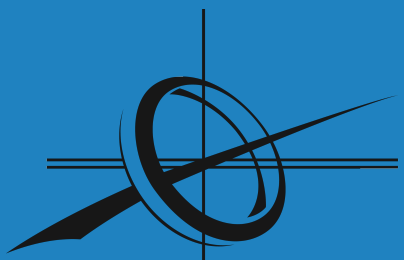
POLITÉCNICA



Universidad Politécnica de Madrid

ETSI SISTEMAS INFORMÁTICOS

Despliegue de una red de honeypots para campañas de defensa activa

Proyecto Fin de Grado

Grado en Ingeniería de Computadores

Autor:
Javier Olaya García

Tutores:
Borja Bordel Sánchez

09 de marzo de 2021

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Despliegue de una red de honeypots para campañas de defensa activa

Proyecto Fin de Grado

Grado en Ingeniería de Computadores

Curso académico 2020-2021

Autor:
Javier Olaya García

Tutores:
Borja Bordel Sánchez

Quisiera agradecer a mi mujer y mi hija que día a día,
con su paciencia y amor, me impulsan a seguir adelante,
además de saber que mis logros son también los suyos.

Resumen

El panorama actual, en el que los datos son cada vez más importantes para el éxito de los negocios y en el que la crisis causada por la enfermedad del coronavirus (COVID-19) ha acelerado el proceso de transformación digital de las empresas de una manera vertiginosa, los ciberdelincuentes se frotan las manos debido al crecimiento de manera exponencial de oportunidades para exfiltrar información corporativa, robar propiedad intelectual, extorsionar, sabotear, etc.

Se utilizan diferentes tecnologías que actúan de manera reactiva, Sistemas de Detección de Intrusos (IDS), Sistemas de Prevención de Intrusos (IPS), firewalls y muchos otros durante las 24 horas del día para hacer frente a los ciberataques. Sin embargo, se necesitan herramientas proactivas para comprender realmente el motivo por el cual un ciberdelincuente está acometiendo un ciberataque o para anticiparse a sus movimientos.

El proyecto presentado en esta memoria, implementa el prototipo de una infraestructura capaz de generar información para que un analista de seguridad pueda crear inteligencia sobre ciberamenazas recogidas en el ciberespacio. Así, las personas encargadas de tomar decisiones en las distintas empresas u organizaciones podrán anticiparse a posibles ciberataques que puedan comprometer sus activos de negocio. Para ello, se ha modificado la base de T-Pot, una distribución Linux preparada por Deutsche Telekom para ejecutar múltiples herramientas y honeypots, y Elastic Stack, un conjunto de herramientas open source que permite recoger datos de cualquier tipo de fuente y en cualquier formato para realizar búsquedas. Además de otras herramientas actuales como Nginx, Ansible o Docker.

Finalmente, con la ayuda de este prototipo, se mostrarán dos casos de uso, diseñando campañas de defensa activa, de las que se pueden extraer interesantes conclusiones de los resultados obtenidos.

Abstract

In today's landscape, where data is increasingly important for business success and where the crisis caused by COVID-19 has accelerated the process of digital transformation of companies in a dizzying way, cybercriminals are rubbing their hands together due to the exponential growth of opportunities to exfiltrate corporate information, steal intellectual property, extort, sabotage, etc.

Different technologies that act reactively, IDS, IPS, firewalls and many others are used around the clock to deal with cyberattacks. However, proactive tools are needed to really understand why a cybercriminal is undertaking a cyberattack or to anticipate his movements.

The project presented in this report implements the prototype of an infrastructure capable of generating information for a security analyst to create intelligence on cyber threats gathered in cyberspace. In this way, decision-makers in different companies or organisations will be able to anticipate possible cyber-attacks that could compromise their business assets. For this purpose, the basis of T-Pot, a Linux distribution prepared by Deutsche Telekom to run multiple tools and honeypots, and Elastic Stack, a set of open source tools that allows collecting data from any type of source and in any format to perform searches, have been modified. In addition to other current tools such as Nginx, Ansible or Docker.

Finally, with the help of this prototype, two use cases will be shown, designing active defence campaigns, from which interesting conclusions can be drawn from the results obtained.

Índice general

Índice de figuras

Índice de tablas

Índice de Códigos

1 Introducción

1.1 Motivación

El panorama actual de ciberamenazas sigue creciendo y es cada vez más complejo. Los adversarios, encuentran nuevas formas de acceder a los sistemas de las organizaciones con el fin de exfiltrar información corporativa, robar propiedad intelectual, extorsionar, sabotear o por motivos puramente económicos.

El mercado de la ciberseguridad necesita evolucionar al mismo ritmo que crecen las técnicas ofensivas que utilizan los adversarios. Por ello, en estos últimos años surgen nuevas formas y herramientas de seguridad que mejoran las técnicas de defensa como son las técnicas de engaño y contrainteligencia en el mundo digital.

Las técnicas de engaño y contrainteligencia han sido empleadas desde el principio de los tiempos en el ámbito militar. La inteligencia, y más concretamente la información obtenida acerca del adversario a través del espionaje y otros medios, son elementos fundamentales para conseguir mantener la ventaja sobre los adversarios.

Se ha visto a lo largo de la historia con ejemplos interesantes desde la mitología griega, pero que sigue siendo igualmente necesario en tiempos más recientes. El Arte de la Guerra de Sun Tzu es, probablemente, el mejor libro de estrategia de todos los tiempos, inspiró a Napoleón, Maquiavelo, Mao Tse Tung y a muchos otras grandes estrategas a lo largo de la historia. Para Sun Tzu, conocer al enemigo es fundamental para prevalecer en el campo de batalla.

En el mundo digital, es a partir de 1989 cuando se empezó a hablar de técnicas de engaño con el libro de Cliff Stoll, «The Cuckoo's Egg». El concepto de honeypot siguió en desarrollo y mejorando durante esos años. Si se avanza hasta el año 2000, se ve una clara evolución con el «Honeynet Project» y, pocos años después, se empieza a oír hablar también de honeytokens. Se puede denominar este periodo como la era de los honeypots.

La detección de amenazas a través de técnicas de engaño es algo sencillo de entender, a medida que el atacante está buscando o moviéndose hacia un objetivo, éste accede a recursos monitorizados cuya función es la de actuar de cebos y generar alertas que son detectadas por el equipo defensor, permitiendo observar al intruso mientras efectúa su ataque u así obtener una visión global del mismo que permitirá recopilar información sobre sus motivaciones, objetivos y, en última instancia, incluso atribuir el ataque a un adversario concreto en base a todas las

evidencias recogidas.

El portal de Deutsche Telekom, como resultado de una alianza en ciberseguridad puesta en marcha por la German Federal Office for Information Security (BSI) y la German Federal Association for Information Technology, Telecommunications and New Media (BITKOM), ofrece una sonda open source llamada T-Pot basada en una distribución Debian preparada para ejecutar múltiples programas y herramientas de honeypot, actuando como señuelos para atraer ataques automatizados dirigidos a explotar vulnerabilidades de servicios de red, servicios de páginas web, smartphones y otro tipo de sistemas.

Con esta motivación, surge la idea de crear un entorno que permita diseñar campañas de ciberataques con datos recogidos por T-Pot. Así, las organizaciones podrán tomar mejores decisiones y optimizar sus capacidades defensivas.

1.2 Objetivos

Se exponen a continuación los **objetivos principales** del presente proyecto:

- Una infraestructura basada en T-Pot que permita realizar campañas de engaño mediante sondas, desplegadas en cualquier localización geográfica.
- Para demostrar su funcionamiento, se creará un prototipo formado por un **nodo principal**, que almacenará todos los logs de los ataques recibidos en las distintas sondas en una base de datos y los mostrará de una manera amigable, y una **sonda**, que enviará la información de los ataques recibidos al nodo principal. Se diseñará y desplegarán dos campañas de engaño y se mostrarán los resultados obtenidos.

Otros **objetivos implícitos** a tener en cuenta son los siguientes:

- Como consecuencia de los objetivos principales, se realizará un estudio previo sobre las distintas tecnologías necesarias para el funcionamiento, tanto del nodo principal como de la sonda, en el mercado actual.
- Se evaluarán los costes del proyecto para predecir los gastos que supondrá el despliegue del prototipo.
- Debido al dominio en el que se encuentra enmarcada la herramienta a diseñar (Inteligencia de ciberamenazas y honeypots), se introducirán los conceptos imprescindibles de este área del conocimiento de una manera breve.
- Garantizar que el desarrollo de la infraestructura sea técnicamente factible.
- Realizar dos casos de uso para poner en práctica conceptos teóricos.

1.3 Difusión del contenido generado

Siguiendo la filosofía de software libre, se ha creado un **repositorio** de **GitHub** con el código desarrollado y los archivos de configuración necesarios para llevar a cabo este proyecto. De esta forma, cualquier profesional será libre de usar el código, estudiarlo y adaptarlo a sus necesidades, distribuirlo, o trabajar de manera colaborativa mejorándolo y publicándolo.

Además, se hará uso de la **Wiki** propia de GitHub para añadir la documentación que se vaya generando.

También, se ha hecho uso de la plataforma **Youtube** para publicar algunos vídeos explicativos acerca de distintas instalaciones y configuraciones que se han realizado.

El contenido del repositorio de GitHub se puede consultar en el siguiente enlace `https://github.com/0l4y4/helioslove` y los enlaces a los vídeos publicados se podrán ir viendo a lo largo de este proyecto.

1.4 Reflexión sobre los impactos sociales y ambientales del trabajo realizado

Este proyecto tiene repercusión social debido a la naturaleza del campo en el que se ubica, la **ciberseguridad**.

En el panorama actual,está muy extendido el término de ciberseguridad, pero muy pocos son conscientes de sus implicaciones y como puede afectar a los derechos de los ciudadanos.

La ciberseguridad se podría definir como la práctica encargada de evitar que las distintas ciberamenazas pongan en riesgo los sistemas y redes de una organización, empresa o usuario doméstico. Sin embargo, este hecho no solo afecta a dispositivos como servidores, bases de datos, teléfonos móviles o Sistemas de Control Industrial (ICS), también pueden verse perjudicados los derechos fundamentales de las personas.

La ciberseguridad es necesaria para garantizar el honor, así como la intimidad personal y familiar de los ciudadanos.

La ciberseguridad, garantiza que el Reglamento General de Protección de Datos (RGPD), establecido por el Parlamento Europeo y Consejo de la Unión Europea, pueda tener éxito protegiendo la privacidad de los datos de las personas.

Un ataque con éxito en cualquier infraestructura crítica, como por ejemplo el suministro de agua, podría provocar el envenenamiento de una red de abastecimiento de agua potable. Una central eléctrica comprometida podría dejar de suministrar luz en hospitales. Tener el control de los semáforos de una ciudad podría provocar accidentes de tráfico a gran escala.

Además, la ciberseguridad también es importante para el desarrollo de la vida de las personas tal y como se conoce a día de hoy. El comercio electrónico no sería posible sin la confianza generada en los ciudadanos gracias a las transacciones seguras o, aplicaciones co-

mo ”WhatsApp”, no tendrían éxito si cualquiera pudiera capturar las conversaciones entre usuarios.

