automatizados.

Además, el trabajo explora el potencial de técnicas de desanonimización parcial, fundamentadas en errores de configuración, coincidencias criptográficas y reutilización de recursos. Estas técnicas, si bien no permiten una identificación directa, sí aportan indicios valiosos para la atribución y correlación de servicios ocultos.

Los resultados de esta investigación no solo evidencian un alto grado de estandarización técnica en la Dark Web, sino que también ofrecen un marco metodológico aplicable a futuras labores de análisis, atribución, seguimiento e interrupción de infraestructuras delictivas en entornos anónimos y descentralizados. Se concluye que, incluso en contextos diseñados para maximizar el anonimato, es posible extraer información estructural útil mediante el análisis técnico riguroso y el cruce inteligente de fuentes de información.

Agradecimientos

Agradezco profundamente a mi tutor de este Trabajo de Fin de Máster, Francisco Rodríguez, por haberme guiado en el fascinante ámbito del análisis técnico de la Dark Web. A lo largo de este proceso, he adquirido conocimientos valiosos, tanto técnicos como personales, que me han permitido crecer profesionalmente. Le estoy especialmente agradecida por su apoyo y por recordarme la importancia de cuidar los pequeños detalles en cada paso del trabajo diario.

También deseo expresar mi sincero agradecimiento a todos los profesores que me han acompañado a lo largo del máster, por compartir generosamente su conocimiento y por su dedicación. Mi experiencia en el campus ha sido muy enriquecedora, gracias al compromiso del profesorado y al compañerismo de mis compañeros, quienes hicieron que este recorrido fuera mucho más llevadero.

Finalmente, agradezco de todo corazón a mi familia, y en especial a mi pareja, Noel, por su amor, comprensión y paciencia a lo largo de este camino. Gracias por estar a mi lado incluso en los momentos en los que no pude dedicarles el tiempo que merecían. Rindo un sentido homenaje a mis padres, por su amor incondicional, sus sacrificios y sus bendiciones constantes, sin los cuales este logro no habría sido posible.

Índice general

Abstract	III
Agradecimientos	v
Índice general	VII
Índice de figuras	x
Índice de cuadros	xi
1. Introducción	1
1.1. Contexto	1
1.2. Objetivos del TFM	2
1.2.1. Objetivo principal	2
1.2.2. Objetivos específicos	3
1.3. Estado del Arte: RaaS, APTs y Dark Web	4
1.3.1. Modelo RaaS y su ecosistema en la dark web	4
1.3.2. Grupos APT: motivaciones y características	5
1.3.3. Infraestructuras .onion	5
1.4. Metodología	6
1.4.1. Herramientas utilizadas	6
1.4.2. Fuentes de información	7
1.4.3. Técnicas de análisis	7
1.5. Hipótesis de trabajo	8
2. Fundamentos técnicos	9
2.1. Tor	9

2.2.	Ahmia	11
2.3.	Project Discovery Suite	12
2.4.	Nuclei	12
2.5.	GoWitness	13
3.	Trabajo realizado	14
3.1.	Obtención de dataset	14
3.2.	Escaneo de servicios	16
3.2.1.	Encabezados HTTP	17
3.2.2.	ETags	18
3.2.3.	Favicons	20
3.2.4.	Certificados TLS	21
3.2.5.	Estado del servidor	22
4.	Hallazgos	23
4.1.	Resultados del análisis de encabezados HTTP	23
4.2.	Resultados del análisis de ETags	26
4.2.1.	Inc. Ransom - Portal principal	27
4.2.2.	Inc. Ransom - Blog secundario	29
4.2.3.	Lynx - Chat	29
4.2.4.	Lynx - Blog	34
4.2.5.	Análisis técnico: De INC a LYNX ransomware	34
4.3.	Resultados del análisis de favicons	35
4.4.	Resultados del análisis de certificados TLS en servicios .onion	37
4.5.	Resultados del análisis de /server-status	39
5.	Trabajo futuro	42
6.	Conclusiones	44
A.	Apéndice - Scripts	46
A.1.	Script <code>fingerprinting.py</code> : Análisis de encabezados HTTP	46
A.2.	Script <code>favicon.py</code> : Extracción y análisis de favicons	51
A.3.	Script <code>tls_extract.py</code> : Extracción de certificados TLS sobre Tor	55

B. Apéndice - Cuadros	63
B.1. Cuadro Favicon	63
Bibliografía	65

Índice de figuras

2.1.1.Esquema de la red Tor	10
2.2.1.Web de Ahmia	11
3.1.1.Servidor local GoWitness	17
3.2.1.Parámetro ETag en la cabecera de respuesta HTTP	20
4.2.1.Infraestructura atribuida al grupo LYNX/INC Ransom construida a partir de relaciones ETag	28
4.2.2.Portal web de Lynx Blog	30
4.2.3.Uso de herramienta dig en el dominio <code>lynxr.blog</code>	31
4.2.4.Portal web blog <code>incrans.com</code>	32
4.2.5.Información relacionada con la IP <code>193.46.217.98</code>	33
4.5.1.Captura de la página <code>/server-status</code> del grupo Cactus.	40

Índice de cuadros

4.1.1.URLs y conjuntos de cabeceras HTTP idénticos.	24
4.1.2.Servidores HTTP más frecuentes detectados en los servicios analizados.	26
4.2.1.Resumen de sitios .onion: URLs, valores ETag y consultas FOFA asociadas.	27
4.2.2.Hosts identificados para el ETag W/"3a-5Ed5M0Vt0oLDha0QsMEz0eQ4seo"	28
4.2.3.Host identificado del ETag "6622b950-1c0"	29
4.2.4.Host identificado del ETag "6739ab3c-322"	29
4.2.5.Hosts identificados para la IP 193.46.217.98	31
4.2.6.Resultados de resoluciones y detecciones en VirusTotal y fuentes asociadas.	31
4.2.7.Hosts identificados del ETag "66ad19f5-323"	34
4.3.1.Resumen de favicons: URLs, hashes Murmur3 y consultas FOFA asociadas.	37
4.4.1.Análisis y clasificación de certificados TLS en servicios .onion	39
4.5.1.Resumen de servicios identificados en el host 5.175.142.66.	41
B.1.1Resumen de favicons: URLs, hashes Murmur3 y consultas FOFA asociadas.	64

1

Introducción

1.1 CONTEXTO

En los últimos años, el ransomware ha evolucionado significativamente hasta consolidarse como una de las amenazas más disruptivas y sofisticadas en el ámbito de la ciberseguridad global. Su impacto trasciende sectores e industrias, afectando tanto a entidades gubernamentales como a empresas privadas y usuarios. Esta amenaza se ha agravado con el auge del modelo **Ransomware as a Service (RaaS)**, que permite a atacantes con pocos conocimientos técnicos lanzar ataques sofisticados utilizando una aplicación como servicio. RaaS introduce una lógica de economía colaborativa en el cibercrimen, donde desarrolladores de ransomware alquilan su software malicioso a afiliados a cambio de una parte del rescate obtenido, profesionalizando y escalando la actividad delictiva.

En paralelo, la red Tor —una infraestructura que permite el acceso anónimo a Internet mediante el enrutamiento cifrado del tráfico a través de múltiples nodos— y sus servicios ocultos proporcionan una capa adicional de anonimato, descentralización y resiliencia a los actores maliciosos. Esta infraestructura, diseñada originalmente para proteger la privacidad de los usuarios, ha sido aprovechada para ocultar operaciones, evadir la detección y garantizar la continuidad de actividades ilícitas de grupos delictivos y de **Amenazas Persistentes Avanzadas (o APTs, por sus siglas en inglés)**, es decir, grupos organizados y capacitados que llevan a cabo ataques dirigidos y sostenidos a lo largo del tiempo. En este entorno, los sitios `.onion` —que son dominios accesibles únicamente a través de la red Tor y que permiten mantener en el anonimato tanto al servidor como al visitante— se utilizan como plataformas para la publicación de blogs de filtración de datos, la distribución y venta de herramientas maliciosas, el almacenamiento de información exfiltrada y el establecimiento de canales de comunicación cifrados entre atacantes y víctimas.

El presente trabajo se sitúa en la **intersección entre estas dos dimensiones**: el ransomware como fenómeno técnico y económico, y la red Tor como ecosistema de soporte con el objetivo de analizar cómo las infraestructuras `.onion` son utilizadas dentro de campañas de ransomware. En particular, se estudia su papel como elementos clave en las fases de extorsión, publicación de datos robados y contacto con las víctimas. Asimismo, se exploran los posibles vínculos entre estas infraestructuras y actores APT, considerando patrones de reutilización de dominios, tácticas comunes y relaciones de afiliación dentro del modelo RaaS. Este enfoque permite aportar una visión más completa sobre las dinámicas operativas y estratégicas del cibercrimen moderno, con implicaciones para la atribución, la prevención y la respuesta ante incidentes.

Como parte del análisis, se abordará también un intento técnico de **desanonimización** parcial de varios servicios `.onion` relacionados con campañas de ransomware activas o históricas. Este esfuerzo se centra en el estudio de características técnicas del protocolo Tor, así como en la recopilación y correlación de datos de fuentes abiertas (OSINT), metadatos de infraestructura, certificados TLS, huellas digitales de servidores y posibles errores de configuración que puedan exponer pistas sobre la ubicación o identidad de los operadores. Aunque se reconoce la dificultad inherente a esta tarea, el objetivo es evaluar la viabilidad de técnicas que permitan reducir el grado de anonimato de ciertos servicios ocultos, contribuyendo al entendimiento de sus operaciones y facilitando su atribución.

1.2 OBJETIVOS DEL TFM

A continuación, se presentan los objetivos que estructuran y orientan este estudio.

1.2.1 OBJETIVO PRINCIPAL

El objetivo principal de este Trabajo de Fin de Máster es analizar técnica y estructuralmente los dominios `.onion` vinculados a actividades de ransomware para identificar patrones de infraestructura, relaciones con plataformas RaaS y posibles vínculos con grupos APT. Asimismo, se evaluará el grado de anonimato ofrecido por estas infraestructuras mediante el estudio de configuraciones técnicas, errores comunes y técnicas de fingerprinting, con el fin de explorar posibles vectores de desanonimización o correlación entre servicios aparentemente independientes.

1.2.2 OBJETIVOS ESPECÍFICOS

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

- **Recolectar y clasificar dominios .onion activos:** Se desarrollará un proceso sistemático de recopilación de dominios .onion que estén activos y relacionados con actividades de ransomware. La clasificación se basará en criterios como tipología del contenido y la finalidad del servicio.
- **Analizar servicios desplegados (HTTP, FTP, SMTP, etc.):** Se investigará qué servicios están expuestos en los dominios recopilados, tanto a nivel de capa de aplicación (por ejemplo, servidores web HTTP/HTTPS, correo electrónico, FTP) como en cuanto a tecnologías utilizadas (frameworks, certificados TLS, headers, fingerprints). Esto permitirá inferir posibles configuraciones comunes y errores explotables.
- **Mapear hosting compartido o patrones de comportamiento técnico:** Se buscarán indicios de infraestructura compartida entre distintos dominios, como el uso de certificados TLS similares, software y versiones coincidentes, tiempos de respuesta similares o correlaciones en las cabeceras de respuesta. Este mapeo puede revelar la existencia de servicios operados por un mismo actor o grupo, y aportar pistas para la atribución.
- **Identificar nexos entre estos servicios y grupos de ransomware:** A través del análisis de contenido, referencias cruzadas, y búsquedas OSINT, se tratará de establecer conexiones entre los servicios .onion identificados y grupos de ransomware conocidos.
- **Evaluar técnicas de desanonimización en base a malas configuraciones:** Se explorarán distintas metodologías para reducir el anonimato de los servicios .onion, evaluando desde errores de configuración hasta técnicas pasivas y activas basadas en fingerprinting, análisis de tráfico, correlación temporal, o uso de servicios externos expuestos.
- **Estudiar vínculos con grupos APT y plataformas RaaS:** Finalmente, se buscarán relaciones estructurales y operativas entre las infraestructuras .onion identificadas y los ecosistemas RaaS, a través del análisis de reportes técnicos, informes de inteligencia de amenazas y documentación pública especializada.

1.3 ESTADO DEL ARTE: RAAS, APTS Y DARK WEB

La línea entre cibercrimen financiero y ciberespionaje se ha vuelto cada vez más difusa. Grupos APT han adoptado tácticas asociadas al ransomware para sabotaje o distracción, mientras que actores criminales emplean técnicas avanzadas propias de operaciones estatales. La dark web actúa como punto de encuentro entre ambos, facilitando el acceso a herramientas, servicios y datos comprometidos. Esta convergencia refuerza la necesidad de estudiar técnicamente las infraestructuras .onion como vectores clave en campañas híbridas y difíciles de atribuir.

1.3.1 MODELO RAAS Y SU ECOSISTEMA EN LA DARK WEB

El modelo de *Ransomware as a Service* (RaaS) ha revolucionado la economía del cibercrimen al introducir una estructura empresarial en la distribución y operación del ransomware. En este modelo, los desarrolladores de ransomware —también conocidos como *operators*— diseñan y mantienen el software malicioso, mientras que terceros —conocidos como *affiliates*— lo utilizan para comprometer sistemas y gestionar la fase operativa del ataque. A cambio, los beneficios económicos obtenidos tras el pago de los rescates se dividen entre ambos actores, normalmente en proporciones que oscilan entre el 70% y 90% para el afiliado y el resto para el operador [8, 4].

Este modelo ha logrado reducir significativamente las barreras de entrada al cibercrimen, permitiendo que individuos con escasos conocimientos técnicos participen en campañas altamente sofisticadas. Los foros clandestinos y plataformas alojadas en la dark web, accesibles a través de dominios .onion, actúan como puntos de encuentro para el reclutamiento de afiliados, el intercambio de conocimientos, la venta de credenciales comprometidas y la distribución de kits de ransomware. Ejemplos notables de plataformas RaaS incluyen a grupos como LockBit, Conti, BlackCat y REvil, cuyas operaciones han causado interrupciones significativas a infraestructuras críticas a nivel mundial.

La profesionalización del cibercrimen bajo el modelo RaaS se manifiesta también en el soporte técnico ofrecido a los afiliados, paneles de control personalizados para gestionar víctimas, y mecanismos automatizados para el cifrado de archivos y la entrega de notas de rescate. Este entorno, sumado al anonimato proporcionado por la red Tor y el uso de criptomonedas, hace del RaaS un modelo altamente escalable y difícil de erradicar.

1.3.2 GRUPOS APT: MOTIVACIONES Y CARACTERÍSTICAS

Los grupos de Amenazas Persistentes Avanzadas (APT, por sus siglas en inglés) son actores ciberneticos altamente organizados y respaldados en muchos casos por Estados. Su actividad no se orienta únicamente al beneficio económico, sino que responde a motivaciones geopolíticas, estratégicas o militares. Sus objetivos incluyen el espionaje prolongado, la infiltración de infraestructuras críticas, la obtención de propiedad intelectual y la manipulación de información para influir en decisiones políticas o económicas [12, 3].

A diferencia de los grupos cibercriminales convencionales, los APT operan con una elevada persistencia en el tiempo, emplean recursos avanzados y están altamente especializados. Utilizan un conjunto amplio de Tácticas, Técnicas y Procedimientos (TTPs), muchos de los cuales están recogidos en el marco MITRE ATT&CK. Entre estas técnicas se encuentran la explotación de vulnerabilidades zero-day, el spear phishing, el movimiento lateral dentro de redes comprometidas, y sofisticados mecanismos de evasión y persistencia [2].

Algunos grupos APT también han utilizado ransomware como herramienta táctica, no con fines de lucro directo, sino como forma de sabotaje o desestabilización. Es lo que se ha denominado *pseudo-ransomware*, en el que el cifrado y las notas de rescate simulan una motivación económica, cuando en realidad el objetivo es causar daño operativo. Casos como NotPetya (2017), atribuido al grupo Sandworm, ilustran este enfoque, al haber sido diseñado para destruir sistemas críticos bajo la apariencia de ransomware [3].

1.3.3 INFRAESTRUCTURAS .ONION

La red Tor proporciona un entorno de comunicación anónima mediante el uso de múltiples capas de cifrado y enrutamiento a través de nodos voluntarios distribuidos globalmente. Dentro de esta red, los servicios ocultos accesibles mediante dominios .onion ofrecen anonimato tanto al cliente como al servidor, lo que dificulta la trazabilidad de las comunicaciones y la localización física de los servicios [10].

Estas infraestructuras han sido aprovechadas para albergar una amplia variedad de servicios ilícitos, incluyendo páginas de filtración de datos robados, portales de ransomware, marketplaces de malware, servicios de mensajería cifrada, y centros de comando y control (C2) de botnets. El anonimato inherente a los servicios .onion representa una ventaja operativa para actores cibercriminales, ya que permite establecer canales de comunicación con víctimas, ofrecer soporte técnico, o publicar información sensible sin exponerse a una localización directa.