En general, no aplicar actividades de ciberseguridad tendría un impacto negativo, pudiendo afectar directamente al bienestar social de los ciudadanos y, este proyecto, cubre aspectos importantes en este área.

Cabe destacar también, el impacto social que puede suponer, como alternativa a la inscripción de software en el Registro de Propiedad Intelectual, el hecho de utilizar herramientas de **software libre** y **código abierto** durante el desarrollo del proyecto, y la publicación del código creado en un repositorio de GitHub, favoreciendo en la difusión sin restricciones del conocimiento generado.

Por otro lado, el despliegue de este proyecto pudiera ser arrastrado por el impacto negativo generalizado que pueden ocasionar en el medio ambiente las infraestructuras tecnológicas. La contratación de un servicio de VPS implica el consumo de recursos de un sistema en un Centro de Proceso de Datos (CPD). Éstos, consumen un alto porcentaje de la electricidad mundial y generan una gran cantidad de emisiones globales de CO2.

Como solución a este problema, se han encontrado proveedores de servicio de VPS comprometidos con el crecimiento sostenible que alimentan sus CPD con energías renovables, garantizando una emisión cero de CO2.

Por último, se ha querido plantear una cuestión ética en cuanto al uso de los **honeypots** se refiere. Si bien es cierto que su uso se considera totalmente lícito, existen opiniones diversas al respecto. Existe un sector que considera que, al igual que es ilegal atraer a alguien para que robe algo material, debería ser ilegal atraer a alguien para que cometa un delito informático. Sin embargo, para otro grupo de personas, su uso se puede considerar ético, ya que se pueden solucionar brechas de seguridad antes de ser explotadas sin causar ningún daño a terceros durante el proceso.

2 Estudio de viabilidad

A continuación se estudiarán distintas opciones arquitectónicas y tecnológicas para la consecución de una herramienta que cumpla con los objetivos propuestos en la sección 1.2 Objetivos.

2.1 Definición de requisitos

El propósito de esta sección es establecer un punto de partida, relativamente preciso, para el desarrollo del proyecto. A continuación, se enumerarán los requisitos funcionales y los no funcionales.

2.1.1 Requisitos funcionales

1. El nodo principal deberá ser capaz de recolectar en una base de datos toda la información posible de ciberataques que estén ocurriendo en las distintas sondas desplegadas.

2. El nodo principal deberá ser capaz de mostrar los datos recolectados en la base de datos mediante gráficas de barras, gráficas circulares, tablas, histogramas y mapas.

3. El usuario deberá poder desplegar una o varias sondas en cualquier momento.

4. El usuario deberá poder habilitar o deshabilitar uno o varios honeypots en una o varias sondas.

5. Las sondas desplegadas deberán enviar la información de los ciberataques recibidos al nodo principal.

6. El canal de comunicación por el cual las sondas enviarán la información a la base de datos deberá ser seguro.

7. El canal de comunicación por el cual los usuarios accederán al panel de información deberá ser seguro.

8. El acceso a los servicios del nodo principal será mediante un nombre de dominio y sus subdominios.

9. El acceso a los servicios del nodo principal será mediante un proxy inverso.
10. La conexión para administrar la sonda debe ser segura.

2.1.2 Requisitos no funcionales

1. El sistema operativo que se desplegará en las sonda deberá ser T-Pot.
2. Todas las herramientas utilizadas serán open-source.
3. El protocolo de seguridad para cifrar los canales de acceso a los distintos servicios será Secure Sockets Layer / Transport Layer Security (SSL/TLS).
4. La interfaz gráfica mostrará un panel global teniendo en cuenta todos los ciberataques registrados en cada sonda.
5. La interfaz gráfica permitirá al usuario ver paneles específicos de cada honeypot.
6. El nombre de dominio para acceder a los recursos deberá estar relacionado con Helios, Dios del sol, "que todo lo ve".
7. El protocolo para administrar las sondas remotamente deberá ser SSH.

2.2 Alternativas a la solución

Teniendo en cuenta los apartados de 1.2 Objetivos, 2.1.1 Requisitos funcionales y 2.1.2 Requisitos no funcionales, se buscarán alternativas en dos campos bien diferenciados: **arquitectónicamente** y **tecnológicamente**.

2.2.1 Alternativas arquitectónicas

2.2.1.1 Sonda

En primer lugar, se ha de tener en cuenta las características hardware mínimas requeridas para la instalación de T-Pot que aconseja Deutsche Telekom en su repositorio oficial.

- 8GB de memoria RAM.
- 128GB de almacenamiento (aconsejable SSD).
- Conexión de red mediante DHCP.
- Conexión de red sin ningún tipo de proxy configurado.

Una vez están claros las especificaciones hardware recomendadas del equipo en el que se va a desplegar la sonda, surgen dos vertientes en cuanto a dónde ubicarlo.

- La primera opción es hacerlo en un equipo ubicado en una **red doméstica**. Para ello, se necesitará configurar el router que provee el acceso a internet, abriendo los puertos necesarios y redirigiendo el tráfico de entrada hacia la sonda. Desde el punto de vista de la ciberseguridad podría considerarse peligroso para el resto de equipos conectados a la LAN, ya sea por cable o por WIFI, aunque es complicado, un atacante podría llegar a pivotar de la sonda al resto de equipos. Como recomendación, se sugiere hacerlo en un entorno controlado por un firewall, aislando la sonda en una Zona Desmilitarizada (DMZ). Un ejemplo para esta situación se sugiere en la siguiente imagen.

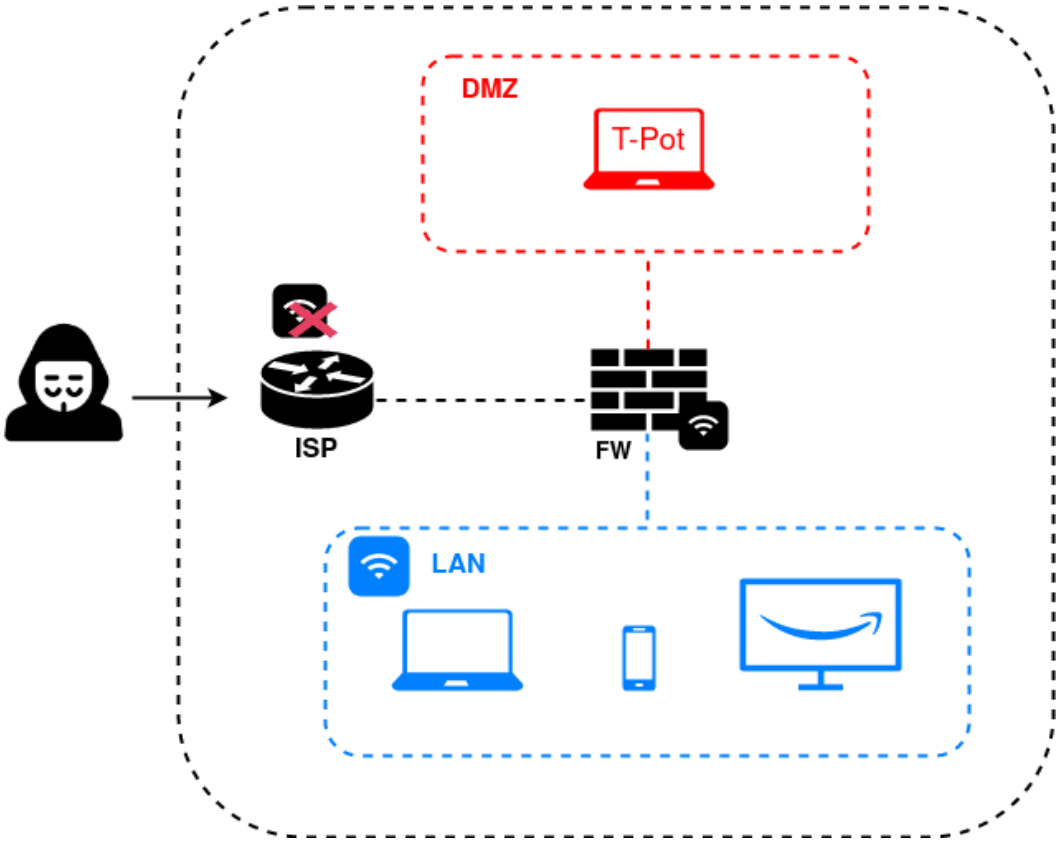


Figura 2.1: Estudio de viabilidad. T-Pot en una red doméstica.

- La segunda opción es hacerlo mediante un **VPS**. Con está alternativa, se pierde el control del equipo físicamente pero se evitan los problemas de seguridad que conlleva el caso anterior.
  Un aspecto importante a tener en cuenta a la hora de contratar el servicio de VPS es tener claro el tipo de instalación de T-Pot que se quiere desplegar (*véase* "4.3.1 Tipos

de instalación"), ya que para el tipo "iso" habrá que buscar proveedores que ofrezcan la posibilidad de instalar el sistema operativo mediante una imagen propia, debido a que la gran mayoría sólo ofrecen sistemas operativos modificados por ellos. En las siguiente tabla se puede observar un estudio previo de servicios VPS que permiten la instalación de imágenes propias.

Servicio de VPS	**Imagen propia**
`https://www.vpsag.com`	No
`https://www.xeonbd.com`	No
`https://www.a2hosting.es`	No
`https://www.hostens.com`	No
`https://www.vpsserver.com`	No
`https://www.cyberneticos.com`	No
`https://www.evidaliahost.com`	No
`https://javapipe.com`	Sí
`https://protonservers.com`	Sí
`https://www.cherryservers.com`	Sí
`https://rentvps.uk`	Sí
`https://www.hostwinds.com`	Sí
`https://www.ionos.es`	Sí
`https://www.liquidweb.com`	Sí

Tabla 2.1: Estudio de viabilidad. Estudio de servicios VPS que permiten iso propia.

2.2.1.2 Nodo principal

Teniendo en cuenta que el prototipo a desplegar en el actual proyecto solo va a constar de una sonda, se van a reutilizar las mismas recomendaciones de Deutsche Telekom, en cuanto a hardware se refiere, para el nodo principal. En caso de querer ampliar el número de sondas, se debería realizar un estudio previo de los recursos necesarios para el nodo principal, ya que dependiendo del número de sondas los recursos pueden variar ampliamente. Como recomendación, para entornos grandes y con muchos cambios según la campaña implementada, se debería hacer uso de servicios en la nube del tipo Infraestructuras como Servicio (IaaS) que permiten ampliar o disminuir recursos a conveniencia. Este estudio queda fuera del alcance del prototipo.

De igual forma que para la sonda, el despliegue del nodo principal se podría realizar mediante la contratación de un VPS o en una red doméstica. En el caso de esta última, se recomienda, como en el caso de la sonda, aislarlo en una DMZ y acceder al panel de visualización desde la red LAN. Un ejemplo se puede apreciar en la siguiente imagen.

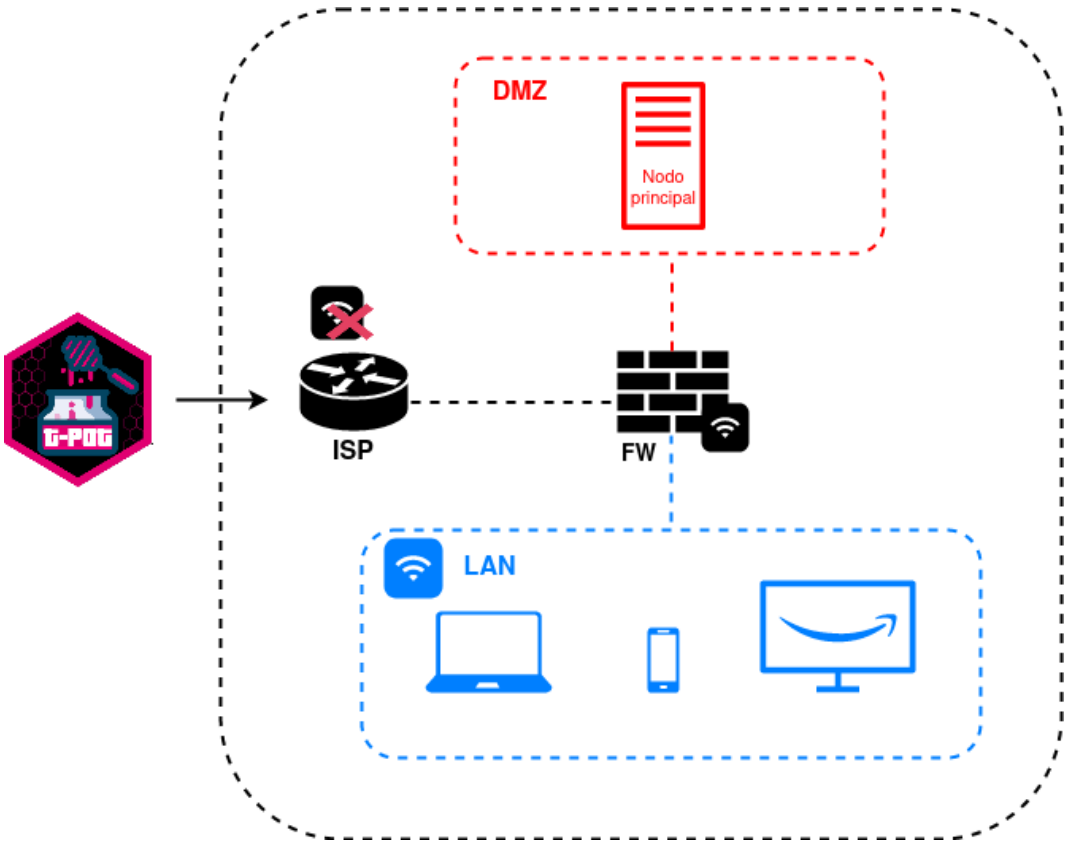


Figura 2.2: Estudio de viabilidad. Nodo principal en una red doméstica.

2.2.2 Alternativas tecnológicas

2.2.2.1 Sonda

Aunque la sonda como componente dentro de la infraestructura ya viene fijada en el apartado de requisitos no funcionales con T-pot, existen distintas posibilidades de despliegue. Deutsche Telekom, ha diseñado las ediciones **Standard**, **Sensor**[1], **Industrial**, **Collector**, **NextGen** y **Medical**, cada una de ellas con herramientas y honeypots enfocados a distintas necesidades, como se puede ver a continuación.

[1]El hardware requerido para esta instalación solo necesita 4GB de RAM debido a que no hace uso de ELK

Herramientas	Standar	Sensor	Industrial	Collector	Nextgen	Medical
Cockpit	Sí	Sí	Sí	Sí	Sí	Sí
Cyberchef	Sí	No	Sí	Sí	Sí	Sí
ELK	Sí	No	Sí	Sí	Sí	Sí
Fatt	Sí	Sí	Sí	Sí	Sí	Sí
Elasticsearch Head	Sí	No	Sí	Sí	Sí	Sí
Ewsposter	Sí	Sí	Sí	Sí	Sí	Sí
Nginx	Sí	No	Sí	Sí	Sí	Sí
Spiderfoot	Sí	No	Sí	Sí	Sí	Sí
P0f	Sí	Sí	Sí	Sí	Sí	Sí
Suricata	Sí	Sí	Sí	Sí	Sí	Sí

Tabla 2.2: Estudio de viabilidad. Herramientas según el modo de instalación de T-Pot

Honeypots	Standar	Sensor	Industrial	Collector	Nextgen	Medical
Adbhoney	Sí	Sí	No	No	Sí	No
Ciscoasa	Sí	Sí	No	No	Sí	No
Citrixhoneypot	Sí	Sí	No	No	Sí	No
Conpot	Sí	Sí	Sí	No	Sí	No
Cowrie	Sí	Sí	Sí	No	Sí	No
Dicompot	Sí	Sí	Sí	No	Sí	No
Dionaea	Sí	Sí	No	No	Sí	No
Glutton	No	No	No	No	Sí	No
Elasticpot	Sí	Sí	No	No	No	No
Heralding	Sí	Sí	Sí	Sí	Sí	No
Honeypy	No	Sí	No	No	Sí	No
Honeysap	Sí	Sí	Sí	No	Sí	No
Ipphoney	No	No	No	No	Sí	No
Honeytrap	Sí	Sí	Sí	Sí	No	No
Mailoney	Sí	Sí	No	No	Sí	No
Medpot	Sí	Sí	Sí	No	Sí	Sí
Rdpy	Sí	Sí	Sí	No	Sí	No
Snare	Sí	Sí	No	No	Sí	No
Tanner	Sí	Sí	No	No	Sí	No

Tabla 2.3: Estudio de viabilidad. Honeypots según el modo de instalación de T-Pot

Además, Deutsche Telekom ofrece la posibilidad de modificar los ficheros de configuración y crear una **imagen propia** personalizada. Para ello, habrá que modificar el archivo de configuración de los dockers "docker-compose" que se encuentra en la ruta "/opt/tpot/etc/compose".

En cuanto al envío de logs, se han evaluado **Logstash** y **Rsyslog**.

Rsyslog, instalado en la mayoría de distribuciones Linux, es capaz de buscar, analizar, almacenar y enviar archivos de logs a varios destinos.

Logstash, de Elastic, puede recolectar prácticamente cualquier tipo de datos, analizarlo y guardarlo o enviarlo a su destino.

La diferencia principal entre ambos es que aún siendo Rsyslog mucho más rápido, es mucho más difícil de configurar y hay mucha menos documentación.

2.2.2.2 Nodo principal

Una parte importante de este proyecto será almacenar la información que las sondas recojan en una base de datos centralizada. Además, esa información deberá ser explotada de manera óptima para que latoma de decisiones. Hoy en día, existen multitud de tecnologías que pueden lograrlo como MongoDB, MySQL, Elasticsearch, Splunk, Kibana, GrayLog y un largo etcétera. En este ámbito, se han evaluado las dos opciones más demandadas actualmente: **Elasticsearch + Kibana** y **Splunk**.

Los tres componentes clave de **Splunk** son su reenviador, que envía datos a los indexadores, su indexador, que tiene funciones para almacenar e indexar datos y responder a solicitudes de búsqueda y su panel de búsqueda. Además, Splunk utiliza un lenguaje de búsqueda patentado, Search Processing Language (SPL), para atender y ejecutar consultas contextuales de grandes conjuntos de datos.

Elasticsearch y **Kibana** son dos productos de código abierto, desarrollados y mantenidos por Elastic. Elasticsearch es una base de datos NoSQL que utiliza el motor de búsqueda Lucene, mientras que Kibana es un panel que funciona sobre Elasticsearch y facilita el análisis de datos mediante paneles atractivamente visuales.

La diferencia principal es que Splunk es un único producto de código cerrado, mientras que Elasticsearch y Kibana no. Elasticsearch y Kibana son gratuitas, con algunas funcionalidades de pago, y una gran variedad de plugins, sin embargo, Splunk es de pago, aunque tiene una prueba del producto gratuita de 60 días.

Otro apartado a tener en cuenta para cumplir con los requisitos funcionales, será la adquisición de un nombre de dominio y un certificado SSL. Dado que existen multitud de registradores de dominio y autoridades certificadoras de certificados SSL, se va a realizar un breve análisis desde dos puntos de vista: **gratuítos** y **de pago**. En las siguientes tablas se pueden ver algunos ejemplos.

Top Level Domain (TLD)	Precio	Registrador de nombres
.com	De pago	`https://www.domain.com/`
.net	De pago	`https://www.domain.com/`
.org	De pago	`https://www.domain.com/`
.es	De pago	`https://www.webnode.com/`
.ga	Gratis	`https://www.freenom.com/`
.tk	Gratis	`https://www.freenom.com/`
.cf	Gratis	`https://www.freenom.com/`

Tabla 2.4: Estudio de viabilidad. Nombres de dominio

Autoridad de certificación	Precio
`https://www.dondominio.com/`	De pago
`https://www.rapidsslonline.com/`	De pago
`https://www.geotrust.com/`	De pago
`https://letsencrypt.org/es/`	Gratis

Tabla 2.5: Estudio de viabilidad. Certificados SSL

Para una mejor gestión del acceso a los recursos se controlará el tráfico entrante (comunicación de la sonda con la base de datos y del cliente con la interfaz visual), mediante un proxy inverso. Además, proporcionará una capa extra de seguridad ante posibles ataques como podrían ser ataques de denegación de servicio (DDOS). Para ello se han evaluado tanto **Nginx** como **Apache**.

Nginx es un servidor web de código abierto que junto con Apache, ocupan prácticamente todo el mercado de servidores web.

A nivel técnico, usa una arquitectura de **subproceso asíncrono**, no crea procesos nuevos en el servidor, logrando un mayor rendimiento. Además, está disponible para todas las versiones de Linux.

Apache es a día de hoy el servidor web más popular. Es también de código abierto y, como Nginx, está disponible para casi todas las versiones de Linux.

Gracias a sus módulos multiproceso para mantener la misma conexión, los administradores pueden aplicar criterios para cada una de ellas. Sin embargo, debido a su consumo, necesita más recursos.
Apache, aunque es una tecnología más antigua, sigue ocupando la primera posición en el mercado, por lo que una ventaja frente a Nginx es que tiene un mayor apoyo de la comunidad de código abierto.
En cuanto a políticas de seguridad, ambos publican parches y actualizaciones constantemente.
Para evitar colapsos o una navegación lenta, el servidor web debe ser lo más rápido posible. En este sentido, Nginx responde mejor a las peticiones de los clientes, además de hacer un consumo más eficiente de los recursos hardware, principalmente la memoria RAM. Finalmente, Apache es mucho más flexible gracias a sus más de 60 módulos.

En cuanto a la administración remota de equipos mediante procesos automatizados, se han analizado Ansible y Puppet.
Aunque las dos podrían cumplir con el propósito de este proyecto, Ansible encajaría mejor a la hora de desplegar un entorno con muchas sondas, ya que solo necesita una conexión Secure SHell (SSH), mientras que Pupplet necesita instalar agentes en cada sonda.
Además, el código para automatizar procesos en Ansible, mediante ficheros YAML Ain't Markup Language (YAML)[2], lo hace mucho más sencillo de implementar y de interpretar.