El estudio técnico de estas infraestructuras —por ejemplo, a través del fingerprinting de servicios, análisis de certificados, configuraciones de servidor o correlaciones de tráfico— permite vislumbrar patrones de operación, errores de implementación, o potenciales puntos de desanonimización que pueden ser aprovechados para la atribución o la interrupción de actividades maliciosas.

1.4 METODOLOGÍA

La metodología empleada en este trabajo combina técnicas de recolección automatizada, análisis técnico detallado y contextualización mediante fuentes de inteligencia abiertas. Dado el carácter anónimo y descentralizado de la red Tor, se diseñó un enfoque mixto que integra herramientas especializadas para la extracción de datos desde servicios ocultos, análisis pasivo y activo de las configuraciones técnicas de los servidores, así como la interpretación de estos datos mediante el contraste con fuentes OSINT e informes de ciberinteligencia.

1.4.1 HERRAMIENTAS UTILIZADAS

Para llevar a cabo el análisis técnico de los servicios `.onion`, se utilizaron diversas herramientas especializadas. En primer lugar, se empleó **Ahmia**, un motor de búsqueda diseñado específicamente para indexar servicios ocultos dentro de la red Tor [9]. También se integró el ecosistema de **Project Discovery**, un conjunto de utilidades de reconocimiento de código abierto orientadas al mapeo de superficies de ataque. En particular, se utilizaron herramientas como `httpx` y `nuclei` para automatizar la recolección de metadatos y realizar escaneos personalizados sobre los servicios detectados. **Nuclei** es un motor de escaneo basado en plantillas que posibilita análisis tanto pasivos como activos [11].

Para la documentación visual de estos sitios, se utilizó **GoWitness**, una herramienta capaz de capturar imágenes automatizadas de servicios web. Esta fue configurada para operar sobre Tor mediante un proxy, lo que permitió capturar evidencias sin interacción directa con las interfaces [13].

Además, se empleó **FOFA**, una plataforma de búsqueda de información expuesta en Internet, útil para identificar posibles correlaciones entre servicios `.onion` y su infraestructura visible. También se desarrollaron y utilizaron diversos **scripts en Python** para automatizar tareas de recopilación, análisis y procesamiento de datos obtenidos durante la investigación.

1.4.2 FUENTES DE INFORMACIÓN

Las fuentes de información utilizadas en este trabajo incluyen tanto recursos abiertos como documentación técnica especializada. En primer lugar, se recurrió a **motores de búsqueda en la red Tor**, como Ahmia, para identificar dominios `.onion` relevantes, con un enfoque en sitios relacionados con filtración de datos, ransomware y otras actividades maliciosas [10, 9]. También se consultaron **repositorios OSINT**, en particular feeds de inteligencia como AlienVault OTX y ThreatFox, que proporcionan indicadores de compromiso (IoCs) y listas continuamente actualizadas de servicios vinculados a malware y ransomware.

Por otro lado, se revisaron diversos **informes de threat intelligence** publicados por entidades como CrowdStrike, Palo Alto Networks (Unit 42) y ENISA, los cuales resultaron fundamentales para contextualizar los hallazgos técnicos y establecer posibles conexiones entre dominios `.onion`, campañas APT y plataformas RaaS. Finalmente, se utilizó **documentación técnica**, incluyendo la documentación oficial del proyecto Tor y artículos académicos sobre técnicas de fingerprinting y correlación de tráfico, con el fin de definir los límites técnicos del análisis y guiar el diseño metodológico del estudio.

1.4.3 TÉCNICAS DE ANÁLISIS

El análisis de los dominios `.onion` se desarrolló mediante una combinación de técnicas automatizadas y revisión manual. En primer lugar, se llevó a cabo la **recolección y clasificación de dominios**, utilizando herramientas automatizadas para su detección y una posterior categorización manual —utilizando regex— basada en la tipología del contenido, como filtración de datos, extorsión o servicios técnicos [9, 11]. Posteriormente, se aplicaron técnicas de **fingerprinting de servicios**, mediante utilidades como `httpx` y `nuclei`, que permitieron extraer cabeceras HTTP, tecnologías utilizadas, certificados digitales y firmas asociadas al comportamiento de los servidores, con el objetivo de identificar similitudes entre distintos portales [11].

Para complementar el análisis técnico, se incorporó una fase de **captura y análisis visual**, usando GoWitness configurado sobre Tor, lo que facilitó la obtención segura de capturas de pantalla de los servicios sin necesidad de interacción directa [13]. A continuación, se realizó una **correlación técnica** entre los dominios identificados, basada en elementos como coincidencias en certificados TLS, estructuras HTML o tiempos de respuesta, lo que permitió inferir posibles relaciones operativas o uso compartido de infraestructura. Finalmente, se llevó a cabo un **análisis OSINT cruzado**, en el cual los datos técnicos recolectados fueron contrastados con indicadores de compromiso (IoCs) presentes en bases de datos OSINT y con información

proveniente de análisis previos sobre campañas de ransomware y grupos APT.

1.5 HIPÓTESIS DE TRABAJO

A partir del análisis técnico de los dominios .onion vinculados a campañas de ransomware, se plantea la hipótesis de que existen patrones estructurales y operativos comunes que permiten identificar grupos de servicios gestionados de forma centralizada. Además, se evalúa la posibilidad de que errores de configuración y reutilización de recursos técnicos puedan comprometer, parcial o totalmente, el anonimato de estas infraestructuras, abriendo vías para su correlación y potencial atribución.

2

Fundamentos técnicos

En esta sección se describen en detalle los componentes técnicos utilizados a lo largo de la investigación.

2.1 TOR

Tor es una red diseñada para preservar el anonimato en las comunicaciones. En lugar de enviar los datos directamente del emisor al receptor, Tor los hace pasar por una serie de nodos intermedios —conocidos como **Guard**, **Middle** y **Exit**— formando un *circuito* cifrado por capas, al estilo de una cebolla (*onion routing*). Cada nodo solo conoce a su predecesor y a su sucesor, y descifra únicamente la capa que le corresponde, lo que impide que se pueda conocer simultáneamente el origen, el contenido y el destino del tráfico [10].

En la actualidad, Tor emplea direcciones .onion de tercera generación, basadas en criptografía ED25519, con identificadores de 56 caracteres, que refuerzan la seguridad respecto a versiones anteriores.

Los *Onion Services* permiten alojar sitios web dentro de la red Tor sin revelar la ubicación del servidor ni del usuario. Para lograrlo, el servidor publica su presencia en varios puntos de entrada llamados *introduction points*, y el cliente establece contacto a través de un nodo intermedio, el *rendezvous point*, que actúa como punto de encuentro. Gracias a este diseño, ninguna de las partes conoce la dirección IP de la otra. Además, la información sobre estos servicios se distribuye en directorios (HSDir) que se actualizan cada 24 horas, lo que dificulta rastrear o enumerar estos sitios ocultos.

Por diseño, Tor cambia automáticamente el circuito cada 10 minutos para evitar correlaciones de tráfico a largo plazo. Esto implica latencias más altas (normalmente entre 700 y 1200 ms) y sesiones de corta duración, lo cual plantea desafíos adicionales a la hora de realizar análisis o implementar herramientas de

monitorización.

El estudio se ejecutó en macOS con Homebrew. Los comandos siguientes instalan y gestionan el daemon de Tor; se añaden comentarios aclaratorios:

```
brew install tor # Instala Tor usando Homebrew  
brew services start tor # Inicia el servicio de Tor en segundo plano  
brew services list # Verifica estado de todos los servicios gestionados por Brew  
brew services stop tor # Detiene el servicio de Tor
```

Todas las conexiones salientes del laboratorio se enrutan mediante un proxy SOCKS5 en 127.0.0.1:9050. Se recomienda aislar el tráfico en una máquina virtual.

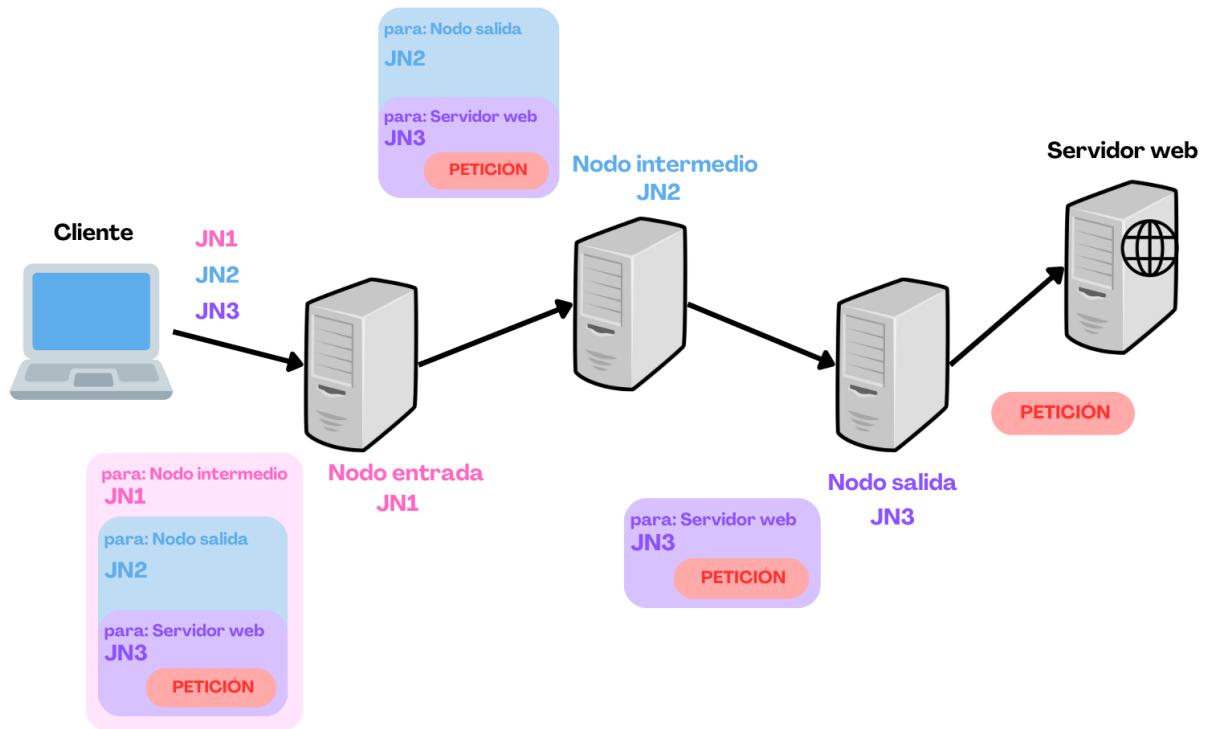


Figura 2.1.1: Esquema de la red Tor

2.2 AHMIA

Ahmia es un motor de búsqueda para servicios ocultos que combina exploración pasiva y envíos voluntarios. Cada resultado se filtra mediante listas de bloqueo y reglas de calidad que eliminan phishing, contenido ilegal y duplicados [9]. Para esta tesis se extrajeron 17 840 URLs usando su API pública REST, aplicando expresiones regulares para identificar términos afines a ransomware.

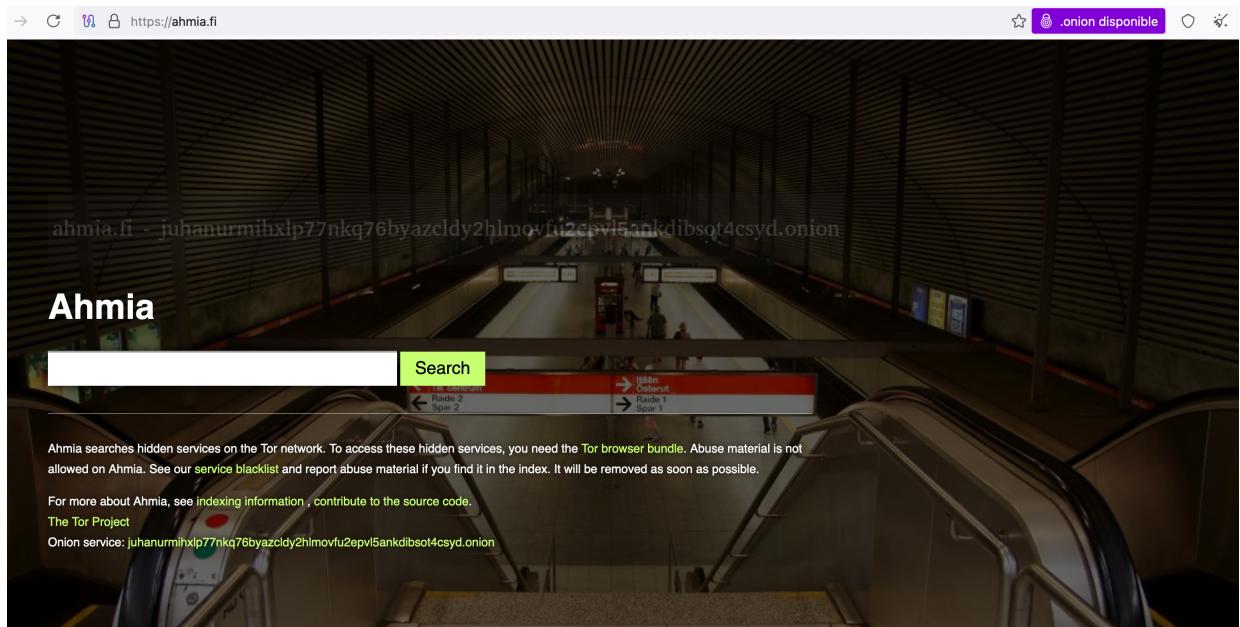


Figura 2.2.1: Web de Ahmia

2.3 PROJECT DISCOVERY SUITE

La suite **ProjectDiscovery** ofrece utilidades orientadas a *offensive security* y *recon* [11]. A continuación se destacan las usadas en este trabajo:

- **httpx**: Framework de fingerprinting HTTP/HTTPS con detección de títulos, status codes, tecnologías y soporte para JA3/JA3S. Útil para agrupar dominios por pila tecnológica.
- **nuclei**: Motor de escaneo basado en plantillas YAML. Se ejecutó con `-proxy socks5://127.0.0.1:9050` y un set de 9200 plantillas.

Gestor pdtm: Para mantener las versiones alineadas se utilizó `pdtm`, un wrapper escrito en Go que descarga los binarios precompilados y gestiona PATH. Requiere Go versión 1.24.2 o superior para instalarse correctamente. El comando de instalación es:

```
go install -v github.com/projectdiscovery/pdtm/cmd/pdtm@latest
```

Una vez instalado, puedes usar `pdtm` para instalar otras herramientas necesarias con comandos como:

```
pdtm install nuclei
```

2.4 NUCLEI

Nuclei es una de las herramientas más potentes de Project Discovery para realizar escaneos de seguridad. Esta herramienta utiliza plantillas YAML definidas por la comunidad o creadas a medida. Estas plantillas permiten realizar comprobaciones de vulnerabilidades conocidas, configuraciones inseguras, metadatos expuestos o cabeceras HTTP mal configuradas.

En el contexto de la investigación, Nuclei fue utilizado para evaluar técnicamente los servicios `.onion` seleccionados, con el objetivo de identificar posibles debilidades de seguridad que pudieran comprometer su anonimato o su infraestructura. El análisis se llevó a cabo a través del proxy TOR, garantizando que las solicitudes se realizaran de forma anónima. Ejemplo de ejecución básica:

```
nuclei -u http://exampleonionurl.onion -proxy socks5://127.0.0.1:9050 -t cves/
```

Las plantillas se pueden actualizar con:

```
nuclei -update-templates
```

2.5 GOWITNESS

GoWitness es una herramienta en línea de comandos diseñada para capturar capturas de pantalla automatizadas de sitios web. Se utiliza principalmente para documentar visualmente la apariencia y el estado de una página en un momento determinado, lo cual resulta especialmente útil cuando se analiza contenido efímero o variable como el alojado en la dark web.

Además de la captura de pantalla, GoWitness permite realizar escaneos automatizados de múltiples URLs, validar el estado de los servicios web y generar informes visuales. En este estudio, GoWitness se empleó para capturar la apariencia de sitios .onion accedidos vía proxy TOR local, permitiendo un análisis comparativo y visual de las distintas plataformas estudiadas.

La instalación puede realizarse a través de Go:

```
go install github.com/sensepost/gowitness@latest
```

Y una vez instalado, puede usarse para capturar una URL específica:

```
gowitness single --url http://exampleonionurl.onion --proxy socks5://127.0.0.1:9050
```

Una vez terminada la captura, el comando `gowitness report server` expone una UI en `localhost:7171`. Desde su API REST se recuperan capturas y cabeceras (véase Apéndice A).