2.3 Solución elegida

Como conclusión al estudio de las alternativas, finalmente se ha elegido la siguiente solución:

Se implementarán tanto el nodo principal como la sonda cada uno en su correspondiente **VPS**.

La base de datos elegida a desplegar en el nodo principal será **Elasticsearch** y la interfaz de visualización de los datos **Kibana** por el hecho de ser de código abierto, gratuitas y poder reutilizar configuraciones del propio T-Pot.

En la sonda se desplegará una **imagen de T-Pot presonalizada** con los siguiente honeypots y herramientas: Adbhoney, Ciscoasa, Citrixhoneypot, Conpot, Cowrie, Dicompot, Dionaea, Elasticpot, Heralding, Honeysap, Honeytrap, Mailoney, Medpot, Rdpy, Snare, Tanner, Fatt, P0f y Suricata. Además, se instalará **Logstash** para enviar los datos recogidos de los distintos ciberataques, integrándose perfectamente con los demás componentes de Elastic a instalar en el nodo principal.

En cuanto al proxy inverso, se ha elegido **Nginx** por su tendencia al alza en cuanto a popularidad se refiere.

[2]Lenguaje orientado a archivos de configuración diseñado para ser interpretado de una manera fácil por las personas.

El Top Level Domain (TLD) elegido será "**.com**" resultando en `www.helioslove.com`, el subdominio `www.sensors.helioslove.com` para el acceso a la base de datos Elasticsearch y el subdominio `www.god.helioslove.com` para el acceso a Kibana y sus paneles de visualización. Los certificado SSL gratuitos de "**Let's Encrypt**", uno para cada dominio y subdominio.

Finalmente, para el mantenimiento de la infraestructura y despliegue masivo de sondas se utilizará **Ansible**.

En la siguiente figura se reflejan los distintos componentes respecto a la solución elegida.

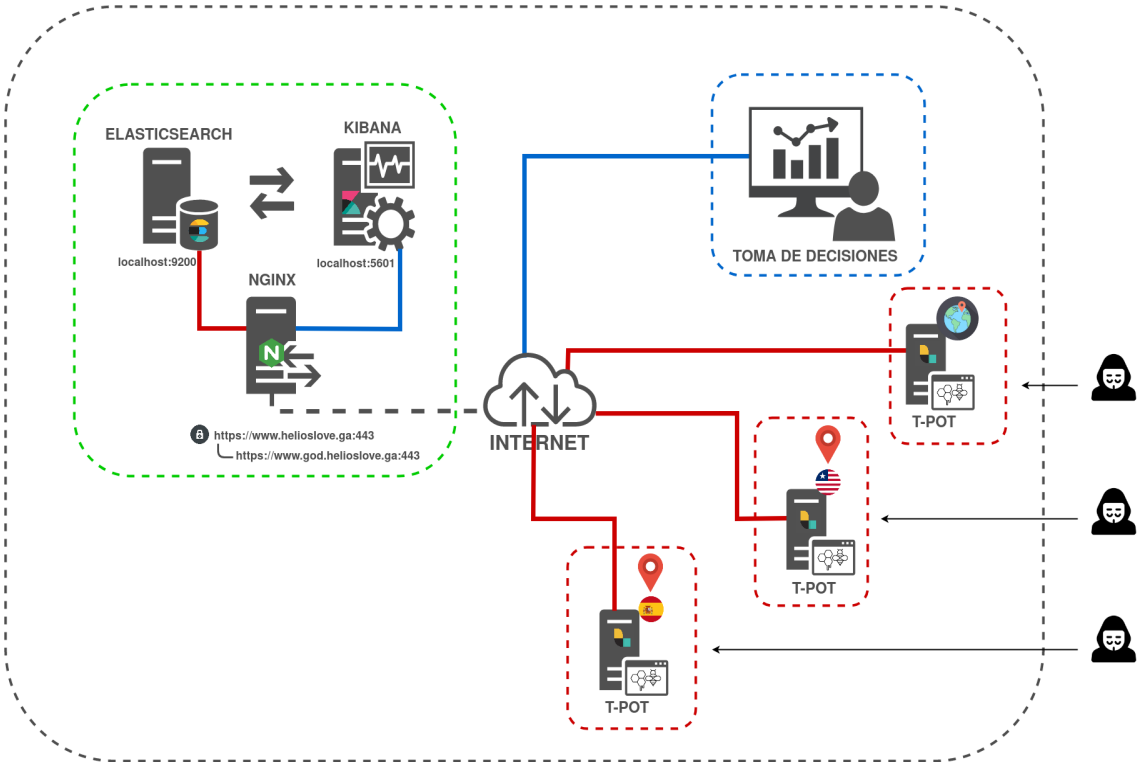


Figura 2.3: Estudio de viabilidad. Solución elegida.

3 Gestión del proyecto

En este apartado, se tendrá en cuenta los siguientes aspectos:

- Despliegue de la infraestructura.
- Equipo de trabajo.
- Coste económico.

3.1 Despliegue de infraestructura

Se estima que será necesario la contratación de dos VPS con al menos las siguientes características:

VPS	
Procesador	8 CPU cores
Memoria RAM	16GB
Capacidad de almacenamiento	256GB / SSD
Capacidad de instalar ISO propia	No aplica

Tabla 3.1: Gestión del proyecto. Características hardware mínimas requeridas de los VPS

3.2 Equipo de trabajo

3.2.1 Composición del equipo de trabajo

Se estima que será necesaria la participación de un equipo compuesto, al menos, por los siguientes perfiles durante toda la ejecución del proyecto:

- Un Jefe de Proyecto.
- Un Administrador de Sistemas Senior.
- Un Administrador de Sistemas Junior.

3.2.2 Dedicación de los componentes del equipo de trabajo

Tras un estudio de las tareas a realizar en este proyecto se estima que para cumplir los objetivos, las horas empleadas por el equipo propuesto y distribuidas por perfiles no deberá ser inferior a lo determinado en la siguiente tabla.

Perfil	**Nº de horas estimadas**
Jefe de proyecto	100
Administrador de sistemas Senior	180
Administrador de sistemas Junior	140

Tabla 3.2: Gestión del proyecto. Número de horas estimadas por perfil

3.2.3 Área de capacidades y conocimiento

Para llevar a cabo las tareas descritas en este documento, el personal del equipo de trabajo debe tener unos conocimientos y una experiencia propios que se relacionan en la siguiente tabla.

Jefe de Proyecto	**Experiencia mínima (años)**	**Cumplimiento**
Gestión de proyectos	1	Obligatorio
Administración de sistemas operativos Linux/Unix	2	Obligatorio
Certificación Linux Server Professional (LPIC-1)	–	Aconsejable
Administración de redes	2	Obligatorio
Automatización / Scripting	3	Obligatorio
Herramienta de administración Ansible	1	Aconsejable
Despliegue de aplicaciones con Docker	1	Obligatorio
Gestión de Bases de Datos	1	Obligatorio
Conceptos avanzados de hardware de servidores	2	Obligatorio
Certificados PKI	1	Obligatorio
Ciberseguridad	3	Obligatorio
Certificación Certified Ethical Hacker (CEH)	–	Aconsejable

Tabla 3.3: Gestión del proyecto. Capacidades y conocimientos del jefe proyecto.

Administrador de Sistemas Senior	**Experiencia mínima (años)**	**Cumplimiento**
Gestión de proyectos	No aplica	No aplica
Administración de sistemas operativos Linux/Unix	1	Obligatorio
Certificación LPIC-1	–	Aconsejable
Administración de redes	1	Obligatorio
Automatización / Scripting	1	Obligatorio
Herramienta de administración Ansible	1	Obigatorio
Despliegue de aplicaciones con Docker	1	Obligatorio
Gestión de Bases de Datos	1	Obligatorio
Conceptos avanzados de hardware de servidores	1	Aconsejable
Certificados PKI	1	Obligatorio
Ciberseguridad	1	Obligatorio
Certificación CEH	–	Aconsejable

Tabla 3.4: Gestión del proyecto. Capacidades y conocimientos del administrador de sistemas junior.

Administrador de Sistemas Junior	**Experiencia mínima (años)**	**Cumplimiento**
Gestión de proyectos	No aplica	No aplica
Administración de sistemas operativos Linux/Unix	1	Obligatorio
Certificación LPIC-1	–	Aconsejable
Administración de redes	1	Obligatorio
Automatización / Scripting	1	Aconsejable
Herramienta de administración Ansible	No aplica	No aplica
Despliegue de aplicaciones con Docker	1	Aconsejable
Gestión de Bases de Datos	1	Aconsejable
Conceptos avanzados de hardware de servidores	No aplica	No aplica
Certificados PKI	No aplica	No aplica
Ciberseguridad	1	Aconsejable
Certificación CEH	–	Aconsejable

Tabla 3.5: Gestión del proyecto. Capacidades y conocimientos del administrador de sistemas junior.

3.2.4 Titulación y formación específica

Además de experiencia y conocimientos, el personal del equipo de trabajo debe tener una formación que se relaciona en la siguiente tabla.

Titulación mínima	**Jefe de Proyecto**	**SysAdmin Junior**	**SysAdmin Senior**
Técnico Superior en materia TIC	No aplica	Obligatorio	Obligatorio
Titulación Universitaria en materia TIC	Obligatorio	No aplica	No aplica

Tabla 3.6: Gestión del proyecto. Titulación y formación específica.

3.3 Presupuesto

Una estimación del presupuesto utilizado para el desarrollo del presente proyecto sería la siguiente:

Material	**Precio**	**Duración**
VPS 1	18,40€/mes	3 meses
VPS 2	46,00€/mes	3 meses
Nombre de dominio	9,99€/año	1 año
Portátil HP Pavilion x360 14-DH0019NS	439€	No aplica
TOTAL: 642.19€		

Tabla 3.7: Gestión del proyecto. Tabla presupuestaria.

3.4 Diagrama de Gantt

A continuación, se muestra el diagrama de Gantt que expone el tiempo estimado que será necesario dedicar a las distintas partes del proyecto.