3

Trabajo realizado

En esta sección se expone de manera detallada el trabajo desarrollado a lo largo de la investigación.

3.1 OBTENCIÓN DE DATASET

El primer paso ha consistido en la recopilación de un conjunto de direcciones `.onion`, es decir, URLs accesibles únicamente a través de la red anónima TOR. Para ello, se han utilizado motores de búsqueda diseñados específicamente para la navegación en la Dark Web, como **Ahmia**, que indexan servicios ocultos disponibles en esta red. A partir de Ahmia, se ha construido un *dataset* inicial con un total de **17.840 direcciones `.onion`**.

Con el objetivo de identificar sitios potencialmente vinculados a grupos de ransomware, se ha aplicado un proceso de filtrado sobre las URLs recolectadas. Para ello, se ha utilizado la herramienta `httpx`, incluida en la suite de *Project Discovery*. Esta herramienta permite realizar peticiones HTTP automatizadas y configurables, incluso a través de la red TOR.

El siguiente comando ha sido empleado para realizar el análisis:

```
httpx -l ahmia-list.txt -proxy socks5://127.0.0.1:9050 \
-mr "(?i)(lockbit|blackcat|alphv|ransomware|leak|ransom|raas)" \
-title -td -json -o ransomware-sites.json
```

Explicación del comando:

- `-l ahmia-list.txt`: archivo con las URLs de entrada.
- `-proxy socks5://127.0.0.1:9050`: enrutamiento de peticiones a través del proxy TOR local.

- **-mr**: expresión regular para identificar palabras clave relacionadas con ransomware.
- **-title**: extrae el título de las páginas.
- **-td**: detecta tecnologías web utilizadas en el sitio.
- **-json -o ransomware-sites.json**: guarda los resultados en formato JSON.

Este proceso ha tardado aproximadamente **1 hora y 50 minutos** y ha devuelto un total de **190 sitios posiblemente relevantes**.

Para facilitar el tratamiento de los resultados, se han extraído únicamente las URLs del archivo JSON generado:

```
cat ransomware-sites.json | jq -r '.url' > extracted_onion_urls.txt
```

Posteriormente, se ha utilizado nuevamente `httpx` para obtener capturas de pantalla, detectar tecnologías web y realizar un escaneo más detallado:

```
httpx -proxy socks5://127.0.0.1:9050 -ss -st 120 -title -td \
--headless-options="--proxy-server=socks5://127.0.0.1:9050" \
-l extracted_onion_urls.txt
```

Parámetros adicionales:

- **-ss**: realiza capturas de pantalla.
- **-st 120**: establece un *timeout* de 120 segundos.
- **--headless-options**: configura el navegador en modo *headless* con proxy TOR.

No obstante, se ha observado que la mayoría de sitios del dataset de Ahmia no estaban activos o no estaban relacionados con ransomware, por lo que se ha recurrido a otra fuente de información.

Como alternativa más precisa, se ha empleado la API pública del proyecto `ransomware.live`, que ofrece datos actualizados sobre grupos de ransomware activos [1]. La descarga de datos se ha realizado con el comando:

```
curl https://api.ransomware.live/v2/groups > ransomware_live_groups.json
```

Las direcciones .onion han sido extraídas con el siguiente filtro de jq:

```
jq -r '.. | strings | select(endswith(".onion"))' \
ransomware_live_groups.json > onions_ransomware_live.txt
```

Para realizar una inspección visual de los sitios web extraídos, se ha utilizado la herramienta **GoWitness**, la cual permite generar capturas de pantalla y explorar resultados mediante una interfaz web local [13].

```
gowitness scan file -f onions_ransomware_live.txt \
--chrome-proxy socks5://127.0.0.1:9050 --write-db
```

```
gowitness report server
```

Este comando levanta un servidor local que permite visualizar las capturas e información adicional sobre cada sitio .

3.2 ESCANEOS DE SERVICIOS

Una vez obtenida y depurada la lista de dominios vinculados a actividades de ransomware, se ha iniciado una segunda fase centrada en el análisis activo de los servicios que estos exponen. Esta etapa tiene como objetivo caracterizar técnica y estructuralmente los sitios identificados, extrayendo metadatos clave que permitan inferir relaciones entre distintas infraestructuras, detectar patrones de configuración comunes y, en última instancia, contribuir a la atribución o correlación de grupos cibercriminales.

Entre los elementos técnicos analizados se encuentran los encabezados HTTP, certificados TLS, tecnologías empleadas, cookies, favicons, así como capturas de pantalla para contextualizar visualmente cada dominio. Esta información no solo es útil para generar perfiles individuales por sitio, sino también para detectar agrupaciones o posibles plantillas compartidas que podrían evidenciar un backend común, un panel de administración clonado, o incluso una red de afiliados dentro de un modelo *Ransomware-as-a-Service* (RaaS).

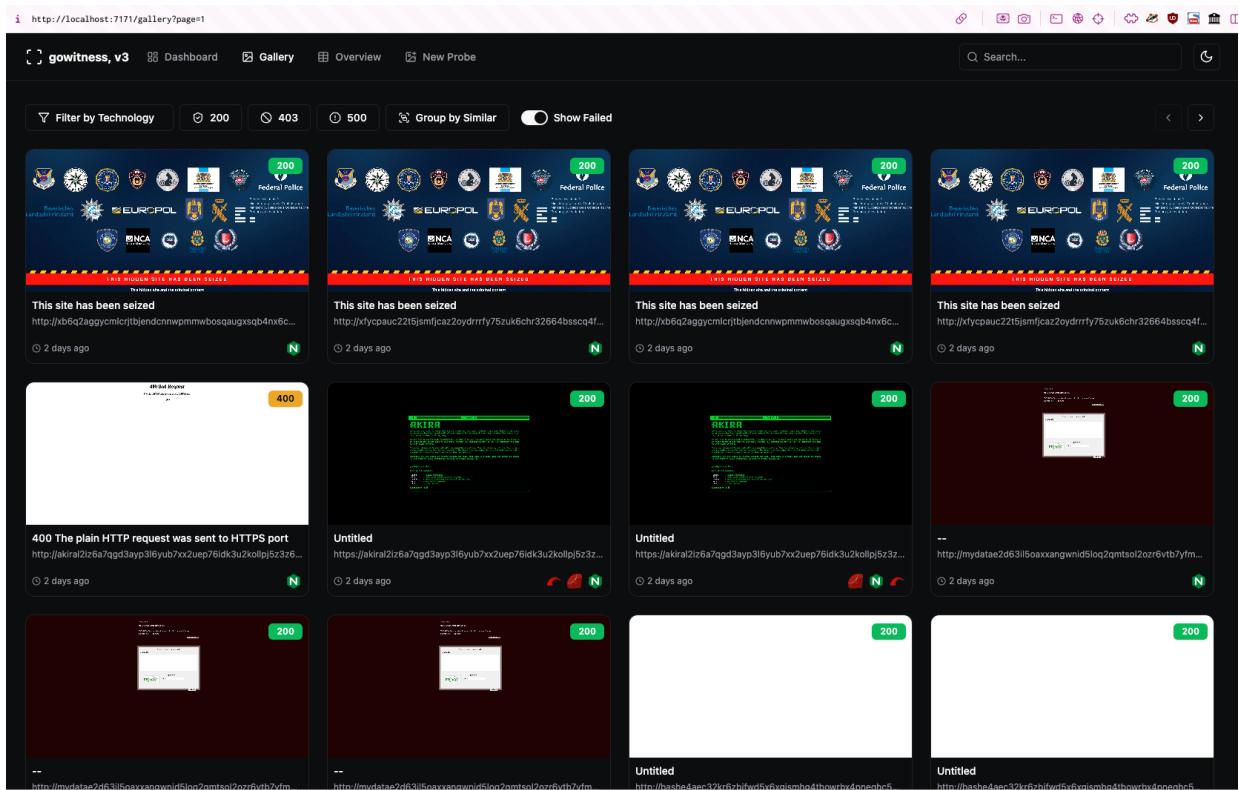


Figura 3.1.1: Servidor local GoWitness

Además, parte de esta información permite realizar consultas inversas en plataformas de inteligencia de amenazas como Shodan o Censys, incrementando así la visibilidad de estas infraestructuras en la red abierta (Clearnet) y facilitando tareas de seguimiento o desanonimización.

Para el análisis automatizado, se han empleado diversas herramientas complementarias. Principalmente se ha utilizado **GoWitness** para la recolección visual y técnica, así como **nuclei** para el escaneo de vulnerabilidades. Posteriormente, se han desarrollado y aplicado scripts específicos en Python y bash para extraer, procesar y correlacionar la información obtenida. A continuación, se detallan los análisis individuales realizados.

3.2.1 ENCABEZADOS HTTP

Los encabezados HTTP proporcionan una visión profunda sobre el entorno de ejecución y las tecnologías web utilizadas en un servidor. A través de campos como **Server**, **X-Powered-By** o **Set-Cookie**, se puede identificar no solo el software que opera en el backend, sino también configuraciones específicas, medidas de seguridad implementadas (como cabeceras CSP o HSTS) e incluso deducir proxies inversos o soluciones de ofuscación de identidad.

Para explotar esta fuente de información, se ha usado la API local de GoWitness, extrayendo todos los registros en bruto en formato JSON. El siguiente script ha permitido la descarga masiva de los datos:

```
for id in $(curl -s http://localhost:7171/api/results/list | jq -r '.[] .id'); do
    curl -s http://localhost:7171/api/results/detail/$id | jq done >
    ↵ resultados-api.json
```

Para facilitar el procesamiento posterior, los resultados se han consolidado en un único array JSON:

```
jq -s '..' resultados-api.json > resultados-array.json
```

Sobre este fichero se ha ejecutado el script `fingerprinting.py`, desarrollado en el marco del presente trabajo. Este programa analiza los encabezados HTTP de cada sitio, generando una huella digital excluyendo campos volátiles (como fechas o identificadores dinámicos) para evitar sesgos. Además de identificar configuraciones únicas, también permite agrupar dominios por patrones comunes.

El análisis incluye:

- Detección y agrupación de sitios con encabezados idénticos.
- Estadísticas de frecuencia para valores como `Server` o `X-Powered-By`.
- Exportación de resultados en CSV para su posterior visualización.

Este enfoque permite no solo caracterizar a nivel individual, sino también establecer relaciones que podrían pasar desapercibidas en un análisis superficial.

3.2.2 ETAGS

Los **ETags** (*Entity Tags*) son un mecanismo del protocolo HTTP que permite identificar de forma única una versión específica de un recurso en la web. Se utilizan principalmente para optimizar el cacheo de contenido por parte del navegador.

1. El servidor genera un identificador único (por ejemplo, un hash del contenido) y lo envía en la cabecera `ETag: ETag: W/"b18069a498868e825b3f5c4aec445ac"`

2. El navegador almacena este valor y lo reenvía en futuras solicitudes mediante la cabecera **If-None-Match**.
3. Si el recurso no ha cambiado, el servidor responde con **304 Not Modified**.

Aunque fueron diseñados para mejorar el rendimiento, los ETags pueden ser **abusados como mecanismo de tracking**, incluso cuando el usuario borra cookies o navega en modo privado. Esto se debe a que el identificador puede persistir en la caché del navegador. Los ETags representan una amenaza para el anonimato en redes como Tor en las siguientes circunstancias:

- Si un servidor genera un ETag único para un usuario, puede identificarlo en visitas posteriores, incluso sin cookies.
- Aunque el usuario cambie de IP (por ejemplo, usando Tor), el ETag puede actuar como identificador persistente.
- Esto permite correlacionar múltiples accesos de un mismo usuario, comprometiendo su anonimato.

Si un ETag observado en un sitio `.onion` se encuentra también en plataformas como **Fofa** asociado a servidores en la **clearnet**, esto puede ser un hallazgo muy significativo. En esos casos, es posible que el mismo recurso —como un archivo CSS o JavaScript— esté siendo servido tanto en la red Tor como en un entorno accesible públicamente. Esta coincidencia puede indicar que ambos sitios comparten código fuente, infraestructura técnica o incluso que están gestionados por la misma entidad. En consecuencia, dicha correlación podría facilitar la **desanonimización** del operador del servicio oculto.

Los ETags, si bien fueron concebidos como herramienta de optimización web, pueden convertirse en un **vector de fingerprinting y desanonimización**. Su reutilización entre servicios `.onion` y dominios públicos en la clearnet permite establecer correlaciones técnicas que pueden romper el anonimato de un servicio oculto.

Response Headers

Header	Value
cache-control	max-age=0, private, must-revalidate
x-permitted-cross-domain-policies	none
content-encoding	gzip
referrer-policy	strict-origin-when-cross-origin
server	nginx
x-request-id	c9c7ebbd-79aa-4117-9d56-71a7bc4f9f26
link	</assets/application-b6d25f6147ebeb3827c09f3cff8e8258621e7ffa510276f5b7f8ae8ed2db01ba.css>; rel=preload; as=style; nopush, </assets/application-9090c5cce3efccfdc89b40560193bb1d805853340dc2f63d55c1927f9a7ada7c.js>; rel=preload; as=script; nopush
strict-transport-security	max-age=631138519
x-frame-options	SAMEORIGIN
x-runtime	0.006502
x-xss-protection	0
content-type	text/html; charset=utf-8
date	Sat, 07 Jun 2025 10:30:23 GMT
etag	W/"b18069a498868e825b3f5c4aec445ac"
x-content-type-options	nosniff

Figura 3.2.1: Parámetro ETag en la cabecera de respuesta HTTP

3.2.3 FAVICONS

Aunque el favicon pueda parecer un detalle menor, su análisis puede revelar coincidencias visuales y técnicas entre sitios distintos. Debido a que los favicons no suelen cambiar con frecuencia y son a menudo reutilizados por operadores de ransomware (o sus afiliados), su huella digital puede ser utilizada para identificar infraestructura compartida.

El script `favicon.py` automatiza esta tarea. Opera sobre el fichero `resultados-array.json` y filtra aquellos sitios con código de respuesta 200. Utiliza una conexión HTTP a través de Tor para descargar el archivo `favicon.ico` y calcula dos hashes:

- Hash MD5 en hexadecimal.
- Hash Murmur3 aplicado a los bytes codificados en Base64 del favicon.

Con estos valores, se generan automáticamente las siguientes consultas:

- `http/favicon.hash:<murmur3>` para Shodan.
- `services.http/favicon_hash:<murmur3>` para Censys.
- `icon_hash=<murmur3>` para FOFA.

Esto permite realizar búsquedas inversas y detectar otros servicios que reutilicen el mismo favicon, ampliando el alcance del análisis y conectando potencialmente servicios de Clearnet y Darknet.

3.2.4 CERTIFICADOS TLS

Algunos sitios accesibles a través de la red TOR implementan conexiones HTTPS, lo que implica el uso de certificados TLS. Aunque los certificados auto-firmados son comunes en la Dark Web, su análisis puede ofrecer pistas valiosas: desde reutilización de campos como el emisor o sujeto, hasta coincidencias exactas en huellas digitales que indiquen infraestructura compartida.

El script `tls_extract.py` se encarga de esta labor. Para cada URL con esquema `https`, establece una conexión TLS a través de un socket SOCKS5 apuntando al proxy Tor local. Si la conexión tiene éxito, extrae y analiza el certificado con la librería `cryptography`. Los campos que se examinan incluyen:

- Huellas SHA-1 y SHA-256.
- Número de serie y versión del certificado.
- Sujeto y emisor.
- Periodo de validez.
- Algoritmos y tamaño de clave pública.
- Extensiones como `Subject Alternative Names`, `Key Usage` y `Extended Key Usage`.
- Indicador de autoridad de certificación (CA).

3.2.5 ESTADO DEL SERVIDOR

Esta técnica aplica principalmente a servidores que utilizan Apache como servidor web. Apache incluye, por defecto, un módulo llamado `mod_status`, el cual proporciona una página de estado accesible mediante la ruta `/server-status`. Esta página permite visualizar información en tiempo real sobre el estado del servidor, incluyendo estadísticas de uso, rendimiento y configuración.

Si esta página no ha sido deshabilitada explícitamente por el administrador del sistema, puede accederse directamente desde un navegador web, por ejemplo: `http://example.com/server-status`. Muchos sistemas la tienen activada por defecto, especialmente en entornos de pruebas o configuraciones descuidadas.

La información expuesta en `/server-status` puede ser muy valiosa desde una perspectiva de auditoría o recopilación de información (*information gathering*), ya que permite obtener datos como:

- La IP pública o nombre de dominio del servidor.
- El nombre o alias de los *virtual hosts* configurados.
- El número de peticiones que se están procesando en tiempo real.
- Los procesos activos del servidor y las conexiones actuales.
- El tiempo de actividad del servidor (*uptime*).
- La carga del servidor y el tráfico que ha manejado.
- Información sobre las rutas accedidas y los agentes de usuario (*user agents*).