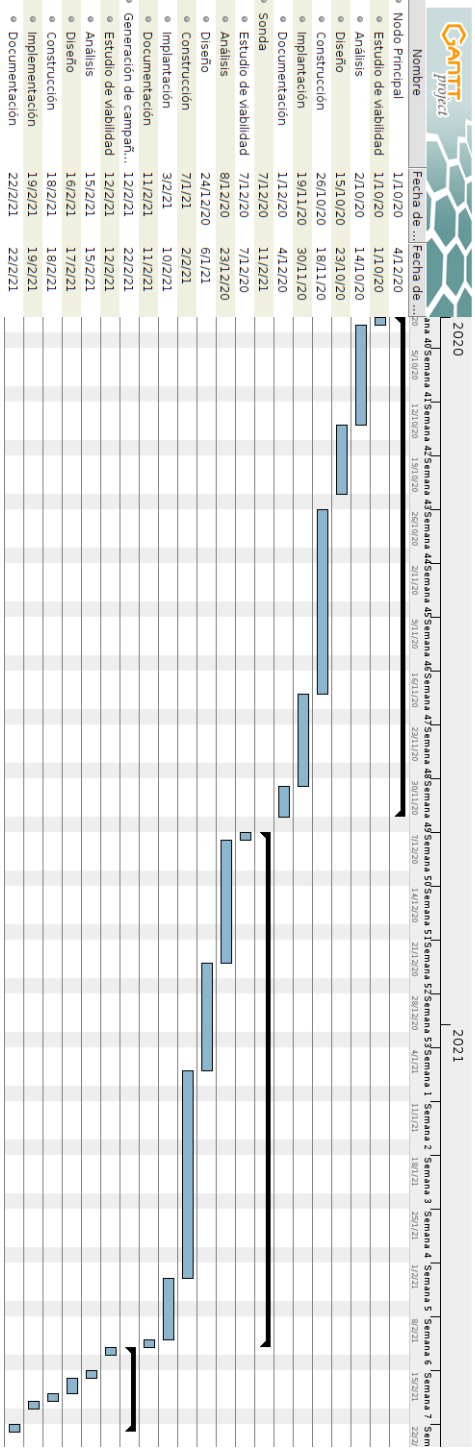


Figura 3.1: Gestión del proyecto. Diagrama de Gantt.

3.5 Herramientas utilizadas

Las herramientas más relevantes utilizadas durante el desarrollo del proyecto son las siguientes:

- **T-Pot**: sistema que combina herramientas y honeypots para estudiar los movimientos de posibles atacantes.
- **Elasticsearch**: base de datos NoSQL basada en Lucene[1] para la búsqueda de texto y documentos.
- **Logstash**: herramienta capaz de administrar logs. Permite recoger datos, analizarlos y distribuirlos.
- **Kibana**: herramienta de visualización de contenido indexado en la base de datos Elasticsearch.
- **Nginx**: servidor web capaz de funcionar como proxy inverso.
- **Docker**: herramienta capaz de desplegar aplicaciones dentro de contenedores de software.
- **Docker-compose**: herramienta capaz de simplificar el uso de Docker, ya que es capaz de realizar multitud de tareas con contenedores desde un solo fichero ejecutado con un solo comando.
- **Ansible**: herramienta de administración de dispositivos remotos de manera automática.
- **LaTeX**: herramienta para la generación de documentos de alta calidad tipográfica.
- **Remmina**: herramienta que permite realizar conexiones remotas mediante protocolos como Remote Desktop Protocol (RDP), SSH o Virtual Network Computing (VNC).
- **Certbot**: herramienta capaz de obtener certificados SSL/TLS gratuitos de Les's Encrypt.
- **Github**: plataforma que permite alojar proyectos para el controlado de versiones mediante Git.
- **Jabref**: herramienta que permite administrar y gestionar bibliografías con el formato BibTex.
- **Tmux**: permite multiplexar un terminal de Linux, permitiendo utilizar varias secciones en una misma interfaz visual.

[1] API de código abierto para recuperación de información.

- **Draw.io**: herramienta para crear y editar todo tipo de diagramas.
- **Ganttproject**: herramienta para crear diagramas de Gantt.

4 Fundamentos

Ya que existen multitud de trabajos dedicados exclusivamente al marco teórico que aborda la naturaleza de este proyecto, se explicarán brevemente los conceptos necesarios para el entendimiento del mismo, de los cuales, los más importantes son:

- Defensa pasiva.
- Defensa activa.
- Comprender el concepto de ”inteligencia” en este proyecto.
- Saber qué es y para qué sirve un honeypot y los tipos que existen.
- Conocer el funcionamiento de T-Pot.

4.1 Defensa pasiva, defensa activa e inteligencia

Conocer los conceptos de defensa pasiva, defensa activa e inteligencia y sus diferencias, es fundamental para entender el presente proyecto.

La **inteligencia** es el resultado de un proceso de recolección de datos que generen información para que una persona con el perfil de **analista de seguridad** pueda crear e interpretar hipótesis.

Una **defensa pasiva**, engloba todos los dispositivos encargados de proteger los sistemas conectados a Internet como pueden ser antivirus, cortafuegos, IPS, IDS, etc. Sin embargo, una **defensa activa** es un proceso de acciones de mejora continua basado en el estudio de cuándo, dónde y cómo se producirá un ataque.

El modelo de defensa activa basado en T-Pot, permitirá, mediante sus herramientas y honeypots, realizar un análisis de amenazas estudiando las **Tácticas**, **Técnicas** y **Procedimientos** de los atacantes (*véase* `https://attack.mitre.org/matrices/enterprise/`), y crear inteligencia que pueda mejorar las capacidades de los dispositivos de defensa pasiva.

Con esta estrategia, se podrá obtener inteligencia detallada e individualizada de los ataques recibidos en la sonda, se podrá mejorar la protección de los activos y de las organizaciones con misma tecnología que la expuesta en la sonda e integrar la información obtenida en tiempo real con herramientas Security Information and Event Management (SIEM), entre otros usos.

4.2 Honeypots

Un honeypot es equipo informático vulnerable, posicionado estratégicamente en algún lugar de Internet o en una red corporativa, para ser comprometido por un atacante informático. Las actividades que un atacante realiza una vez vulnerado, son monitorizadas y registradas con la intención de estudiar al adversario.

Los recursos utilizados para desplegar un honeypot podrían ser un equipo real, una máquina virtual, la simulación de un servicio, etc.

Un honeypot se expone para ser atacado con la finalidad de poder distraer a los posibles adversarios, obtener información tanto de los ataques como de los atacantes, obtener estadísticas sobre tendencias de ataques, detectar posibles campañas maliciosas dirigidas a una vulnerabilidad concreta, aprender sobre nuevas técnicas, tácticas y procedimientos, detectar vulnerabilidades desconocidas, capturar muestras de malware o para complementar soluciones de defensa pasiva.

4.2.1 Tipos de honeypots

Según la localización de despliegue de un honeypot (Internet o una red corporativa), se pueden distinguir dos tipos de honeypots:

- Honeypots de producción: se colocan dentro de una red interna simulando servicios de la red corporativa. Los ataques recibidos podrían provenir de equipos infectados dentro de la red o de usuarios legítimos "curiosos".

- Honeypots de investigación: se colocan en Internet con la intención de aprender lo máximo posible del adversario.

Una consideración importante a la hora de desplegar un honeypot de investigación es que nunca debería hacerse en un sistema en producción o en un sistema desde el cual se pueda acceder a otros equipos en producción.

4.2.2 Clasificación de honeypots

Los honeypots están clasificados según su interacción: **baja**, **media** o **alta**.

La interacción hace referencia a la cantidad de acciones que un atacante puede realizar en un honeypot. Una interacción mayor implica un mayor riesgo de seguridad, pero la cantidad de información obtenida será más extensa.

A continuación, se muestran las características principales de los honeypots según su interacción.

Baja interacción	Media interacción	Alta interacción
Simulan servicios/sistemas	Simulan servicios/sistemas	Simulan sistemas reales
Actividad limitada	Mayor interacción	Interacción alta
Riesgos de seguridad mínimos	Aumentan los riesgos de seguridad	Riesgos de seguridad altos
Poca información	Mayor cantidad de información	Gran cantidad de información
Fáciles de desplegar	Fáciles de desplegar	Difíciles de desplegar
Fáciles de detectar	Más difícles de detectar	Muy dificil de detectar

Tabla 4.1: Fundamentos. Interacción de los honeypots

4.3 T-Pot

T-Pot se basa en una distribución Debian (estable), preparada para ejecutar múltiples programas y herramientas de honeypot (*véase "Anexo I. Programas y herramientas de T-Pot"*).

La gran ventaja de T-Pot es que consigue ejecutar los distintos honeypots y herramientas mediante contenedores Dockers en un mismo equipo, logrando separar cada honeypot mediante distintos segmentos de red con una sola tarjeta de red. Un esquema del diseño de su arquitectura se puede ver en la siguiente imagen.

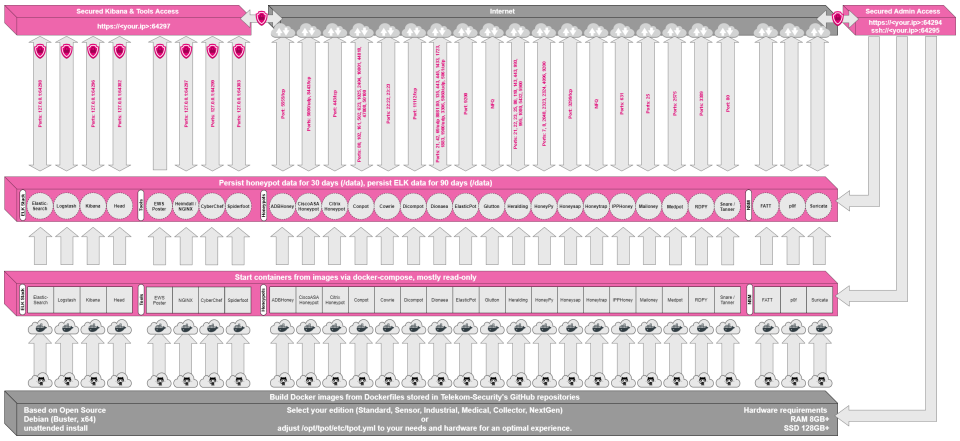


Figura 4.1: Fundamentos. Arquitectura de T-Pot.

4.3.1 Tipos de instalación

T-pot tiene tres maneras diferentes para realizar su instalación:

- Desde una imagen ISO (**tipo iso**), que puede ser descargada del repositorio de GitHub de Deutsche Telekom `https://github.com/telekom-security/tpotce/releases/download/20.06.2/tpot.iso`, donde el sistema operativo viene ya preparado para ser instalado (*véase "Anexo II. Instalación de T-Pot mediante una imagen ISO"*) mediante un USB Bootable, una máquina virtual o en un VPS que lo permita.

- Desde un sistema operativo ya instalado (**tipo user**)con Debian 10. Este modo de instalación fue creado teniendo en cuenta que muchos proveedores de VPS no permiten cargar imágenes ISO para la instalación del sistema operativo.

- Desde un sistema operativo ya instalado y con una configuración previa (**tipo auto**), muy similar al tipo de instalación "user", pero además, se podrá añadir un fichero de configuraciones del sistema para luego poder automatizar despliegues con herramientas como Ansible o Terraform.

4.3.2 Modos de instalación

T-pot tiene distintas ediciones de instalación diseñadas por sus creadores según los objetivos que se quieran estudiar: **Standar**, **Sensor**, **Industrial**, **Collector**, **NextGen** y **Medical**.

Cada una de estas ediciones de instalación tiene sus propias herramientas y honeypots. Se pueden ver en el análisis realizado durante el estudio de viabilidad de este proyecto en las tablas "2.2 Estudio de viabilidad. Herramientas según el modo de instalación de T-Pot" y "2.3 Estudio de viabilidad. Honeypots según el modo de instalación de T-Pot".

Además, al ser una herramienta de código abierto, se podrán crear ediciones propias.

4.3.3 Actualizaciones y mantenimiento

T-Pot está diseñado para funcionar desde el momento en que es instalado sin apenas tener que realizar ningún tipo de mantenimiento. En caso de que algún componente del sistema provocase algún fallo, bastaría con reiniciar el sistema.

En cuanto a las actualizaciones, existe un script en T-Pot que se encarga de poner al día los componentes del sistema. Este script, descarga de la rama principal de GitHub las últimas versiones de los ficheros de configuración, herramientas y honeypots. Con este sistema de actualización se borrará todas las posibles modificaciones realizadas hasta el momento, un aspecto importante a tener en cuenta. Siendo aconsejable antes de realizar una actualización, hacer una copa de seguridad.

4.3.4 T-Pot como ejemplo de uso

Una vez se ha desplegado una sonda de T-Pot, se puede tener acceso a varios componentes.

Para tener acceso al sistema remotamente, T-Pot permite realizar una conexión **SSH** al puerto **64295** de manera predeterminada con el usuario **tsec** y la contraseña introducida durante su instalación. Para evitar ataques de fuerza bruta al protocolo SSH, trae configurado la herramienta **fail2ban**.

También, sería posible acceder a un terminal del sistema, entre otras utilidades, desde la interfaz web que ofrece la herramienta **Cockpit** en el puerto **64294**.

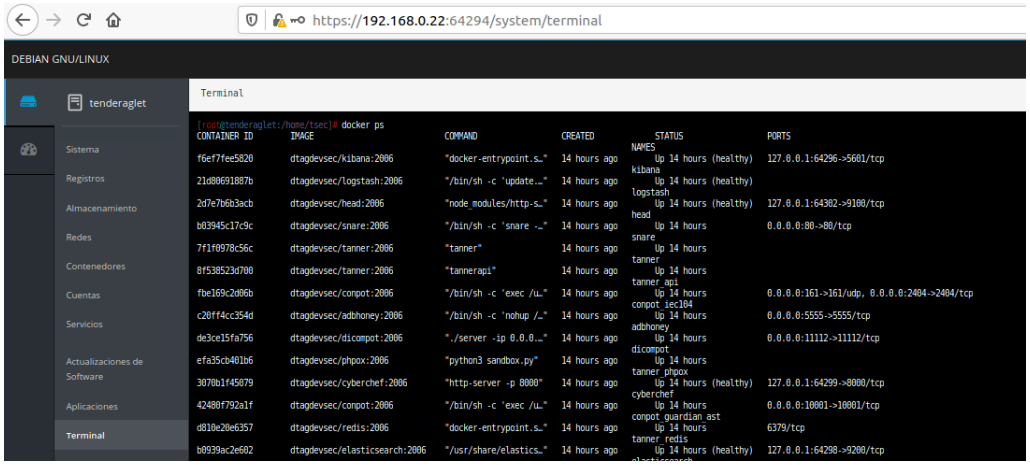


Figura 4.2: Fundamentos. Terminal del sistema mediante Cockpit.

Por otro lado, T-Pot ofrece su interfaz web principal en el puerto **64297**, donde se pueden ver todas las herramientas disponibles ingresando previamente el usuario y contraseña introducidos durante la instalación.

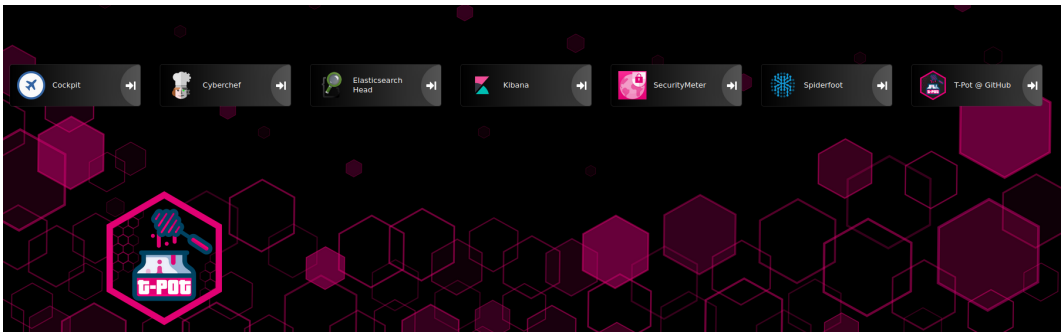


Figura 4.3: Fundamentos. Interfaz web de T-Pot.

Para el tratamiento de los logs obtenidos en cada honeypot, T-Pot se apoya en **Kibana**, permitiendo visualizar y analizar los datos a través de la creación de gráficos de barras, gráficos circulares, tablas, histogramas y mapas en tiempo real. Un ejemplo de ello se puede apreciar en la siguiente imagen, en la cual se muestra el número de ataques recibidos en los distintos honeypots, la geolocalización de los actores que están atacando cada honeypot, los puertos que se están atacando y muchos otros datos de interés.

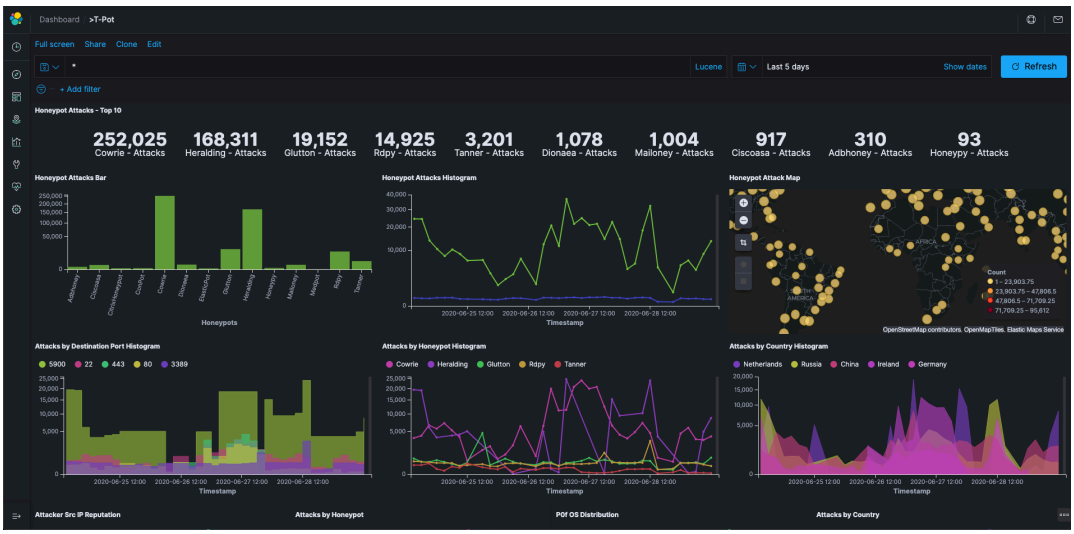


Figura 4.4: Fundamentos. Dashboard de Kibana configurado para T-Pot.

No obstante, para consultar una información más específica sobre T-Pot, se recomienda ver el repositorio oficial de Deutsche Telekom en `https://github.com/telekom-security/tpotce`.

5 Despliegue de la infraestructura

5.1 Nodo principal

5.1.1 Preparación del sistema

Antes de empezar a trabajar con el VPS, se han realizado unas tareas preliminares que, aunque son obvias, son de igual importancia.

La primera de ellas, ha sido contratar un VPS acorde a los requerimientos hardware mencionados en el capítulo Estudio de viabilidad sección "2.2.1.2 Nodo principal". Los pasos a destacar durante la contratación, han sido la elección del sistema operativo, en este caso **Ubuntu Server 18.04 LTS** y la petición de la contraseña de acceso para el usuario "root". Una vez terminado el proceso de contratación, se ha podido acceder al panel de administración del servidor y ver su dirección IP, en este caso **23.254.225.240**.

La segunda tarea preliminar ha sido contratar el dominio **helioslove.com** y crear los siguientes registros DNS para que apunten a la dirección IP del VPS contratado. En este caso:

Tipo	Nombre de dominio	Destino	TTL (segundos)
A	helioslove.com	23.254.225.240	3600
CNAME	god.helioslove.com	23.254.225.240	3600
CNAME	sensors.helioslove.com	23.254.225.240	3600
NS	helioslove.com	dns101.ovh.net	3600
NS	helioslove.com	ns101.ovh.net	3600

Tabla 5.1: Despliegue de la infraestructura. Registros DNS del dominio

5.1.2 Configuración segura del VPS

Una vez contratado el VPS, será importante realizar una serie de configuraciones de seguridad básicas para evitar accesos no autorizados al sistema.

5.1.2.1 Gestión de usuarios

Se ha limitado el usuario "**root**" para aquellas tareas imprescindibles, por ello, un usuario denominado "**helios**" con los privilegios necesarios y con el que se realizarán el máximo de tareas administrativas.

```
root@VPS:~# adduser helios
Adding user `helios' ...
Adding new group `helios' (1000) ...
Adding new user `helios' (1000) with group `helios' ...
Creating home directory `/home/helios' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for helios
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
```

```
root@VPS:~# adduser helios sudo
```

5.1.2.2 Generación de claves SSH

El uso de claves SSH es un método más robusto de autenticación que el uso de contraseñas. Este sistema de autenticación, se basa en un par de claves (pública/privada) generadas a partir de un determinado algoritmo. La clave pública, se almacena en el servidor al cual se quiere acceder por SSH, mientras que la clave privada, protegida con un *passphrase*[1], debe ser almacenada en un lugar seguro por el propietario, ya que solo con ésta se podrá desbloquear el acceso al servidor. Así, un atacante que quiera usurpar la identidad de un usuario necesitará la clave privada y su *passphrase*.

[1] Frase de contraseña.

El propietario de la clave privada, necesitará un cliente SSH que, generalmente, viene instalado por defecto en las distribuciones Linux, como es el caso.

Para generar el par de claves pública/privada, se ha hecho uso de la herramienta **ssh-keygen** proporcionada por el paquete **openSSH-client**, el algoritmo **RSA** y una longitud de **4096** bits.

```
alumno@tfg:~$ ssh-keygen -b 4096 -t rsa
```

Las claves generadas, **id_rsa.pub** (clave pública) y **id_rsa** (clave privada), se guardan por defecto en el directorio oculto ∼/.ssh/.

En cuanto al *passphrase*, se recomienda utilizar una contraseña compleja ya que se basa en un cifrado simétrico.

Una vez se han generado las claves SSH correspondientes en el cliente, se ha copiado el contenido de la clave pública en el fichero **/home/helios/.ssh/authorized_keys**[2] del servidor con el siguiente comando:

```
alumno@tfg:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub helios@23.254.225.240
```

A partir de este momento, se accederá al servidor con la clave privada introduciendo el siguiente comando en un terminal:

```
alumno@tfg:~$ ssh helios@23.254.225.240
```

5.1.2.3 Configuración segura del protocolo SSH

Una vez creado un nuevo usuario, se podrá deshabilitar el acceso al VPS por SSH mediante el usuario *root*. Para ello se ha modificado en el fichero **/etc/ssh/sshd.conf** la directiva "**PermitRootLogin yes**" susituyéndola por "**PermitRootLogin no**".

Por otro lado, en el mismo fichero de configuración se ha deshabilitado el uso de contraseñas como método de autenticación estableciendo la directiva "**PasswordAuthentication no**".