En un contexto ofensivo, esta información podría ser utilizada para identificar debilidades en la configuración del servidor o para planificar ataques más dirigidos.

4

Hallazgos

En esta sección se presentan y analizan en profundidad los principales hallazgos obtenidos a lo largo de la investigación.

4.1 RESULTADOS DEL ANÁLISIS DE ENCABEZADOS HTTP

Una vez procesado el fichero `resultados-array.json` mediante el script `fingerprinting.py`, se generaron estructuras que permiten representar de forma normalizada los encabezados HTTP observados. Estos datos constituyen la base del análisis de huellas digitales (*fingerprints*) aplicadas a los servicios web detectados. El objetivo principal es identificar configuraciones comunes, detectar agrupaciones de servicios y analizar la diversidad tecnológica a partir de la información expuesta en las cabeceras de respuesta. A modo ilustrativo, la siguiente entrada —la primera del fichero `identical_fingerprints.csv`— se presenta en formato JSON para mostrar la estructura típica de los datos analizados:

```
1 {  
2     "url": "http://xb6q2aggycmlcrjtbjendcnnwpmmwbosqaugxsqb4nx6cmod3emy7sad.onion/",  
3     "key_fp": "20b756b024a4",  
4     "kv_fp": "653046009ef3",  
5     "server": "nginx",  
6     "num_headers": 13,  
7     "headers_set": "{ 'connection', 'referrer-policy', 'transfer-encoding',  
8         ↪ 'permissions-policy', 'content-type', 'x-content-type-options', 'x-xss-protection' }",  
9     "kv_pairs": "(( 'connection', 'keep-alive'), ( 'content-encoding', 'gzip'), ( 'content-type',  
    ↪ 'text/html'), ( 'permissions-policy', 'geolocation=(), microphone=())")  
}
```

El conjunto de datos está compuesto por 101 servicios web distintos, para los cuales se generaron dos tipos de huellas digitales: `key_fp`, basada únicamente en el conjunto de claves de cabecera, y `kv_fp`, que considera tanto claves como valores. Esta dualidad permite analizar el grado de similitud entre servicios a diferentes niveles de detalle.

Uno de los hallazgos más relevantes es la reducida diversidad de huellas: solo 16 valores únicos para `key_fp` y 22 para `kv_fp`. Esta baja entropía sugiere una fuerte estandarización en la configuración de los servidores, probablemente derivada del **uso de plantillas, herramientas de despliegue automatizado o configuraciones por defecto no personalizadas**.

Respecto al número de cabeceras por respuesta, el campo `num_headers` revela una media de 7.3 cabeceras, con un rango entre 4 y 13. Esta variabilidad indica distintos niveles de sofisticación en la configuración, desde servicios mínimos con respuestas básicas hasta entornos más elaborados con cabeceras orientadas a la seguridad o la compatibilidad.

El análisis del campo `headers_set` permite identificar conjuntos de cabeceras recurrentes. Por ejemplo, se observan múltiples URLs que comparten el mismo conjunto exacto de cabeceras, lo que refuerza la hipótesis de **replicación de configuraciones**. La siguiente tabla recoge ejemplos representativos:

URL	headers_set
http://xb6q2aggycmlcrjtbjendcnnwpmmwbosqaugxsb4nx6cmmod3emy7sad.onion/	{'x-frame-options', 'x-content-type-options', 'permissions-policy', 'referrer-policy', 'server', 'x-xss-protection', 'connection', 'transfer-encoding', 'content-encoding', 'content-type'}
http://xfycopauc22t5jsmfjcaz2oydrrrfy75zuk6chr32664bsscq4fgyaaqd.onion/	{'x-frame-options', 'x-content-type-options', 'permissions-policy', 'referrer-policy', 'server', 'x-xss-protection', 'connection', 'transfer-encoding', 'content-encoding', 'content-type'}
http://basheqtvzqwz4vp6ks5lm2ocq7i6tozqgf6vjcasj4ezmsy4bkpshyd.onion/	{'server', 'connection', 'transfer-encoding', 'content-encoding', 'content-type'}
http://basherp53eniermxovo3bkduw5qqq5bkqcml3qictfmamgvmzovykyqd.onion/	{'server', 'connection', 'transfer-encoding', 'content-encoding', 'content-type'}

Cuadro 4.1.1: URLs y conjuntos de cabeceras HTTP idénticos.

El campo `server`, encargado de indicar el software del servidor web, revela información relevante sobre las tecnologías utilizadas. En el 20.8 % de los casos (21 respuestas), este valor está ausente, lo que puede inter-

pretarse como una medida deliberada de ocultación (*server hardening*) o como resultado de configuraciones mínimas por defecto.

Entre los valores declarados, el servidor más común con diferencia es `nginx`, presente en 74 casos. A esto se suman diversas versiones explícitas de `nginx`, como `nginx/1.18.0 (ubuntu)`, `nginx/1.27.5`, `nginx/1.18.0` y otras, lo que refuerza su posición dominante en el ecosistema observado. Le sigue `apache`, también con 11 ocurrencias en su forma genérica, además de varias versiones detalladas como `apache/2.4.52 (ubuntu)`. La presencia de variantes de `nginx` y `apache` confirma su popularidad como servidores web ampliamente adoptados, tanto en entornos de producción como en despliegues automatizados.

Asimismo, aparecen otros servidores como `openresty`, una bifurcación avanzada de `nginx`, aunque con menor frecuencia. Esta información permite inferir tanto la popularidad relativa de ciertas tecnologías como la posible existencia de automatización o replicación de configuraciones en los entornos analizados. En contextos donde múltiples servicios comparten el mismo `fingerprint` y servidor, **se fortalece la hipótesis de entornos gestionados de forma homogénea o centralizada.**

Servidor (verbatim)	Cantidad
<code>nginx</code>	74
<code>(vacío)</code>	21
<code>nginx/1.18.0 (ubuntu)</code>	11
<code>apache</code>	11
<code>nginx/1.27.5</code>	11
<code>nginx/1.18.0</code>	4
<code>nginx/1.24.0</code>	2
<code>(ubuntu)</code>	
<code>nginx/1.22.1</code>	2
<code>nginx/1.26.1</code>	2
<code>apache/2.4.52</code>	2
<code>(ubuntu)</code>	
<code>openresty</code>	2
<code>nginx/1.27.0</code>	1
<code>microsoft-iis/10.0</code>	1
<code>nginx/1.25.5</code>	1
<code>nginx/1.27.1</code>	1
<code>kestrel</code>	1
<code>apache/2.4.63</code>	1
<code>(win64) php/8.4.5</code>	

Servidor	Cantidad
nginx/1.26.3	1
apache/2.4.58 (ubuntu)	1

Cuadro 4.1.2: Servidores HTTP más frecuentes detectados en los servicios analizados.

En el contexto de este trabajo, estos resultados tienen múltiples implicaciones. En primer lugar, la baja diversidad de **fingerprints** en el conjunto sugiere que es posible agrupar grandes cantidades de servidores en pocas clases equivalentes, lo cual simplifica significativamente tareas como la detección de infraestructuras homogéneas o la atribución de sistemas. En segundo lugar, la identificación de cabeceras comunes asociadas a buenas prácticas podría servir como una métrica indirecta de madurez en la gestión de la configuración de los servidores. Finalmente, la existencia de **fingerprints** idénticos en URLs distintas podría apuntar a la presencia de sistemas replicados, clonados o gestionados por un mismo proveedor o panel de administración.

Estos resultados refuerzan la hipótesis de que muchas infraestructuras .onion relacionadas con ransomware son desplegadas mediante configuraciones estandarizadas, posiblemente gestionadas desde plataformas compartidas. La escasa diversidad en las huellas digitales sugiere un ecosistema técnico repetitivo, lo cual facilita tareas de atribución y rastreo mediante técnicas pasivas de fingerprinting.

4.2 RESULTADOS DEL ANÁLISIS DE ETAGS

Durante el análisis se identificaron un total de **45 ETags únicos**, extraídos de diferentes servicios .onion. Con el objetivo de explorar posibles vínculos con infraestructura en la **clearnet**, estos valores fueron buscados en la plataforma de inteligencia pasiva **FOFA**. Esta herramienta recopila cabeceras HTTP históricas mediante motores de escaneo, lo que la convierte en un recurso valioso para detectar coincidencias entre servicios aparentemente independientes.

Aunque la mayoría de los ETags no devolvieron resultados —posiblemente debido a su unicidad, naturaleza efímera o generación aleatoria—, **cuatro de ellos sí coincidieron con servidores accesibles públicamente**. Estas coincidencias resultan significativas, ya que permiten inferir relaciones entre servicios ocultos y dominios de la clearnet. La tabla 4.2.1 resume los casos identificados:

URL	ETag	FOFA Query
http://incbacg6bfwtrlwdbqc55gsf1763s3twdtwhp27dzuik6s6rwdcityd.onion	W/"3a-5Ed5M0VtOoLDha0QsMEz0eQ4seo"	header="etag: W/"3a-5Ed5M0VtOoLDha0QsMEz0eQ4seo"
http://incblog6qu4y4mm4zvw5nrmue6qbwtgjsxpw6b7ixzssu36tsajlload.onion	"6622b950-1c0"	header="etag: "6622b950-1c0"
http://lynxchatly4zludmhmi75jrwhycnoqvkxb4prohxmyzf4euf5gjxroad.onion	"6739ab3c-322"	header="etag: "6739ab3c-322"
http://lynxblogxstgzsarfyk2pvhdv45igghb4zmthnzmsipzeoduruz3xwqd.onion	"66ad19f5-323"	header="etag: "66ad19f5-323"

Cuadro 4.2.1: Resumen de sitios .onion: URLs, valores ETag y consultas FOFA asociadas.

A continuación, se desarrollan los resultados específicos de cada ETag.

4.2.1 INC. RANSOM - PORTAL PRINCIPAL

El primer ETag se asoció con el portal principal del grupo **Inc. Ransom**, activo desde mediados de 2023. En FOFA se identificaron **15 hosts en la clearnet** que comparten este valor de cabecera. La mayoría de estos servidores se encuentran en Rusia y Finlandia, con una fuerte concentración en el ASN 198953 (*Proton66 OOO*), proveedor vinculado a otras operaciones cibercriminales.

Asimismo, se localizaron dos dominios bajo infraestructura de **Google Cloud** (`incback.su` y `lynxstorage1.net`), lo que sugiere escenarios como la reutilización de infraestructura entre Inc. Ransom y LYNX, el uso de servidores espejo en la clearnet, o un posible **error de OPSEC** debido a la repetición de configuraciones y código.

Host (IP:Puerto)	ASN	Ciudad, País	Organización
http://45.140.17.28:1000	198953	Saint Petersburg, RU	Proton66 OOO
http://45.135.232.84:1000	198953	Saint Petersburg, RU	Proton66 OOO
http://178.20.41.208:1000	48282	Moscow, RU	Hosting technology LTD
http://45.134.26.39:1000	198953	Saint Petersburg, RU	Proton66 OOO
http://135.181.179.122:1000	24940	Helsinki, FI	Hetzner Online GmbH
http://lynxstorage2.net (45.152.84.73:80)	56971	San Francisco, US	Cgi Global Limited
http://193.3.19.33:1000	50340	Moscow, RU	OOO Network of data-centers Selectel
http://45.130.145.39:1000	49392	Moscow, RU	LLC Baxet
http://incback.su (34.175.159.164:80)	396982	Madrid, ES	GOOGLE-CLOUD-PLATFORM

http://lynxstorage1.net (34.175.159.164:80)	396982	Madrid, ES	GOOGLE-CLOUD-PLATFORM
http://213.109.202.222:1000	208312	Moscow, RU	Red Byte LLC
http://85.209.11.48:1000	49392	Dubai, AE	LLC Baxet
http://31.41.244.135:1000	57678	Saint Petersburg, RU	Cat Technologies Co. Limited
http://62.122.184.243:1000	57523	Moscow, RU	Chang Way Technologies Co. Limited
http://194.26.135.218:1000	57523	Moscow, RU	Chang Way Technologies Co. Limited

Cuadro 4.2.2: Hosts identificados para el ETag W/"3a-5Ed5M0VtOoLDha0QsMEz0eQ4seo"

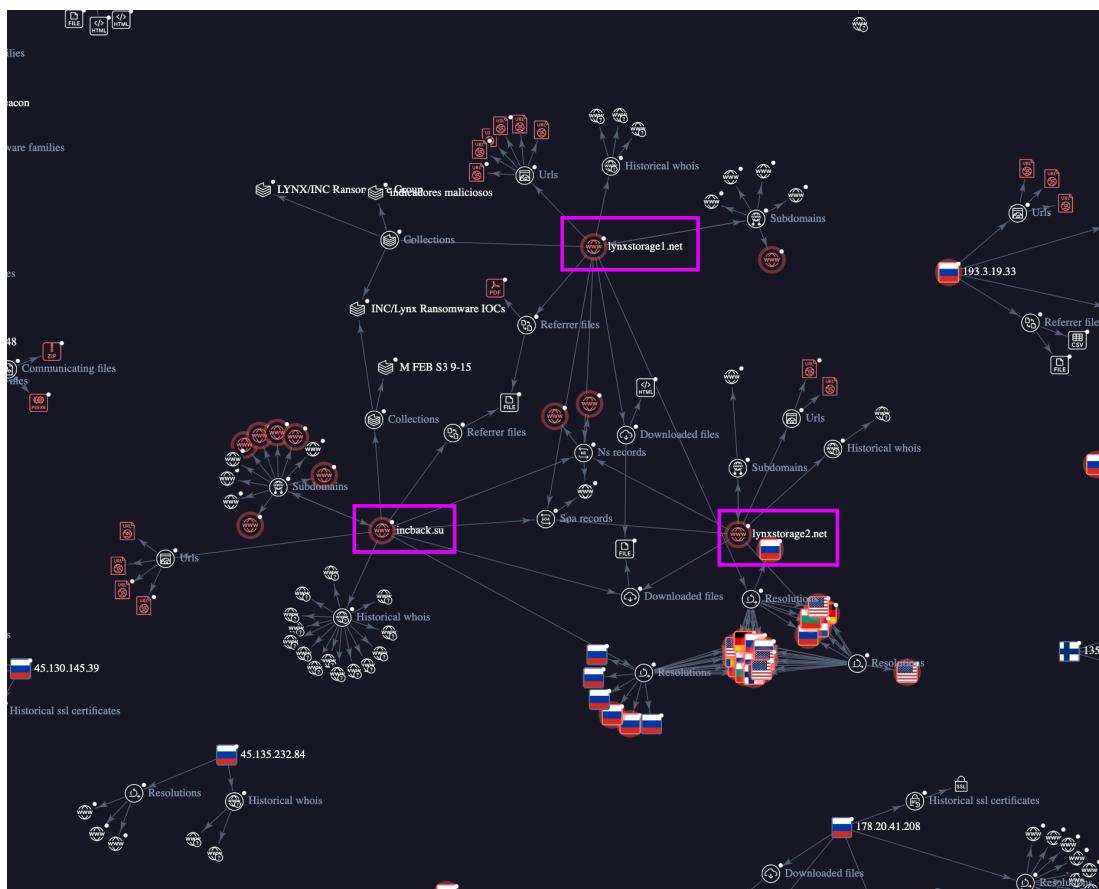


Figura 4.2.1: Infraestructura atribuida al grupo LYNX/INC Ransom construida a partir de relaciones ETag

A partir de los datos obtenidos se elaboró un grafo en VirusTotal (figura 4.2.1) que refleja las relaciones entre direcciones IP asociadas al ETag W/"3a-5Ed5M0Vt0oLDha0QsMEz0eQ4seo". Destacan tres dominios centrales —lynxstorage1.net, lynxstorage2.net e incback.su—, vinculados a múltiples actividades maliciosas como descarga de artefactos, comunicación con otros nodos sospechosos y posible rol como infraestructura C2.

4.2.2 INC. RANSOM - BLOG SECUNDARIO

El segundo ETag, también vinculado a Inc. Ransom, fue identificado en el dominio <http://incblog.su>, alojado en Google Cloud (Madrid). Esta coincidencia puede interpretarse como una réplica parcial del blog .onion, un despliegue de infraestructura redundante entre clearnet y Tor, o como estrategia de evasión mediante servicios cloud.

Host (IP:Puerto)	ASN	Ciudad, País	Organización
http://incblog.su (34.175.159.164:80)	396982	Madrid, ES	GOOGLE-CLOUD-PLATFORM

Cuadro 4.2.3: Host identificado del ETag "6622b950-1c0"

4.2.3 LYNX - CHAT

El tercer ETag analizado corresponde al grupo de ransomware **Lynx**. Dicho valor, observado inicialmente en su servicio .onion, fue también localizado en el dominio lynxchat.net, hospedado en la misma infraestructura de Google Cloud que otros servicios previamente atribuidos a INC Ransom. Esta coincidencia resulta particularmente relevante, ya que sugiere una posible **relación operativa** entre ambos grupos o, en su defecto, un uso compartido de proveedores de infraestructura.

Host (IP:Puerto)	ASN	Ciudad, País	Organización
http://lynxchat.net (34.175.159.164:80)	396982	Madrid, ES	GOOGLE-CLOUD-PLATFORM

Cuadro 4.2.4: Host identificado del ETag "6739ab3c-322"

Una búsqueda posterior de lynxchat.net en **VirusTotal** reveló múltiples direcciones IP asociadas, siendo la más reciente 147.45.198.222. El análisis de esta IP permitió identificar dominios adicionales vinculados al ecosistema de Lynx, entre los que destaca lynxr.blog, portal que se encuentra operativo y que presenta la misma información que los sitios de fuga de datos listados en la plataforma [ransomware.live](#) a fecha del 22 de agosto. Entre los portales ocultos relacionados destacan:

- lynxblogxutufossaeeawlij3j3uikaloll5ko6grzhkwcdclrjngrfoid.onion
- lynxblogoxllth4b46cfwlop5pfj4s7dyv37yuy7qn2ftan6gd72hsad.onion
- lynxblogco7r37jt7p5wrmfxzqze7ghxw6rihzkqc455qluacwotciyd.onion
- lynxblogijy4jfoblgix2klxmkbgee4leoeuge7qt4fpfkj4zbi2sjyd.onion
- lynxblogtwatfsrwj3oatpejwxk5bngqcd5f7s26iskagfu7ouaomjad.onion

The screenshot shows the Lynx Blog website interface. On the left, there's a sidebar with a logo and three menu items: 'News', 'Leaks' (which is highlighted in blue), and 'Report'. The main content area is titled 'Leaks' and displays six document cards in a grid. Each card has a thumbnail, the document title, the date of publication, a brief description, category, views, and a 'Go to the publication' button.

Thumbnail	Title	Date of publication	Description	Category	Views	Action
	abfe	20/08/2025	American Bitrite Far East Inc (ABFE) is located in the hub of Asia, Singapore.	Encrypted	689	Go to the publication
	Bizcom Electronics	20/08/2025	izcom Electronics, Inc. is a service provider based in Milpitas, California, spe...	Encrypted	590	Go to the publication
	HK Hardware & Engineering	20/08/2025	HK Hardware Engineering Pte Ltd aims to be a one-stop industrial supplier for t...	Encrypted	611	Go to the publication
	Frontline Bioenergy	04/08/2025	Since 2003, Frontline BioEnergy has been designing systems and proprietary equip...	Proof, Encrypted, AD Dump	2902	Go to the publication
	True World Group LLC	19/08/2025	True World Group is one of the nation's leading, diversified seafood-products co...	Proof	913	Go to the publication
	Drive & Shine	04/08/2025	Drive & Shine is a premier car care service that offers express car washes, inte...	Encrypted, AD Dump, Proof	2196	Go to the publication

Figura 4.2.2: Portal web de Lynx Blog

Mediante la herramienta `dig`, se comprobó que el dominio `lynxr.blog` resuelve a la dirección IP `193.46.217.98` (figura 4.2.3).

```

❯ dig lynxr.blog

; <>> DiG 9.10.6 <>> lynxr.blog
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53794
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;lynxr.blog.           IN      A

;; ANSWER SECTION:
lynxr.blog.      595     IN      A      193.46.217.98

;; Query time: 665 msec
;; SERVER: 10.2.0.1#53(10.2.0.1)
;; WHEN: Sat Aug 23 13:51:53 CEST 2025
;; MSG SIZE rcvd: 55

```

Figura 4.2.3: Uso de herramienta dig en el dominio lynxr.blog

El análisis de esta IP en **FOFA** arrojó coincidencias con otros servicios asociados a Lynx, así como con dominios de carácter fraudulento:

Host (IP:Puerto)	ASN	Ciudad	Organización
http://incrans.com	400992	Madrid, ES	ZHOUYISAT-COMMUNICATIONS
http://193.46.217.98	400992	Madrid, ES	ZHOUYISAT-COMMUNICATIONS
http://193.46.217.98:8081		Madrid, ES	
http://www.playstores-spain.com		Madrid, ES	
http://playstores-spain.com		Madrid, ES	
https://playstores-spain.com		Madrid, ES	

Cuadro 4.2.5: Hosts identificados para la IP 193.46.217.98

A su vez, VirusTotal muestra resoluciones recientes que refuerzan la actividad constante de esta infraestructura. La tabla 4.2.6 resume algunos de los dominios detectados en agosto de 2025:

Fecha	Dominio
2025-08-23	hoaxilla.lynxcdn.com
2025-08-23	inccdn1.online
2025-08-23	inccdn2.online
2025-08-23	incrans.com
2025-08-23	lynxcdn.com
2025-08-23	lynxr.blog
2025-08-23	lynxr.help
2025-08-23	second.lynxcdn.com
2025-08-23	www.incrans.com
2025-08-23	www.lynxr.blog
2025-08-23	www.lynxr.help
2025-08-23	first.lynxcdn.com
2025-08-21	blog.sinobi.us.org
2025-08-21	cdn.sinobi.us.org

Entre estos resultados destaca incrans.com, que a fecha del 24 de agosto se encontraba plenamente operativo (figura 4.2.4).

The screenshot shows a dark-themed website for 'INC Ransom'. On the left sidebar, there are links for 'News', 'Disclosures', and 'Report'. The main content area has a header 'Blog / Disclosures / 68a8bc0ec7b82dfe0b7ece16'. Below this, a list of victims is displayed with their logos, names, and country flags. To the right, a detailed view of the 'Quadrangle Imaging Center' entry is shown.

Victim	Country	Replies
Quadrangle Imaging Center	Puerto Rico	572
Universal Group For Engineering and Consulting	Palestine	1053
EUROFILM MANTZAPHS A.E	Greece	1220
Vitec	USA	9097
NETSTAR	South Africa	1277
Töller	Belgium	2664
MJETS	France	2492
www.reliablecontrols.com	USA	7931
mycpaconnection.com	USA	1990

Quadrangle Imaging Center 22.08.2025 18:50
Revenue: 5M\$
Patients' personal data and related medical information
Information for organizing appointments and feedback
Accounting and finance
mri@qicpr.com
Tel. 787-746-1688
ct@qicpr.com
xr@qicpr.com

Part1 22.08.2025 18:51

Figura 4.2.4: Portal web blog incrans.com

Finalmente, la dirección IP 193.46.217.98 fue atribuida al ASN AS400992 (*ZhouyiSat Communications*), con ubicación en Fremont, California, y gestionada por **Huize Holdings LLC**.

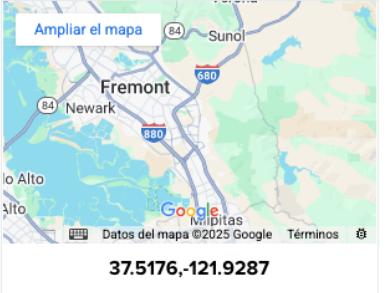
193.46.217.98

📍 Fremont, California, US 🇺🇸 SSH Webserver

Summary	
Geolocation	
Privacy	
ASN	
Company	
Abuse	

Summary	
ASN	AS400992 - ZhouyiSat Communications
Hostname	No Hostname
Range	193.46.217.0/24
Company	Huize Holdings LLC
Hosted domains	0
Privacy	✗ False
Anycast	✗ False
ASN type	ISP
Abuse contact	support@62yun.com

IP Geolocation	
City	Fremont
State	California
Country	🇺🇸 United States
Postal	94539
Local time	05:23 AM, Saturday, August 23, 2025
Timezone	America/Los_Angeles
Coordinates	37.5176,-121.9287



Ampliar el mapa Datos del mapa ©2025 Google Términos

37.5176, -121.9287

Figura 4.2.5: Información relacionada con la IP 193.46.217.98

4.2.4 LYNX - BLOG

El cuarto ETag fue detectado en `lynxblog.media` y `lynxblog.net`, ubicados en Moscú y Donetsk respectivamente. Ambos comparten el mismo valor, lo que sugiere reutilización de código o configuración. La diversidad geográfica puede estar orientada a resiliencia, aunque refleja escasa preocupación por diversificar huellas digitales.

Host (IP:Puerto)	ASN	Ciudad, País	Organización
<code>http://lynxblog.media</code> (46.173.214.16:80)	47196	Moscow, RU	Garant-Park-Internet LLC
<code>http://lynxblog.net</code> (185.218.0.29:80)	56769	Donetsk, RU	Donclip LLC

Cuadro 4.2.7: Hosts identificados del ETag "66ad19f5-323"

4.2.5 ANÁLISIS TÉCNICO: DE INC A LYNX RANSOMWARE

En mayo de 2024 el grupo **INC** anunció en foros clandestinos la venta de su proyecto, marcando el cierre de su actividad pública [5, 6]. Apenas dos meses después, en julio de 2024, emergió **LYNX**, considerado sucesor directo. Diversos análisis confirman que LYNX reutiliza código de INC, mantiene una estructura organizativa similar y presenta evidencias de transición o reventa del proyecto original [6, 7].

Característica	INC Ransomware	LYNX Ransomware
Año de aparición	2023	Julio 2024
Modelo operativo	RaaS	RaaS
Sistemas operativos afectados	Windows, Linux, ESXi	Windows
Métodos de extorsión	Cifrado de archivos	Doble extorsión (cifrado + exfiltración)
Código compartido	Original	Reutiliza funciones del código de INC
Extensión usada	.inc, .lnx	.lynx
Parámetros CLI	Básicos	Soporte avanzado: --mode, --verbose, --norename
Técnicas de evasión	Terminación de procesos (SQL, Veeam, Exchange)	Igual, pero optimizado

Leak site	Limitado y no siempre activo	Leak site activo y público, con víctimas en sectores legal, energía y retail
-----------	------------------------------	--

LYNX representa así una evolución técnica y operativa: adopta un modelo de **doble extorsión**, incrementa su visibilidad pública y mejora funcionalidades, aunque conserva la arquitectura y táctica heredada. Similitudes en código, parámetros CLI, victimología y despliegue infraestructural sustentan su identificación como sucesor directo.

Además, el análisis de ETags revela su valor como **identificadores persistentes**. En servicios ocultos, su correlación puede tener varias implicaciones:

- **Fingerprinting de infraestructura:** coincidencias permiten vincular servidores que comparten recursos o código.
- **Errores de OPSEC:** reutilización de ETags entre clearnet y Tor facilita procesos de *deanonymization*.
- **Relaciones entre grupos:** como se observa en la conexión INC–LYNX, sugieren nexos logísticos u operativos.

Finalmente, se constató que numerosos servidores no aplican rotación de ETags ni aumentan su entropía, lo que constituye una debilidad relevante en términos de privacidad y seguridad operativa.

4.3 RESULTADOS DEL ANÁLISIS DE FAVICONS

El archivo generado a partir del procesamiento de los *favicons*, mediante el script `favicon.py` aplicado sobre el fichero `resultados-array.json`, contiene distintas entradas, cada una con un conjunto de parámetros relevantes para el análisis. A continuación, se muestra como ejemplo la primera entrada del fichero `favicons_hashes.csv`, representada en formato JSON para ilustrar la estructura de los datos obtenidos.

```

1  {
2      "url": "http://34o4m3f26ucyeddzpf53bksy76wd737nf2fytslovwd3viac3by5chad.onion/",
3      "md5_hex": "51813cf5cc9721494b3e9e5c57234964",
4      "murmur3_int": -491684375,
```

```

5   "shodan_query": "http.favicon.hash:-491684375",
6   "censys_query": "services.http.favicon_hash:-491684375",
7   "fofa_query": "icon_hash=-491684375",
8   "fofa_hits": 0
9 }
```

El análisis del conjunto de datos de favicons, compuesto por 54 entradas, se centra en el uso de hashes del ícono del sitio web como una técnica auxiliar para la identificación de servicios web, especialmente en entornos opacos como la red Tor. Cada favicon ha sido procesado para obtener su representación hash mediante el algoritmo MurmurHash3, cuya versión entera se usa comúnmente en motores de búsqueda de inteligencia como Shodan, Censys y FOFA.

Una de las observaciones más destacadas es la presencia de **33 valores únicos** de hash Murmur3 entre las 54 URLs, lo que indica una **moderada redundancia de favicons** entre los servicios analizados. Este nivel de coincidencia sugiere que algunos servicios comparten el mismo software, plantilla o panel de administración, ya que muchos sistemas reutilizan favicons por defecto sin personalización. Por ejemplo, dos de los hashes más frecuentes (1986960332 y 1150569800) están presentes en 6 servicios distintos cada uno, lo cual puede apuntar a la utilización de frameworks específicos o entornos clonados para el despliegue de los sitios .onion.

Este tipo de *fingerprinting* basado en favicon es particularmente útil cuando se combina con huellas de cabeceras HTTP y otras métricas observables del servidor. En el contexto de este TFM, los resultados refuerzan la hipótesis de que es posible correlacionar y agrupar múltiples servicios aparentemente independientes en función de huellas técnicas compartidas. Además, el uso de consultas generadas automáticamente para plataformas como Shodan, Censys o FOFA, permite escalar esta estrategia de detección fuera de la red Tor y contrastar con la superficie web, lo cual aporta valor añadido al proceso de atribución y mapeo de infraestructura.

En conclusión, el análisis de los favicons revela patrones de reutilización que complementan el *fingerprinting* clásico basado en cabeceras HTTP, y aporta un eje más para clasificar y estudiar servicios web en entornos anónimos. Su integración como parte de una metodología multivariable en este TFM fortalece la capacidad de detección y agrupación de servicios homogéneos.

URL	Murmur3	FOFA Query
http://34o4m3f26ucyeddzpf53bksy76wd737nf2fytslovwd3viac3by5chad.onion	-491684375	icon_hash="-491684375"
http://3pktrcgbmssvrnwe5skburdwe2h3v6ibdnn5kbjqihs6eu6s6b7ryqd.onion	1497298042	icon_hash="1497298042"
http://7ukmkdtxdkdivtjad57klqnd3kdsmq6tp45rrsxqnu76zzv3jvitlqd.onion	95028350	icon_hash="95028350"
http://aby6efzmp7jzbwgidgqc6ghxi2vwp06d7eaood5xuoxutrfosmzcjqd.onion	-2008527151	icon_hash="-2008527151"
http://basheqtvzqzw4vp6ks51m2ocq7i6tozqgf6vjcasj4ezmsy4bkpshyd.onion	-1729062639	icon_hash="-1729062639"

Cuadro 4.3.1: Resumen de favicons: URLs, hashes Murmur3 y consultas FOFA asociadas.

4.4 RESULTADOS DEL ANÁLISIS DE CERTIFICADOS TLS EN SERVICIOS .ONION

Este análisis se enfoca en un subconjunto de once servicios accesibles mediante dominios .onion dentro de la red Tor, los cuales ofrecen conectividad cifrada a través del protocolo HTTPS. La presencia de este protocolo implica el uso de certificados digitales bajo TLS, lo que brinda una oportunidad para examinar tanto la configuración criptográfica como las posibles prácticas operativas adoptadas por los administradores de dichos servicios.

La recolección de certificados se realizó durante la fase de establecimiento de las conexiones TLS, extrayendo los certificados X.509 directamente desde el canal cifrado. Para ello se empleó el script `tls-extract.py`, desarrollado específicamente en el contexto de este trabajo. Esta herramienta automatiza la conexión a servicios .onion vía HTTPS sobre la red Tor, y permite capturar información clave del certificado: tipo y tamaño de la clave pública, período de validez, campos de control como CA=TRUE, así como los identificadores del emisor (`issuer`) y el sujeto (`subject`).

El análisis de estos certificados permite no solo evaluar la solidez criptográfica de las conexiones, sino también detectar errores de configuración y prácticas potencialmente inseguras. Esta información resulta especialmente relevante en el contexto de la red Tor, donde la confianza tradicional basada en autoridades certificadoras reconocidas (CAs) es a menudo sustituida por modelos más flexibles o auto-gestionados.