5.1.2.4 Configuración del cortafuegos UFW

Uncomplicated Firewall (UFW) es un cortafuegos que facilita la creación de reglas IPTables. El cortafuegos viene instalado por defecto en las distribuciones Linux en modo deshabilitado. Antes de activarlo, es necesario configurar las reglas de acceso al VPS ya que

[2]Comprobar si la carpeta .ssh y el fichero authorized_keys existen, si no, crearlos manualmente.

de no hacerlo se perdería la conexión con él debido a que su configuración por defecto es permitir todo el tráfico saliente y denegar todo el entrante.

Como el nodo principal será un servidor web, se deben permitir las conexiones entrantes http y https. Además, para la administración del VPS se deben permitir también las conexiones entrantes SSH. Para ello, se han creado las siguientes reglas en el cortafuegos escribiendo en un terminal:

```
helios@VPS:~# sudo ufw allow ssh
helios@VPS:~# sudo ufw allow https
helios@VPS:~# sudo ufw allow http
```

Una vez creadas las reglas, se ha activado el cortafuegos y revisado su estado.

```
helios@VPS:~# sudo ufw enable
```

```
helios@VPS:~# sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To Action From
-- ------ ----
22/tcp ALLOW IN Anywhere
80/tcp ALLOW IN Anywhere
443/tcp ALLOW IN Anywhere
22/tcp (v6) ALLOW IN Anywhere (v6)
80/tcp (v6) ALLOW IN Anywhere (v6)
443/tcp (v6) ALLOW IN Anywhere (v6)
```

5.1.3 Puesta en marcha

Una vez realizadas las tareas básicas de seguridad, se procederá a poner en marcha los servicios descritos en la sección 2.3 Solución final referentes al nodo principal.

5.1.3.1 Uso de contenedores Docker

Siguiendo la linea de Deutsche Telekom con T-Pot, se ha decidido implementar todos los servicios del nodo principal mediante contenedores **Docker**. De esta forma, se facilitará el despliegue de otros nodos principales en el mismo entorno, o en otros, en el futuro y se logrará una mayor productividad entre el entorno de pruebas y los entornos de preproducción y producción, ya que al estar en un contenedor, éstos son idénticos.

Además, se ha utilizado la herramienta **Docker Compose** para orquestar de manera centralizada los contenedores.

Los pasos a realizar para la instalación de Docker y Docker Compose en el nodo principal se pueden seguir en el "*Anexo III. Instalación de Docker y Docker Compose*".

Existen multitud de imágenes en `https://hub.docker.com/` de todos los servicios a desplegar en el nodo principal, **Nginx**, **Elasticsearch** y **Kibana**, preparadas para descargar y empezar a utilizar en un instante levantando un contenedor. En el presento proyecto, se han usado las dos imágenes oficiales de Elastic en su versión más reciente **Elasticsearch:7.10.1**[3] y **Kibana:7.10.1**.
Para descargarlas, se han introducido los siguientes comando en un terminal:

```
helios@VPS:~# docker pull elasticsearch:7.10.1
helios@VPS:~# docker pull kibana:7.10.1
```

También existen multitud de imágenes preparadas con el servicio web de Nginx, desplegadas en variedad de sistemas operativos como base, pero se ha decidido desarrollar una imagen Docker que, además de cumplir el requisito de proxy inverso, cumpla con el requisito de cifrar los canales de comunicación establecidos para acceder a los servicios de Elastic.

5.1.3.2 Desarrollo de una imagen Docker para el servidor web

El propósito de desarrollar esta imagen Docker, es poder desplegar el servicio web automáticamente con sus certificados SSL/TLS de **Let's Encrypt** instalados. Para realizar esto, se ha hecho uso de **Certbot**, una herramienta de software de código abierto y gratuita para gestionar automáticamente certificados de Let's Encrypt.

Para trabajar de manera organizada, se ha creado el directorio "**despliegue**" en la carpeta personal del usuario helios.

```
helios@VPS:~# mkdir /home/helios/despliegue
```

[3]Para levantar un contenedor con esta imagen habrá que asignar la memoria virtual del sistema operativo anfitrión en 262144 con el comando sysctl -w vm.max_map_count=262144.

A continuación, se ha creado un fichero con el nombre de "**Dockerfile**".

```
helios@VPS:~# touch /home/helios/despliegue/Dockerfile
```

La imagen a desarrollar, utilizará como base la imagen oficial **ubuntu:18.04**, por lo que habrá que descargarla. Para ello, se ha introducido el siguiente comando en un terminal:

```
helios@VPS:~# docker pull ubuntu:18.04
```

La instalación básica de Certbot requiere tener configurada la zona horaria en el sistema operativo, en caso de no tenerla, se podría configurar mediante un asistente durante su instalación. Esto es un problema para el desarrollo de la imagen Docker, ya que ésta no sabe interactuar con el sistema cuando un dato es requerido, hay que darle las instrucciones de tal forma que esté todo contemplado. Para solucionarlo, se ha configurado la zona horaria en la imagen:

Código 5.1: Dockerfile. Configuración de la zona horaria

```
ENV TZ=Europe/Madrid
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/↩
    ↪ timezone
```

Para que al lanzar el contenedor tenga instalado Certbot y Nginx, se han añadido las siguiente lineas en la imagen Docker:

Código 5.2: Dockerfile. Instalación de Nginx y Certbot

```
RUN apt update && apt install -y nginx certbot
```

Para configurar el servidor web automáticamente se han borrado los ficheros de configuración necesarios de Nginx y se han añadido unos personalizados a la imagen ubicados en el directorio actual de trabajo (*véase* "Anexo IV. Archivos de configuración de Nginx personalizados") . También, se ha creado un directorio donde se servirá la página web principal y, como buena práctica, se han creado enlaces simbólicos para facilitar futuros cambios.

Código 5.3: Dockerfile. Configuración de Nginx

```
RUN rm /etc/nginx/sites-available/default
```

```
RUN rm /etc/nginx/sites-enabled/default
COPY helioslove /etc/nginx/sites-available/
COPY god.helioslove /etc/nginx/sites-available/
COPY sensors.helioslove /etc/nginx/sites-available/
RUN mkdir /var/www/helioslove.com
RUN ln -s /etc/nginx/sites-available/helioslove /etc/nginx/sites-↩
    ↪ enabled/
RUN ln -s /etc/nginx/sites-available/god.helioslove /etc/nginx/sites-↩
    ↪ enabled/
RUN ln -s /etc/nginx/sites-available/sensors.helioslove /etc/nginx/↩
    ↪ sites-enabled/
```

Como la caducidad de los certificados de Let's Encrypt es de noventa días, se ha automatizado el proceso de renovado copiando un script en **Bash** llamado **renew.sh** (*véase* "*Anexo V. Scripts necesarios para la imagen Docker*") en la imagen y añadiendo una tarea programada que se encargue de ejecutarlo una vez al mes.

Código 5.4: Dockerfile. Renovación automática de certificados Let's Encrypt

```
COPY renew.sh /renew.sh
RUN chmod +x /renew.sh
RUN echo "0 0 1 * * root /renew.sh" >> /etc/crontab
```

Por último, se ha copiado otro script en **Bash** llamado **run.sh** (*véase* "Anexo V. Scripts necesarios para la imagen Docker") en la imagen que se encarga de levantar el servicio de Nginx y mantener activo el contenedor.

Código 5.5: Dockerfile. Activación del servicio de Nginx

```
COPY run.sh /run.sh
RUN chmod +x /run.sh
CMD sh /run.sh
```

Se puede ver el fichero completo con la imagen Docker desarrollada en el "*Anexo VI. Imagen Docker desarrollada*".

5.1.3.3 Orquestación con Docker Compose

Con la herramienta **Docker Compose** se podrán poner en marcha los tres servicios y que interactúen entre sí de una forma organizada y fácil.

En primer lugar, será importante pensar en el diseño que se necesitará para que todo funcione correctamente. De manera muy resumida, las partes principales a tener en cuenta de los ficheros docker-compose son la **versión**, los **servicios**, los **volúmenes** y las **redes**.

Se recomienda utilizar la versión más reciente, en este caso la versión "3.7".

En cuanto a los servicios, serán Elasticsearch y Kibana con sus respectivas imágenes oficiales y Nginx con la imagen Docker desarrollada en la sección anterior. Se habilitarán los puertos por defecto para cada uno de ellos, dejando el mismo tanta para el contenedor como para el equipo anfitrión:

- Elasticsearch: 9200.
- Kibana: 5601.
- Nginx: 80 y 443.

Además, se crearán las variables de entorno básicas que necesita el contenedor de Elasticsearch, de las cuales la más relevante será la asignación de hasta 4GB de memoria Ram para la máquina virtual de Java. También se activará y configurará la extensión de seguridad **X-Pack** tanto para Elasticsearch como para Kibana.

El apartado de volúmenes es uno de los más importantes. Un diseó erróneo, podría conllevar pérdidas de datos importantes, o incluso impedir poner en marcha un contenedor.
Para el servicio de Elasticsearch, será necesario almacenar los datos de la base de datos de manera persistente, por lo que se creará el volumen:

- **elastic-data**, que se mapeará en el directorio **/usr/share/elasticsearch/data** del contenedor.

Para el servicio de Nginx, se crearán los volúmenes:

- nginx-web, que se mapeará en /var/www/helioslove.com
- nginx-certs, que se mapeará en /etc/letsencrypt/archive
- nginx-certs-enlaces, que se mapeará en /etc/letsencrypt/live

Así, se conseguirá modificar la pagina web del dominio principal desde el equipo anfitrión y no desde el contenedor, pudiéndose guardar guardar los certificados de Let's Encrypt como copia de seguridad.

Por último, los tres servicios estarán dentro de la misma red y habrá que darles IPs estáticas para que el proxy inverso pueda redirigir correctamente el tráfico. La red creada se llamará "**app_net**" y tendrá una dirección de red **10.10.10.0/24**. Las IPs para cada contenedor serán las siguientes:

- Elasticsearch: 10.10.10.2.
- Kibana: 10.10.10.3.
- Nginx: 10.10.10.4.
- Puerta de enlace: por defecto, 10.10.10.1.