Para cada servicio analizado se han considerado los siguientes aspectos:

- **Tipo y tamaño de clave pública:** Se identificaron dos algoritmos principales: RSA y curvas elípticas (EC). Las claves RSA observadas presentan tamaños de 2048 y 4096 bits, mientras que las claves EC corresponden a curvas de 384 bits. Las claves de 4096 bits, aunque más seguras en teoría, pueden implicar mayor sobrecarga computacional, mientras que las EC ofrecen una mayor eficiencia con niveles comparables de seguridad. La elección del tipo de clave puede reflejar las bibliotecas criptográficas empleadas, la orientación del operador (rendimiento vs. robustez) o simples decisiones por defecto.
- **Período de validez:** Se observaron certificados con rangos de validez atípicamente extensos, como de 1975 a 4096. Este patrón es típico de certificados autofirmados generados sin seguir políticas de expiración formales. En entornos como Tor, donde los servicios a menudo no se integran con el ecosistema tradicional de confianza, estos certificados podrían haberse creado mediante herramientas automatizadas sin restricciones temporales realistas.
- **Campo CA=TRUE:** Este campo determina si el certificado tiene autoridad para emitir otros certificados. En algunos casos, los certificados analizados marcan este campo como TRUE, lo que supone una mala práctica en certificados de servidor y puede ser explotable si el cliente no valida correctamente la cadena de confianza. Aunque en la red Tor este riesgo puede ser menor debido a la ausencia de validación tradicional, sigue siendo una señal de una posible configuración negligente.
- **Emisor del certificado:** Se detectaron tanto certificados autofirmados (donde emisor y sujeto son idénticos) como certificados emitidos por autoridades reconocidas, como HARICA. Esta diferencia permite distinguir entre servicios que buscan una legitimidad mínima frente al cliente (mediante una CA) y aquellos que simplemente garantizan una conexión cifrada funcional, sin respaldo formal.
- **Errores en la conexión TLS:** En al menos un caso, el intento de conexión TLS fue rechazado o fallido, impidiendo la recolección del certificado. Esto puede deberse a múltiples factores: bloqueo de conexiones desde ciertos nodos Tor, configuración restrictiva del servidor, errores de disponibilidad o políticas de acceso limitadas por IP.

La tabla 4.4.1 resume las características clave de los certificados extraídos, permitiendo una comparación directa entre los distintos servicios analizados.

En conjunto, el análisis de certificados TLS evidencia una falta de homogeneización en prácticas criptográficas, así como la presencia de errores operacionales (como certificados autofirmados con CA=TRUE o períodos de validez irreales). Estos hallazgos pueden aprovecharse en futuras investigaciones para crear firmas

Host .onion	Tipo de clave	Vigencia	CA=TRUE	Emisor
akiral2i...bad.onion	RSA 2048-bit	2024–2034	Sí	Internet Widgits
hunters55...yid.onion	EC 384-bit	1975–4096	No	Hunters Intl.
hunters33...qyd.onion	EC 384-bit	1975–4096	No	Hunters Intl.
hunters55...yd.onion	EC 384-bit	1975–4096	No	Hunters Intl.
metacrpt...pyd.onion (x2)	RSA 4096-bit	2024–2034	Sí	CommonName / C=XX
47glxkux...mqd.onion (x2)	RSA 2048-bit	2025–2026	No	HARICA CA, GR
weyhro27...rqd.onion	—	—	No	Error de conexión
imncreww...id.onion	RSA 2048-bit	2025–2033	Sí	IMN crew, AU
worldleak...uid.onion	EC 384-bit	1975–4096	No	—

Cuadro 4.4.1: Análisis y clasificación de certificados TLS en servicios .onion

técnicas más precisas que permitan identificar servicios clonados o desplegados desde el mismo conjunto de herramientas.

4.5 RESULTADOS DEL ANÁLISIS DE /SERVER-STATUS

Tras analizar el conjunto de datos recopilado, se identificaron **seis instancias** en las que la ruta `/server-status` no estaba deshabilitada, lo que permitía el acceso público a información sensible sobre el estado interno de los servidores. Las URLs correspondientes son las siguientes:

- <http://mydatae2d63il5oaxxangwnid5loq2qmts0l2ozr6vtb7yfm5ypzo6id.onion/server-status>
- <http://sonarmsng5vzwqezlvtu2iiwwdn3dxkh0ftikh0wpfjuzg7p3ca5eid.onion/server-status>
- <http://hellcatdcy653ma43t2ryf2ztw5yfanqsbffmapndbqvteh5itctoijyd.onion/server-status>
- <http://hellcatdnrsu4i5uctbklunpfyv2ppioh5sb31eu4dfgizinrve3gqd.onion/server-status>
- <http://hellcatdohzngkuh7zruzhi2wojrawbnzbyz1jtkw6iluv5ussfer4id.onion/server-status>
- <http://rnsmwareartse3m4hjsumjf222pnka6gad26cqxqmbjvevhbnym5p6ad.onion/server-status>

De estas, destaca la segunda dirección, asociada al grupo de ransomware **Cactus**. Al acceder a su página de estado del servidor, fue posible observar información detallada del entorno web, incluyendo direcciones IP públicas tanto en el campo *Client* como en el *VirtualHost*.

A continuación, se resumen los principales datos extraídos:

- **Versión del servidor:** Apache/2.2.15 (Unix) con PHP/5.3.3. Estas versiones presentan vulnerabilidades críticas y están fuera de soporte oficial.
- **Tiempo de actividad:** 4 horas y 1 minuto, procesando 2.748 peticiones y generando un total de 9.6 MB de tráfico.
- **Actividad detectada:** múltiples accesos desde una IP externa (115.113.134.14) al endpoint `/server-status`, lo que sugiere el uso de herramientas de monitorización automatizadas.
- **IP pública del servidor:** 5.175.142.66, identificada en el campo `VirtualHost`, lo que confirma su visibilidad en Internet.

En la Figura 4.5.1 se presenta una captura de la interfaz de `/server-status` correspondiente a esta instancia.

```

1          Apache Server Status for localhost
2  Server Version: Apache/2.2.15 (Unix) DAV/2 PHP/5.3.3
3  Server Built: Aug 13 2019 17:29:28
4
5 -----
6  Current Time: Tuesday, 14-Jan-2019 04:34:13 EST
7  Restart Time: Tuesday, 14-Jan-2019 00:33:05 EST
8  Parent Server Generation: 0
9  Server uptime: 4 hours 1 minute 7 seconds
10 Total accesses: 2748 - Total Traffic: 9.6 MB
11 CPU Usage: u.0.9 s1.06 cu0 cs0 - .0135% CPU load
12 .19 requests/sec - 695 B/second - 3658 B/request
13 1 requests currently being processed, 4 idle workers
14 .__._W...
15
16 Scoreboard Key:
17 " _" Waiting for Connection, "S" Starting up, "R" Reading Request,
18 "W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
19 "C" Closing connection, "L" Logging, "G" Gracefully finishing,
20 "I" Idle cleanup of worker, "." Open slot with no current process
21
22 Srv PID    Acc      M CPU      SS Req Conn Child Slot     Client           VHost           Request
23 0-0 -    0/0/428   . 0.30 5572 0  0.0  0.00  1.34 127.0.0.1      5.175.142.66 OPTIONS * HTTP/1.0
24                                         GET
25 1-0 5606 0/639/639 _ 0.46 4    0  0.0  2.18  2.18 [115.113.134.14] 5.175.142.66 /server-status?refresh=5
26                                         HTTP/1.1
27                                         GET
28 2-0 5607 0/603/603 _ 0.43 0    0  0.0  2.09  2.09 115.113.134.14 5.175.142.66 /server-status?refresh=5
29                                         HTTP/1.1
30 3-0 -    0/0/337   . 0.23 5573 0  0.0  0.00  1.09 127.0.0.1      5.175.142.66 OPTIONS * HTTP/1.0
31                                         GET
32 4-0 5701 0/317/317 _ 0.23 9    0  0.0  1.21  1.21 115.113.134.14 5.175.142.66 /server-status?refresh=5
33                                         HTTP/1.1
34                                         GET
35 5-0 5708 0/212/213 _ 0.15 6    0  0.0  0.85  0.85 115.113.134.14 5.175.142.66 /server-status?refresh=5
36                                         HTTP/1.1
37 6-0 5709 0/210/210 W 0.16 0    0  0.0  0.84  0.84 127.0.0.1      5.175.142.66 GET /server-status
38                                         HTTP/1.1
39 7-0 -    0/0/1     . 0.00 5574 0  0.0  0.00  0.00 127.0.0.1      5.175.142.66 OPTIONS * HTTP/1.0
40
41 -----

```

Figura 4.5.1: Captura de la página `/server-status` del grupo Cactus.

La IP identificada fue posteriormente analizada en la plataforma FOFA, revelando varios servicios expuestos sobre distintos puertos. A continuación se muestra un resumen:

Host (IP:Puerto)	ASN	Ciudad, País	Organización
5.175.142.66:22	12586	Frankfurt am Main, DE	GHOSTnet GmbH
http://5.175.142.66:3128	12586	Frankfurt am Main, DE	GHOSTnet GmbH
http://5.175.142.66:50001	12586	Frankfurt am Main, DE	GHOSTnet GmbH
http://5.175.142.66:50111	12586	Frankfurt am Main, DE	GHOSTnet GmbH
http://5.175.142.66:50002	12586	Frankfurt am Main, DE	GHOSTnet GmbH

Cuadro 4.5.1: Resumen de servicios identificados en el host 5.175.142.66.

Estos hallazgos demuestran cómo la exposición de páginas de estado del servidor, cuando no están debidamente restringidas, puede revelar información técnica crítica que facilite el reconocimiento de la infraestructura de grupos maliciosos, permitiendo incluso identificar IPs públicas, software obsoleto o herramientas de monitoreo activas.

5

Trabajo futuro

Los resultados obtenidos a lo largo de esta investigación abren múltiples líneas de trabajo que pueden contribuir a mejorar la caracterización y detección de infraestructuras maliciosas ocultas en la red Tor. Una de las direcciones más prometedoras reside en el análisis longitudinal de dominios .onion, observando la evolución de su configuración técnica a lo largo del tiempo. Si se conservaran snapshots periódicos de encabezados HTTP, certificados TLS o favicons, sería posible detectar patrones de persistencia, reutilización o rotación de elementos clave, lo que permitiría construir perfiles operacionales más precisos de los actores implicados. Esta evolución temporal facilitaría la detección de reapariciones de infraestructura, la identificación de actores reincidentes, o incluso la inferencia de ciclos operativos y ventanas de mantenimiento de los grupos criminales.

Otra vertiente complementaria consiste en la aplicación de técnicas de aprendizaje automático a los datos técnicos recopilados. Al tratarse de variables estructuradas y numéricas, tales como hashes Murmur3, fingerprints de cabeceras, tiempos de respuesta o tamaños de certificados, es viable entrenar modelos supervisados o no supervisados para detectar agrupaciones anómalas o infraestructuras relacionadas entre sí. Algoritmos como SVM, clustering jerárquico o DBSCAN podrían aplicarse a datasets etiquetados o semi-estructurados, permitiendo construir clasificadores capaces de estimar la probabilidad de que un dominio .onion pertenezca a una red maliciosa o de que comparta backend con otros portales ya identificados.

De igual importancia es el estudio específico de los paneles de administración y los frontends reutilizados en operaciones de ransomware. Muchos grupos RaaS emplean plantillas visuales o frameworks similares para desplegar sus portales de extorsión, blogs de filtración o interfaces de contacto con las víctimas. El análisis sistemático de la estructura HTML, las cadenas de texto presentes en las capturas o incluso la aplicación de OCR sobre screenshots puede revelar patrones de reutilización visual que refuerzan la atribución técnica. Esta línea podría enriquecerse con la comparación directa de portales .onion activos con filtraciones públicas de

kits RaaS, generando una base de huellas visuales y estructurales que complemente el análisis técnico clásico.

En este mismo sentido, una línea de trabajo particularmente interesante es la automatización completa del pipeline de análisis planteado en esta investigación. Actualmente, gran parte del proceso —desde la recolección de dominios hasta la ejecución de los scripts de fingerprinting, análisis de favicons y extracción de certificados TLS— puede ser orquestado de forma modular mediante entornos reproducibles como Docker. Una arquitectura automatizada permitiría ejecutar tareas de descubrimiento recurrente sobre fuentes como Ahmia o feeds OSINT, filtrar los resultados según criterios dinámicos, realizar escaneos activos o pasivos sobre los servicios encontrados y almacenar los resultados en una base de datos estructurada para su posterior análisis o visualización. Esta automatización escalaría el análisis a centenares o miles de dominios .onion, reduciendo significativamente la carga manual e incrementando la velocidad de detección de infraestructuras sospechosas. Además, permitiría generar alertas automáticas ante la reaparición de elementos técnicos previamente observados, como fingerprints conocidos, certificados TLS clonados o favicons ya relacionados con operaciones anteriores.

En definitiva, el trabajo aquí presentado constituye un punto de partida metodológico y técnico que puede ser extendido, automatizado y adaptado a múltiples contextos de aplicación. La combinación de técnicas de fingerprinting, análisis de errores de configuración, correlación de recursos y desanonimización parcial demuestra que, incluso en entornos diseñados para maximizar la privacidad y el anonimato, es posible extraer inteligencia técnica significativa mediante un enfoque riguroso, replicable y automatizable.

6

Conclusiones

El presente trabajo ha demostrado que es posible aplicar técnicas de *fingerprinting*, análisis pasivo de cabeceras, favicons, ETags y certificados TLS sobre servicios ocultos de la red Tor, con el fin de caracterizar y agrupar dominios .onion relacionados con campañas de ransomware. A pesar del alto grado de anonimato inherente a esta red, la investigación ha revelado patrones técnicos replicados, configuraciones reutilizadas y elementos de infraestructura comunes, lo cual permite reducir el espacio de atribución y plantear hipótesis operacionales con mayor solidez.

De los resultados obtenidos se desprenden varias conclusiones relevantes:

- **Estandarización en la configuración:** el análisis de más de un centenar de servicios evidenció una diversidad muy limitada de huellas digitales de cabeceras (solo 16 valores únicos de *key_fp* y 22 de *kv_fp*). Esta baja entropía apunta al uso de plantillas, despliegues automatizados o configuraciones por defecto, lo que refuerza la hipótesis de entornos clonados o replicados para diferentes portales de ransomware.
- **Valor de los ETags como identificadores persistentes:** se recopilaron 45 ETags únicos, de los cuales cuatro fueron correlacionados con servidores de la *clearnet* mediante FOFA. Este hallazgo confirma que la reutilización de recursos (p. ej., ficheros CSS o JavaScript) entre Tor y la web pública constituye un vector de desanonimización parcial. Además, los ETags permitieron establecer vínculos operativos entre **INC Ransomware** y su sucesor **LYNX**, mostrando continuidad técnica en el uso de portales e infraestructuras. La ausencia de rotación de ETags y la falta de entropía en sus valores suponen un error de OPSEC que facilita su explotación como huella digital.
- **Favicons y correlación interdominios:** el uso de hashes Murmur3 de favicons permitió agrupar servicios .onion que compartían iconos por defecto o plantillas visuales. Al combinar esta información

con las huellas de cabeceras y los ETags, se refuerza la capacidad de correlación y atribución de infraestructuras aparentemente independientes.

- **Certificados TLS como indicador de negligencia operativa:** aunque su uso en servicios .onion es limitado, los certificados recopilados mostraron configuraciones erróneas o autofirmados mal estructurados. Estos fallos, además de reducir la seguridad de las conexiones, constituyen otro punto de apoyo para los esfuerzos de inteligencia técnica.

En conjunto, los resultados evidencian que incluso en entornos diseñados para garantizar el anonimato, la falta de buenas prácticas operativas y la reutilización de configuraciones abren la puerta a la extracción de inteligencia técnica significativa. Desde una perspectiva práctica, los hallazgos de este TFM pueden ser aprovechados por equipos de ciberinteligencia, CERTs o fuerzas de seguridad, al ofrecer una metodología replicable y basada en fuentes abiertas para el mapeo de infraestructura maliciosa en redes anónimas.

Finalmente, se recomienda continuar esta línea de investigación con un enfoque longitudinal, que permita estudiar la evolución temporal de los ETags y otras huellas técnicas, así como valorar el impacto de las configuraciones inseguras en la exposición inadvertida de los operadores de ransomware.

A

Apéndice - Scripts

A.1 SCRIPT FINGERPRINTING.PY: ANÁLISIS DE ENCABEZADOS HTTP

El script `fingerprinting.py` fue desarrollado para analizar y caracterizar encabezados HTTP obtenidos a través de herramientas como GoWitness o ZGrab. A partir de un archivo en formato JSON que contiene respuestas HTTP de múltiples servicios (por ejemplo, `resultados-array.json`), el script extrae información estructurada que permite realizar tareas de *fingerprinting*, detección de configuraciones comunes y agrupación de sitios web con cabeceras similares o idénticas.

A continuación se resumen sus principales funcionalidades:

- **Carga y limpieza de datos:** Se procesan los encabezados HTTP, ignorando campos volátiles como `Date` o `ETag`, con el fin de evitar falsos positivos en las comparaciones. Para cada sitio, se generan dos huellas digitales:
 - `key_fp`: basada únicamente en los nombres de las cabeceras.
 - `kv_fp`: basada en los pares clave-valor completos.
- **Detección de huellas idénticas:** Se agrupan los sitios web que comparten exactamente la misma estructura de cabeceras (`kv_fp`), exportando los resultados en `identical_fingerprints.csv`. Esta operación permite identificar instancias potencialmente replicadas, como clones o paneles de administración reutilizados.
- **Detección de huellas similares:** Si la biblioteca `datasketch` está disponible, el script emplea la técnica de *MinHash* para detectar configuraciones de cabeceras parcialmente similares entre sitios (*near-duplicates*), con un umbral de similitud configurable (por defecto, 0.90). Los grupos detectados se guardan en `near_duplicates.txt`.

- **Estadísticas de servidores:** El campo `Server` es extraído y normalizado para todos los registros, generando un ranking de los servidores web más utilizados en el conjunto analizado. El resultado se almacena en `top_server.csv`.
- **Exportación de datos:** Todos los resultados relevantes se exportan en formato CSV o TXT, lo que permite su posterior análisis o visualización mediante herramientas externas.

En conjunto, este script constituye una herramienta esencial para la caracterización pasiva de servicios web en redes ocultas o entornos donde no es viable realizar escaneos activos, permitiendo establecer correlaciones entre servicios y evaluar la diversidad real de configuraciones HTTP.

```

1 #!/usr/bin/env python3
2
3 """
4     fingerprinting.py - Análisis de cabeceras HTTP (como los generados por gowitness, zgrab, etc.)
5
6 Requisitos:
7
8     pip install ujson pandas datasketch tqdm
9
10
11 import hashlib
12 import pathlib
13 import sys
14 from typing import Union, List, Dict, Set
15
16 #      dependencias básicas
17 try:
18     import ujson as json
19 except ModuleNotFoundError:
20     import json
21
22 import pandas as pd
23 from tqdm import tqdm
24
25 #      datasketch opcional
26 try:
27     from datasketch import MinHash, MinHashLSH

```

```

26     except ModuleNotFoundError:
27         print("![!] 'datasketch' no está instalado - se omite near-duplicate MinHash")
28         MinHash = MinHashLSH = None
29
30     #     configuración
31     PATH: str = "resultados-array.json"
32     OUT_DIR = pathlib.Path(".")
33     VOLATILES: Set[str] = {"date", "etag", "last-modified", "content-length"}
34     SIM_THRESHOLD: float = 0.90
35     PERMUTATIONS: int = 128
36
37     #     utilidades
38     def load_json_any(path: Union[str, pathlib.Path]) -> List[Dict]:
39         path = pathlib.Path(path)
40         try:
41             with path.open("rb") as f:
42                 return json.load(f)
43         except (json.JSONDecodeError, ValueError):
44             data: List[Dict] = []
45             with path.open("rb") as f:
46                 for n, line in enumerate(f, 1):
47                     line = line.strip()
48                     if not line:
49                         continue
50                     try:
51                         data.append(json.loads(line))
52                     except (json.JSONDecodeError, ValueError) as e:
53                         print(f"![!] Línea {n} ignorada: {e}", file=sys.stderr)
54             return data
55
56     def canon_pairs(hdrs: List[Dict], ignore_vals: bool = False):
57         if ignore_vals:
58             return tuple(
59                 sorted(
60                     h["key"].lower()

```

```

61             for h in hdrs
62                 if h["key"].lower() not in VOLATILES
63             )
64         )
65     return tuple(
66         sorted(
67             (h["key"].lower(), h["value"].strip())
68             for h in hdrs
69             if h["key"].lower() not in VOLATILES
70         )
71     )
72
73 def sha1_short(s: Union[str, bytes]) -> str:
74     return hashlib.sha1(s if isinstance(s, bytes) else s.encode()).hexdigest()[:12]
75
76 #     carga
77 print(f"[*] Cargando '{PATH}'...")
78 records = load_json_any(PATH)
79 print(f"    → {len(records)} registros\n")
80
81 #     pre-procesado
82 rows = []
83 for site in tqdm(records, desc="Procesando"):
84     hdrs = site.get("headers", [])
85     keys_only = canon_pairs(hdrs, ignore_vals=True)
86     kv_pairs = canon_pairs(hdrs)
87
88     rows.append({
89         "url" : site.get("final_url") or site.get("url"),
90         "key_fp" : sha1_short(repr(keys_only)),
91         "kv_fp" : sha1_short(repr(kv_pairs)),
92         "server" : next((h["value"] for h in hdrs if h["key"].lower() == "server"),
93                         "").lower(),
94         "num_headers" : len(hdrs),
95         "headers_set" : set(keys_only),

```

```

95     })
96
97 df = pd.DataFrame(rows)
98
99 # Mapeamos URL a kv_pairs para usarlos luego en el CSV
100 url_to_kv_pairs = {
101     site.get("final_url") or site.get("url"): canon_pairs(site.get("headers", []))
102     for site in records
103 }
104
105 print(df.head(3), "\n")
106
107 #      1) Huellas idénticas de cabeceras
108 identical = df.groupby("kv_fp").filter(lambda g: len(g) > 1)
109 # Añadimos columna con los headers clave-valor
110 identical["kv_pairs"] = identical["url"].map(url_to_kv_pairs)
111
112 identical.to_csv(OUT_DIR / "identical_fingerprints.csv", index=False)
113 print(f"[+] identical_fingerprints.csv ({len(identical)})")
114
115 #      2) Near-duplicates usando MinHash
116 near_path = OUT_DIR / "near_duplicates.txt"
117 if MinHash and len(df) > 1:
118     lsh = MinHashLSH(threshold=SIM_THRESHOLD, num_perm=PERMUTATIONS)
119     minhashes: List[MinHash] = []
120
121     for i, headers in enumerate(df.headers_set):
122         m = MinHash(num_perm=PERMUTATIONS)
123         for k in headers:
124             m.update(k.encode())
125         lsh.insert(i, m)
126         minhashes.append(m)
127
128     clusters, visited = [], set()
129     for i in range(len(df)):

```

```

130     if i in visited:
131         continue
132     g = set(lsh.query(minhashes[i]))
133     if len(g) > 1:
134         clusters.append(g); visited |= g
135
136     with near_path.open("w") as out:
137         for g in clusters:
138             for idx in g:
139                 out.write(df.loc[idx, "url"] + "\n")
140             out.write("---\n")
141     print(f"[+] near_duplicates.txt      ({len(clusters)} grupos)")
142 else:
143     print("[-] MinHash omitido (datasketch no disponible)")
144
145 #      4) Estadísticas de 'Server'
146 df.server.value_counts().to_csv(OUT_DIR / "top_server.csv", header=["count"])
147 print(f"[+] top_server.csv")
148
149 print("\n Proceso completado")
150

```

A.2 SCRIPT FAVICON.PY: EXTRACCIÓN Y ANÁLISIS DE FAVICONS

El script `favicon.py` ha sido desarrollado para automatizar la extracción, análisis y correlación de archivos `favicon.ico` de servicios accesibles mediante dominios `.onion`. Su funcionamiento se basa en los siguientes pasos:

- **Carga de URLs:** Se procesan los resultados almacenados en el fichero `resultados-array.json`, seleccionando únicamente aquellos que han respondido con código HTTP 200.
- **Descarga del favicon:** Para cada URL válida, se realiza una petición HTTP a través de la red Tor utilizando un proxy `socks5h`, con el objetivo de recuperar el archivo `favicon.ico`.
- **Cálculo de huellas:** Una vez descargado, el favicon es sometido a dos funciones de *hash*: MD5 y

Murmur3 (esta última tras codificar el contenido en Base64), generando identificadores únicos ampliamente utilizados en motores de búsqueda de servicios como Shodan, Censys y FOFA.

- **Consultas inversas:** A partir del valor Murmur3, se construyen automáticamente las consultas de búsqueda específicas para cada plataforma (`http.favicon.hash` en Shodan, `services.http/favicon_hash` en Censys, e `icon_hash` en FOFA). Si se dispone de una API key válida, el script consulta directamente FOFA para obtener el número de coincidencias globales.
- **Exportación:** Los resultados se almacenan en el fichero `favicons_hashes.csv`, incluyendo para cada URL su hash MD5, su hash Murmur3, las consultas generadas para cada motor y el número de resultados obtenidos desde FOFA (si aplica).

```
1 #!/usr/bin/env python3
2
3 """
4 favicon.py - MD5 + Murmur3 de favicons (.onion), integración con FOFA API (solo con API key)
5
6 import base64, hashlib, json, pathlib, sys, time
7 from typing import Dict, List, Union, Optional
8 from urllib.parse import urlparse, quote
9
10 import mmh3, requests
11 from tqdm import tqdm
12
13 #     CONFIGURACIÓN
14 JSON_PATH = "resultados-array.json"
15 OUT_CSV    = "favicons_hashes.csv"
16 TOR_PROXY  = "socks5h://127.0.0.1:9050"
17 TIMEOUT    = 20
18
19 # CREDENCIAL FOFA - solo API key
20 FOFA_API_KEY = "..." # ← Rellena con tu API key
21 FOFA_API_URL = "https://fofa.info/api/v1/search/all"
22
23 #     FUNCIONES
```

```

24 def load_json_any(path: Union[str, pathlib.Path]) -> List[Dict]:
25     path = pathlib.Path(path)
26     try:
27         with path.open("rb") as f:
28             return json.load(f)
29     except (json.JSONDecodeError, ValueError):
30         data: List[Dict] = []
31         with path.open("rb") as f:
32             for n, line in enumerate(f, 1):
33                 if not (line := line.strip()):
34                     continue
35                 try:
36                     data.append(json.loads(line))
37                 except Exception:
38                     print(f"![{n}] Línea {n} ignorada (JSON inválido)", file=sys.stderr)
39     return data
40
41
42 def download_favicon(base_url: str, session: requests.Session) -> Optional[bytes]:
43     try:
44         p = urlparse(base_url)
45         url = f"{p.scheme or 'http'}://{(p.netloc or p.path).rstrip('/')}/favicon.ico"
46         r = session.get(url, timeout=TIMEOUT, verify=False)
47         r.raise_for_status()
48         return r.content or None
49     except Exception:
50         return None
51
52
53 def murmur3_from_bytes(blob: bytes) -> int:
54     return mmh3.hash(base64.b64encode(blob))
55
56
57 def fofa_search(query: str) -> int:
58     try:

```

```

59     q_base64 = base64.b64encode(query.encode()).decode()
60     url = f"{FOFA_API_URL}?key={FOFA_API_KEY}&qbase64={q_base64}&size=1"
61     r = requests.get(url, timeout=15)
62     if r.status_code == 200:
63         data = r.json()
64         return data.get("size", 0)
65     else:
66         print(f"[FOFA] Error HTTP {r.status_code} → {r.text}")
67 except Exception as e:
68     print(f"[FOFA] Error: {e}")
69 return -1
70
71 #     MAIN
72 def main() -> None:
73     raw = load_json_any(JSON_PATH)
74     ok = [o for o in raw if o.get("response_code") == 200]
75     urls = sorted({o.get("final_url") or o.get("url") for o in ok})
76
77     s = requests.Session()
78     s.proxies = {"http": TOR_PROXY, "https": TOR_PROXY}
79     s.headers.update({"User-Agent": "Mozilla/5.0"})
80
81     rows = [("url", "md5_hex", "murmur3_int", "shodan_query", "censys_query",
82             "fofa_query", "fofa_hits")]
83
84     for url in tqdm(urls, desc="Favicons"):
85         blob = download_favicon(url, s)
86         if not blob:
87             continue
88         md5_hex = hashlib.md5(blob).hexdigest()
89         mur     = murmur3_from_bytes(blob)
90
91         shodan_q = f"http.favicon.hash:{mur}"
92         censys_q = f"services.http.favicon_hash:{mur}"
93         fofa_q   = f'icon_hash="{mur}"'

```

```

94
95     hits = fofa_search(fofa_q)
96     time.sleep(1.2)  # evitar rate limit de FOFA
97
98     rows.append((url, md5_hex, mur, shodan_q, censys_q, fofa_q, hits))
99
100    pathlib.Path(OUT_CSV).write_text(
101        "\n".join(", ".join(map(str,r)) for r in rows), encoding="utf-8"
102    )
103    print(f"\n {len(rows)-1} favicons procesados (código 200). CSV → {OUT_CSV}")
104
105
106 if __name__ == "__main__":
107     main()
108

```

A.3 SCRIPT TLS_EXTRACT.PY: EXTRACCIÓN DE CERTIFICADOS TLS SOBRE TOR

El script `tls_extract.py` permite recuperar y analizar los certificados TLS presentados por servicios accesibles mediante dominios `.onion` que utilizan el protocolo HTTPS. Está diseñado para funcionar a través de la red Tor, utilizando un proxy local (`socks5h`) y procesando un conjunto de URLs almacenadas en el archivo `resultados-array.json`.

El flujo de trabajo del script puede resumirse en los siguientes pasos:

- **Carga de entradas:** Se leen los registros JSON de servicios previamente recopilados, filtrando aquellos que utilizan HTTPS y pertenecen al dominio `.onion`.
- **Conexión TLS sobre Tor:** Para cada host identificado, el script establece una conexión TLS a través del proxy Tor, desactivando la validación del certificado para garantizar la compatibilidad con certificados autofirmados o no reconocidos.
- **Extracción y análisis del certificado:** Una vez establecida la conexión, se obtiene el certificado

X.509 en formato DER y se extraen los campos más relevantes:

- Huellas digitales (SHA-1 y SHA-256).
 - Información del sujeto y emisor.
 - Periodo de validez (campos `notBefore` y `notAfter`).
 - Algoritmos de clave pública y firma.
 - Tamaño de la clave pública.
 - Extensiones como Key Usage, Extended Key Usage, Subject Alternative Name (SAN) y el indicador CA=TRUE.
- **Gestión de errores y reintentos:** Si se produce un error de conexión (por ejemplo, `timeout` durante el *handshake* TLS), el script reintenta la operación hasta un máximo definido. Si el error persiste, se registra como tal.
 - **Presentación de resultados:** Los datos se muestran por consola en un formato estructurado y legible, lo que facilita tanto la validación manual como el análisis cualitativo.

```
1 #!/usr/bin/env python3
2 """
3     tls_extract.py
4 ~~~~~
5     Recupera los certificados TLS de servicios .onion que usen HTTPS y los
6     muestra con información detallada (fingerprints, sujeto, emisor, fechas,
7     algoritmos, SAN, Key Usage, Extended KU, flag CA, etc.).
8
9 Requisitos:
10    pip install pysocks ujson cryptography
11    # y Tor escuchando en 127.0.0.1:9050
12 """
13
14 #             IMPORTS
15 import ssl
16 import socket
17 import socks
```

```

18 import pathlib
19 import warnings
20 from urllib.parse import urlparse
21 from datetime import datetime
22
23 # JSON rápido
24 try:
25     import ujson as json
26 except ModuleNotFoundError:
27     import json
28
29 # X.509 análisis
30 from cryptography import x509
31 from cryptography.hazmat.backends import default_backend
32 from cryptography.hazmat.primitives import hashes
33 from cryptography.x509.oid import ExtensionOID
34 from cryptography.utils import CryptographyDeprecationWarning
35
36 #     SILENCIAR ADVERTENCIAS MOLESTAS
37 warnings.filterwarnings("ignore", category=UserWarning)
38 warnings.filterwarnings("ignore", category=CryptographyDeprecationWarning)
39
40 #     CONFIG
41 TOR_HOST, TOR_PORT = "127.0.0.1", 9050 # proxy Tor local
42 INPUT_FILE = "resultados-array.json"      # fichero con URLs
43 TIMEOUT = 30                            # seg. para handshake
44 RETRIES = 2                             # reintentos si timeout
45 #
46
47
48 #     FUNCIONES UTILIDAD
49 def load_json(path: str):
50     """Carga JSON array o JSONL; devuelve lista de dicts."""
51     p = pathlib.Path(path)
52     try: # intenta JSON array

```

```

53     with p.open("rb") as f:
54         return json.load(f)
55     except Exception:
56         data = []
57         with p.open("rb") as f: # fallback a JSONL
58             for ln in f:
59                 ln = ln.strip()
60                 if ln:
61                     try:
62                         data.append(json.loads(ln))
63                     except Exception:
64                         pass
65         return data
66
67
68 def fp_hex_to_colon(hexstr: str, length: int) -> str:
69     """Convierte 'aabbc...' → 'aa:bb:cc', truncando a length*2 chars."""
70     h = hexstr[:length]
71     return ":".join(h[i:i + 2] for i in range(0, len(h), 2))
72
73
74 def get_ext(cert, oid):
75     """Devuelve extensión X.509 o None si no existe."""
76     try:
77         return cert.extensions.get_extension_for_oid(oid).value
78     except x509.ExtensionNotFound:
79         return None
80 #
81
82
83 #           DECODIFICAR CERT
84 def der_to_info(der: bytes):
85     """Extrae campos interesantes de un certificado DER."""
86     cert = x509.load_der_x509_certificate(der, default_backend())
87

```

```

88     pub = cert.public_key()
89     pub_alg = pub.__class__.__name__
90     pub_bits = getattr(pub, "key_size", "-")
91
92     san = get_ext(cert, ExtensionOID.SUBJECT_ALTERNATIVE_NAME)
93     sans = san.get_values_for_type(x509.DNSName) if san else []
94
95     ku = get_ext(cert, ExtensionOID.KEY_USAGE)
96     eku = get_ext(cert, ExtensionOID.EXTENDED_KEY_USAGE)
97     bc = get_ext(cert, ExtensionOID.BASIC_CONSTRAINTS)
98
99     # Lista legible de Key Usage
100    key_usage = []
101
102    if ku:
103        basic_flags = [
104            "digital_signature", "content_commitment", "key_encipherment",
105            "data_encipherment", "key_agreement", "key_cert_sign", "crl_sign"
106        ]
107
108        for flag in basic_flags:
109            if getattr(ku, flag):
110                key_usage.append(flag)
111
112            # encipher_only / decipher_only solo válidos si key_agreement=True
113            if ku.key_agreement:
114
115                if ku.encipher_only:
116                    key_usage.append("encipher_only")
117
118                if ku.decipher_only:
119                    key_usage.append("decipher_only")
120
121
122    return {
123        "fp_sha1": fp_hex_to_colon(cert.fingerprint(hashes.SHA1()).hex(), 12),
124        "fp_sha256": fp_hex_to_colon(cert.fingerprint(hashes.SHA256()).hex(), 16),
125        "serial": hex(cert.serial_number),
126        "version": cert.version.name,
127        "subject": cert.subject.rfc4514_string(),

```

```

123     "issuer": cert.issuer.rfc4514_string(),
124     "not_before": cert.not_valid_before_utc.isoformat(),
125     "not_after": cert.not_valid_after_utc.isoformat(),
126     "pub_alg": pub_alg,
127     "pub_bits": pub_bits,
128     "sig_alg": cert.signature_algorithm_oid._name,
129     "sig_hash": cert.signature_hash_algorithm.name,
130     "sans": sans,
131     "is_ca": (bc.ca if bc else False),
132     "key_usage": key_usage,
133     "ext_ku": [e._name for e in eku] if eku else [],
134 }
135 #
136
137
138 #           TLS VIA TOR
139 def get_tls_cert(host: str, timeout: int = TIMEOUT):
140     """Devuelve info de cert o {'error': ...}."""
141     ctx = ssl.create_default_context()
142     ctx.check_hostname = False
143     ctx.verify_mode = ssl.CERT_NONE
144
145     socks.setdefaultproxy(socks.PROXY_TYPE_SOCKS5, TOR_HOST, TOR_PORT, rdns=True)
146     s = socks.socksocket()
147     s.settimeout(timeout)
148
149     try:
150         s.connect((host, 443))
151         with ctx.wrap_socket(s, server_hostname=host) as ssock:
152             der = ssock.getpeercert(binary_form=True)
153             return der_to_info(der)
154     except Exception as e:
155         return {"error": f"{type(e).__name__}: {e}"}
156     finally:
157         try:

```

```

158         s.close()
159     except Exception:
160         pass
161 #
162
163
164 #           FILTRAR HOSTS HTTPS
165 def onion_https_hosts(records):
166     """Devuelve lista (con duplicados) de hosts .onion que usen https://."""
167     hosts = []
168     for r in records:
169         url = r.get("final_url", "")
170         if url.startswith("https://") and ".onion" in url.lower():
171             hosts.append(urlparse(url).hostname.lower())
172
173     return hosts
174 #
175
176 #           IMPRESIÓN FORMATEADA
177 def show(host: str, info: dict):
178     """Imprime bonito el resultado o el error."""
179     print(f"[~] TLS → {host}")
180     if "error" in info:
181         print(f"      {info['error']}\n")
182         return
183
184     print(f"      fp (SHA-256/16): {info['fp_sha256']}")
185     print(f"      fp (SHA-1/6)   : {info['fp_sha1']}")
186     print(f"      serial       : {info['serial']}    version: {info['version']}")
187     print(f"      subject      : {info['subject']}")
188     print(f"      issuer       : {info['issuer']}")
189     print(f"      valid UTC    : {info['not_before']} → {info['not_after']}")
190     print(f"      pubkey        : {info['pub_alg']} {info['pub_bits']} -bit")
191     print(f"      sig alg/hash : {info['sig_alg']} / {info['sig_hash']}")
192     if info["sans"]:

```

```

193     print(f"      SANS      : {', '.join(info['sans'])})")
194 if info["key_usage"]:
195     print(f"      Key Usage      : {', '.join(info['key_usage'])})")
196 if info["ext_ku"]:
197     print(f"      Ext Key Usage  : {', '.join(info['ext_ku'])})")
198 if info["is_ca"]:
199     print("          * Este certificado declara CA=TRUE *")
200 print()
201 #
202
203
204 def main():
205     t0 = datetime.utcnow()
206     print(f"[*] {t0.isoformat()} - Cargando '{INPUT_FILE}'...")
207     records = load_json(INPUT_FILE)
208     hosts = onion_https_hosts(records)
209     print(f"[+] {len(hosts)} hosts .onion con HTTPS.\n")
210
211     for host in hosts:
212         # Reintento en caso de timeout del handshake
213         attempt = 0
214         while attempt <= RETRIES:
215             result = get_tls_cert(host)
216             if "error" not in result or "timeout" not in result["error"].lower():
217                 break
218             attempt += 1
219             show(host, result)
220
221     dt = datetime.utcnow() - t0
222     print(f" Terminado en {dt.total_seconds():.1f}s")
223
224
225 if __name__ == "__main__":
226     main()

```

B

Apéndice - Cuadros

B.1 CUADRO FAVICONS

URL	Murmur3	FOFA Query
http://34o4m3f26ucyeddzpf53bksy76wd737nf2fytslowwd3viac3by5chad.onion	-491684375	icon_hash="-491684375"
http://3pktrcgbmssvrnwe5skburdwe2h3v6ibdnn5kbjqihs6eu6s6b7ryqd.onion	1497298042	icon_hash="1497298042"
http://7ukmkdtyxdkdivtjad57klqnd3kdsdq6tp45rrsxqnu76zzv3jvitlqd.onion	95028350	icon_hash="95028350"
http://aby6efzmp7jzbwgldgqc6ghxi2vypo6d7eaoood5xuoxutrfosmzczqd.onion	-2008527151	icon_hash="-2008527151"
http://basheqtqvzwz4vp6ks5lm2ocq7i6tozqgf6vjcasj4ezmsy4bkpshhyd.onion	-1729062639	icon_hash="-1729062639"
http://basheroq53eniermxovo3bkdwu5qqq5bkqcm13qictfmamgvzmovskyqd.onion	-1729062639	icon_hash="-1729062639"
http://basherykagbxoaiaxkgqhmhd5gbmedwb3di4ig3ouovziagosv4n77qd.onion	-1729062639	icon_hash="-1729062639"
http://bashete63b3gcijfopfw6fmn3rwnmyi5aclp55n6awcfbexivexhyad.onion	-1729062639	icon_hash="-1729062639"
http://bashex7mokreyoxl16wlswx14foi7okgs7or7aergnuiockuoq35yt3ad.onion	-1729062639	icon_hash="-1729062639"
http://black3gnkizshuynieigw6ejgpblb53mpasftzd6pydqpmq2vn2xf6yd.onion	488718742	icon_hash="488718742"
http://direwolfcdkv5whaz2spehizdg22jsuf5aeje4asmetpb6ri4jnd4qd.onion	1243379715	icon_hash="1243379715"
http://dreadytofatroptsdj6io713xptbet6onono2yv7jicoxknyazubrad.onion	1457792351	icon_hash="1457792351"
http://embargobe3n5okxyzqphpmk3moinoap2snz5k6765mvtkk7hh1544jid.onion	-41331478	icon_hash="-41331478"
http://gunrabxbig445sjqa535uaymzerj6fp4nwc6ngc2xughf2pedjdhk4ad.onion	1900856161	icon_hash="1900856161"
http://hellcatdcy653ma43t2ryf2ztw5yfanqsbfmapndbqvteh5itctoijyd.onion	-1414862290	icon_hash="-1414862290"
http://hellcatdnrsu4i5uctbklunpfyv2ppiioh5bsb3leu4dfgizinrve3gqd.onion	-1414862290	icon_hash="-1414862290"
http://hellcatdohzngkuh7zruzh12wojrawbnzbyzljtkw6iluv5ussfer4id.onion	-1414862290	icon_hash="-1414862290"
http://hptqq2o2qjva7lcaaq67w36jihzivkaitkexorauw7b2yul2z6zozpqd.onion	-318804344	icon_hash="-318804344"
http://ijzn3sicrcy7guixkzkib4ukbiilwc3xhnmb4mcbccnsd7j2rekvqd.onion	-808728822	icon_hash="-808728822"
http://incbacg6bfwtrlwdbqc55gsfl763s3twdtwhp27duik6s6rwdcityd.onion	1249114015	icon_hash="1249114015"
http://incblog6qu4y4mm4zvw5nrmue6qbwtgjsxpw6b7ixzssu36tsajlload.onion	2091856766	icon_hash="2091856766"
http://j5o5y2feotmhvr7cbcp2j2ewayv5mn5zen13joqwx67gtfcchvezjznad.onion	147884613	icon_hash="147884613"
http://ks5424y3wpr5zlug5c7i6svvxweinhbdcqcfnpktfcutrcfazzgz5id.onion	-968411615	icon_hash="-968411615"
http://lockbit7z36ynytxwjzuoao46ck7b3753gpedary3qvuizn3iczhe4id.onion	1150569800	icon_hash="1150569800"
http://lockbit7z3azdoxdpxzlisztufbc2fldagztdu47xyucp25p4xtqad.onion	1150569800	icon_hash="1150569800"

http://lockbit7z5ehshj6gzpetw5kso3onts6ty7wrnneya5u4aj3vzkeoaqd.onion	1150569800	icon_hash="1150569800"
http://lockbit7z5hfw6ywfuzipoa42tjlma13x5suuccngsamsqklww2xgyqd.onion	1150569800	icon_hash="1150569800"
http://lockbit7z6f3gu6rvysn5gjbsqj3hk3bvsg64ns6pjldqr2xvhvsyd.onion	1150569800	icon_hash="1150569800"
http://lockbit7z6rzyojiye437jp744d4uwtff7aq7df7gh2jvwqtv525c4yd.onion	1150569800	icon_hash="1150569800"
http://lynxblogco7r37jt7p5wrmfxzqze7ghxw6rihzkqc455qluacwotciyd.onion	1986960332	icon_hash="1986960332"
http://lynxblogijy4jfoblgix2klxmkbgee4leo euge7qt4fpfkj4zbi2sjyd.onion	1986960332	icon_hash="1986960332"
http://lynxblogmx3rbiwg3rpj4nds25hjsnrwpkxt5gaznetfikz4gz2csyad.onion	1986960332	icon_hash="1986960332"
http://lynxblogoxllth4b46cfwlop5pfj4s7dyv37yuy7qn2ftan6gd72hsad.onion	1986960332	icon_hash="1986960332"
http://lynxblogt watfsrwj3oatpejwxk5bngqcd5f7s26iskagfu7ouaomjad.onion	1986960332	icon_hash="1986960332"
http://lynxblogxutufossaeawlij3j3uikaloll15ko6grzhkwclrngrfoid.onion	1986960332	icon_hash="1986960332"
http://lynxchatly4z1udmhmi75jrwhycnoqvkb4prohxmyzf4euf5gjxroad.onion	-136913557	icon_hash="-136913557"
http://nalr2uqsave7y2r235am5jsfi klfjh5h4jc5nztu3rzvmhklwt5j6kid.onion	-1235857848	icon_hash="-1235857848"
http://nerqnacjmdy3obvevyol7qhazkwkv57dwqvye5v46k5bcujtfa6sduad.onion	-638554083	icon_hash="-638554083"
http://nitrogenczs1prh3xyw6l h5xyjvmsz7c iljoqxxknd7uymkfetfhgvqd.onion	225480400	icon_hash="225480400"
http://nspirebcv4sy3yydtaercuut34hwc4fsxqqv4b4ye4xmo6qp3vxhulqd.onion	-171828992	icon_hash="-171828992"
http://nspiremkiq44zcxjbgvab4mdedyh2pzj5kzbmvftcugq3mczx3dqogid.onion	-171828992	icon_hash="-171828992"
http://rhysidaf c61m7qa2mkiukbezh7zuth3i4wo f4mh2audkymscjm6yegad.onion	509701725	icon_hash="509701725"
http://rhysidaf ohrhyy2aszi7bm32tnjat5xri65fopcxkdfxhi4tidsg7cad.onion	509701725	icon_hash="509701725"
http://santat7kp1lt6iyvqbr7q4amdv6dzrh6paatvyrz17ry3zm72zigf4ad.onion	1196018593	icon_hash="1196018593"
http://silentbgdghp3zel dwpumnwabglreq17jcffhx5vqkvtf2lshc4n5zid.onion	-921038583	icon_hash="-921038583"
http://termiteuslbumdge2zmf mfc srvmv sfe4gvyudc5j6cdnisnhtftvokid.onion	-1703232255	icon_hash="-1703232255"
http://vg6xwkmyfiv r316qtqus7jykcuvgx6imegb73hqny2avxccnmqt5m2id.onion	-1332023843	icon_hash="-1332023843"
http://xfv4jzckytb4g3ckwemcny3ihv4i5p4lqzdpi624cxisu35my5fwi5qd.onion	-1265334674	icon_hash="-1265334674"
http://z3wqggtxtft7id3ibr7srivv5gjof5fwg76slewnzwwakjuf3nlhukdid.onion	1497298042	icon_hash="1497298042"
http://zktnif5vc khmz5tyruk p5bam atbfh kxjn b23rspsanyzywcrx3bvtqad.onion	2060145568	icon_hash="2060145568"
https://47glxkuxyayqrvugfumgsblrdagvrah7gttfscgzn56eyss5wg3uvmqd.onion	1565868303	icon_hash="1565868303"
https://hunters33mmcww7ek7q5ndahul6nmzmrsumfs6aenichbqon6mxfigyd.onion	-850477437	icon_hash="-850477437"
https://hunters55atbdusuladzv7vzv6a423bkh6ksl2uftwrx yuarbz1fh7yd.onion	-850477437	icon_hash="-850477437"
https://hunters55rdxciehoqzwv7vgvv6nt37tbwax2reroyzxhou7my5ejyid.onion	-850477437	icon_hash="-850477437"

Cuadro B.1.1: Resumen de favicons: URLs, hashes Murmur3 y consultas FOFA asociadas.

Bibliografía

- [1] Ransomware.live. URL: <https://www.ransomware.live>.
- [2] MITRE Corporation. Mitre att&ck: Grupos de amenazas, 2024. URL: <https://attack.mitre.org/groups/>.
- [3] CrowdStrike. Advanced persistent threat (apt), 2024. URL: <https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/advanced-persistent-threat-apt/>.
- [4] European Union Agency for Cybersecurity (ENISA). Enisa threat landscape, 2024. URL: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024>.
- [5] The Raven File. Ioc - inc/lynx ransomware, 2024. URL: <https://github.com/TheRavenFile/IOC/blob/main/INC-Lynx%20Ransomware>.
- [6] Fortinet. Ransomware roundup: Lynx, 2024. URL: <https://www.fortinet.com/blog/threat-research/ransomware-roundup-lynx>.
- [7] MITRE ATT&CK. Group g1032 - lynx, 2024. URL: <https://attack.mitre.org/groups/G1032/>.
- [8] Palo Alto Networks. What is ransomware as a service (raas)?, 2024. URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-ransomware-as-a-service>.
- [9] Ahmia Project. Ahmia - search engine for tor hidden services, 2025. URL: <https://ahmia.fi>.
- [10] The Tor Project. Hidden service protocol, 2024. URL: <https://community.torproject.org/onion-services/overview/>.
- [11] ProjectDiscovery. Projectdiscovery - open source security tools, 2024. URL: <https://projectdiscovery.io>.
- [12] Google Cloud Security. Grupos apt: conocimiento y seguimiento de amenazas persistentes avanzadas, 2024. URL: <https://cloud.google.com/security/resources/insights/apt-groups?hl=es>.

[13] sensepost. Gowitness - a web screenshot utility using chrome headless, 2023. URL: <https://github.com/sensepost/gowitness>.