Para aplicar todas estas configuraciones en los distintos contenedores se ha creado el fichero **docker-compose.yml** en el directorio de trabajo actual y editado (*véase su contenido en el "Anexo VII. Fichero de configuración docker-compose"*).

```
helios@VPS:~# touch /home/helios/despliegue/docker-compose.yml
```

5.1.3.4 Creación de un servicio Systemd

La motivación de este apartado es poder gestionar el estado de los contenedores desde un servicio. **Systemd** es un gestor de servicios de Linux que permite acelerar el arranque de servicios, averiguar y gestionar el estado exacto de cada proceso que forma parte de un servicio y tener el control total de éstos. Para ello, se ha creado el fichero **/lib/systemd/system/docker-compose.service**:

```
helios@VPS:~# sudo touch /lib/systemd/system/docker-compose.service
```

Una vez creado, se ha añadido, con permisos de administrador, el siguiente código:

Código 5.6: Consola. Código del servicio docker-compose

```
[Unit]
Description=nodo_principal
Requires=docker.service
After=docker.service

[Service]
Restart=always
RestartSec=5
TimeoutSec=infinity

WorkingDirectory=/home/helios/despliegue/
```

```
# Remove old containers, images and volumes
ExecStartPre=/usr/local/bin/docker-compose down -v
ExecStartPre=/usr/local/bin/docker-compose rm -fv
ExecStartPre=-/bin/bash -c 'docker volume rm $(docker volume ls -q)'
ExecStartPre=-/bin/bash -c 'docker network rm $(docker network ls -q)'
ExecStartPre=-/bin/bash -c 'docker rm -v $(docker ps -aq)'
ExecStartPre=-/bin/bash -c 'docker rmi $(docker images | grep "<none>" ↩
    ↪ | awk \'{print $3}\')'

# Compose up
ExecStart=/usr/local/bin/docker-compose up

# Compose down, remove containers and volumes
ExecStop=/usr/local/bin/docker-compose down -v

[Install]
WantedBy=multi-user.target
```

Finalmente, se ha habilitado para que el servicio se ejecute con el arranque del sistema operativo:

```
helios@VPS:~# sudo systemctl enable docker-compose
```

5.1.3.5 Configuración del dashboard de Kibana

Kibana ofrece una gran cantidad de funciones para representar gráficamente los datos indexados, en este caso, en Elasticsearch. Sin embargo, se ha aprovechado el trabajo realizado por Deutsche Telekom exportando todos los objetos que forman parte del dashboard de T-Pot e importándolos en el nodo principal. Para ello, se han seguido los siguientes pasos:

1. Descargar el fichero **kibana_export.ndjson.zip** del repositorio oficial de GitHub de T-Pot de `https://github.com/telekom-security/tpotce/blob/master/etc/objects/kibana_export.ndjson.zip`
2. Descomprimir el fichero descargado en el paso anterior.
3. Desde el panel de Kibana del nodo principal hacer click en "**Stack Management**".
4. Hacer click en "**Saved Objects**".

5. Hacer click en "**Import**".

6. Elegir el fichero "**kibana_export.ndjson**" descomprimido en el paso dos y cuando se pregunta si se quiere sobrescribir los objetos en caso de que ya existan contestar "**Sí, sobrescribir todo**".

Se puede seguir el proceso mediante un video de Youtube en `https://youtu.be/67TjuhdySbc`

5.2 Sonda

5.2.1 Personalización de T-Pot

Deutsche Telekom permite personalizar el sistema de T-Pot modificando su código fuente y archivos de configuración. Sin embargo, la documentación para este propósito es escasa, haciendo únicamente mención al estudio del fichero de configuración docker-compose /opt/t-pot/etc/**tpot.yml** y al servicio /etc/systemd/system/**tpot.service**.

El archivo **tpot.yml**, es un enlace simbólico apuntando al archivo de configuración correspondiente a la edición de T-Pot que se quiera instalar (véase "4.3.2 Modos de instalación").

Por su naturaleza, no se encuentra en el repositorio de GitHub de Deutsche Telekom. Para encontrarlo, se han tenido que analizar los directorios de una sonda instalada mediante una imagen ISO (*véase "7.4 Anexo II. Instalación de T-Pot mediante una imagen ISO*") en su edición **Standard**, puesta en marcha para su investigación (*véase "5.2.5 Investigación para la instalación de T-Pot en un VPS*") y, por ende, el archivo de configuración a estudiar es /etc/compose/**standard.yml** (*véase* `https://github.com/telekom-security/tpotce/blob/master/etc/compose/standard.yml`).

Por otro lado, el servicio de T-Pot **tpot.service** (*véase "Anexo X. Servicio de T-Pot*") es el encargado de arrancar o detener los contenedores Docker definidos en el archivo de configuración al que apunta el enlace simbólico tpot.yml, en la sonda investigada, /etc/compose/**standard.yml**.

Una vez claros estos conceptos, se podría parar el servicio tpot.service, realizar modificaciones en tpot.yml, por ejemplo borrar el contenedor completo del honeypot ciscoasa, y volver a arrancar el servicio tpot.service, que arrancaría todos los contenedores excepto el del honeypot ciscoasa.

Realizar este proceso en una sonda cuesta poco tiempo y esfuerzo, pero para un número mayor de éstas, sería totalmente inviable. Este hecho, obliga a buscar una alternativa que sea capaz de desplegar tantas sondas como se deseen con una configuración personalizada (*vease "5.2.1.1 Diseño a medida*").

5.2.1.1 Diseño a medida

Tras la investigación del funcionamiento de T-Pot realizada en la sección "5.2.5 Investigación para la instalación de T-Pot en un VPS", se ha decidido modificar el contenido del docker-compose "**standard.yml**" y el script **install.sh**.

Cambios en el docker-compose standard.yml

Todos los contenedores que aparezcan en este fichero serán los encargados de arrancar el sistema, gracias al servicio creado con el nombre de **tpot.service**. La versión **Standard** de T-Pot es la más completa, por ello se ha tomado como referencia.

Se han borrado los contenedores que no son necesarios:

- Cyberchef.
- Elasticsearch.
- Kibana.
- Elasticsearch-head
- Ewsposter
- Nginx
- Spiderfoot

Se han modificado los parámetros del contenedor **Logstash** para evitar errores de arranque del contenedor, ya que anteriormente dependía de Elasticsearch (eliminado en el paso anterior) (*véase "Anexo XI. Configuración para el contenedor de Logstash*")

Cambios en el script install.sh

Se han eliminado todas las tareas y funciones que no son necesarias para el despliegue de este proyecto:

- Tareas relacionadas con los contenedores desactivados en la configuración del docker-compose standard.yml, así como las variables y funciones correspondientes.
- Módulos relacionados con el tipo de instalación "**iso**" y "**auto**" (*véase "4.3.1 Tipos de instalación*").

Se puede ver el código resultante en el "*Anexo IX. Script Install.sh personalizado*" o en GitHub en `https://github.com/0l4y4/helioslove/blob/main/install.sh`. El diagrama de flujo correspondiente al script es el siguiente:

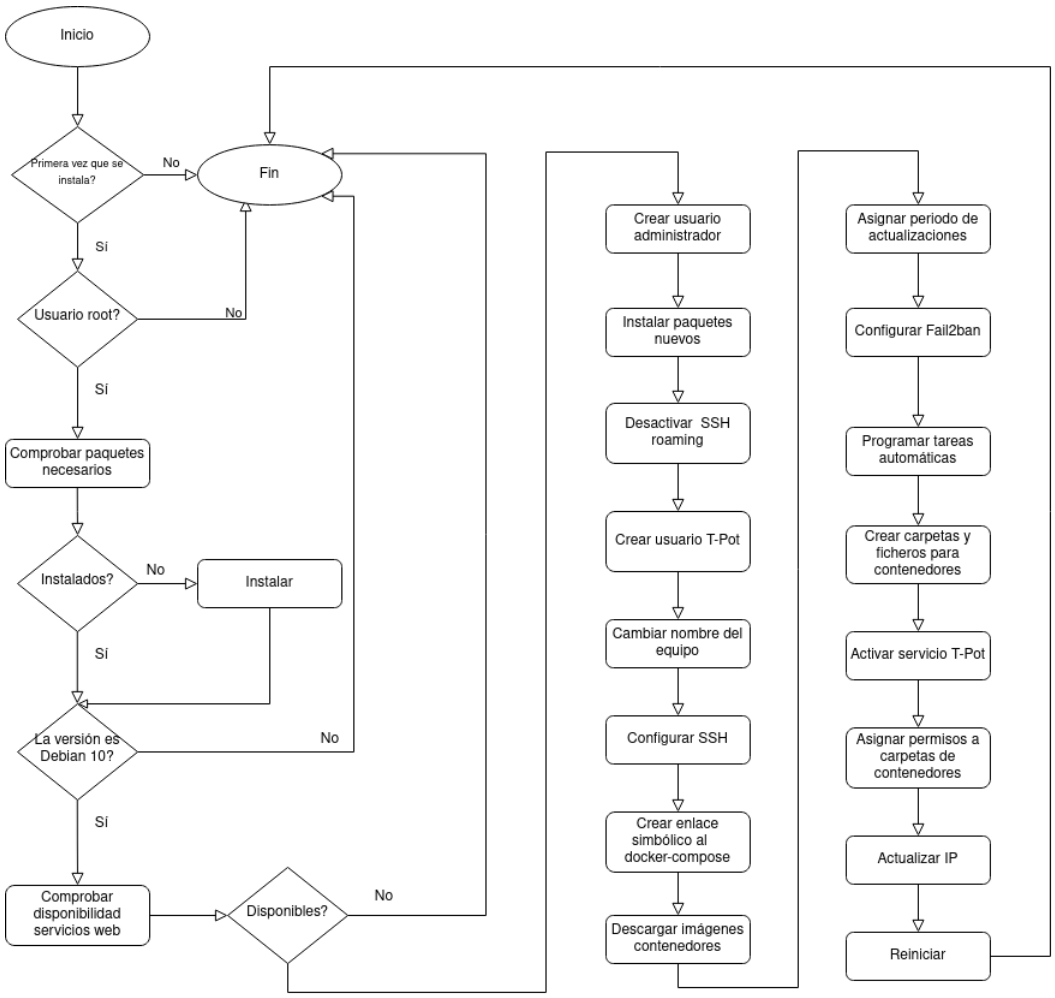


Figura 5.1: Despliegue de la infraestructura. Diagrama de flujo del script install.sh modificado

5.2.2 Preparación del sistema

Para poder empezar con la instalación de la sonda, se ha contratado otro VPS acorde a los requerimientos hardware mencionados en el estudio de viabilidad sección "2.2.1.1 Sonda", con el sistema operativo **Debian 10**.

5.2.3 Instalación de la sonda personalizada en un VPS

Para realizar esta instalación de manera satisfactoria, se ha tenido que abrir una linea de investigación (*véase* "5.2.5 Investigación para la instalación de T-Pot en un VPS"), debido a

que la documentación ofrecida por Deutsche Telekom era insuficiente, consiguiendo instalar la versión original de T-Pot en un VPS siguiendo los pasos documentados en el "Anexo XII. Instalación de T-Pot en un VPS".

Como el prototipo sólo está formado por una sonda, se podrían realizar las modificaciones manualmente una vez desplegada la versión original de T-Pot, pero si durante una campaña de inteligencia el número de sondas requerido es grande, realizar todas las modificaciones después de desplegar cada una de ellas manualmente sería inviable, ya que consumiría demasiado tiempo e incluso podría dar lugar a errores. Dada esta situación, se ha creado un script con el nombre de **helios.sh** para desplegar las sondas con las modificaciones necesarias y llegar a la solución deseada (véase "*2.3 Solución elegida*"). El script realiza las siguientes tareas:

- Instala los paquetes necesarios tomando como modelo los obtenidos de la semilla de configuración añadida en la imagen ISO de T-Pot.
- Aplica la configuración correcta en Logstash.
- Levanta los contenedores correspondientes.
- Descarga los ficheros necesarios de GitHub.
- Ejecuta el script personalizado install.sh.

El resultado final del script se puede ver en el "Anexo VIII. Script para el despliegue automatizado de sondas".

Una vez creado el script de automatización, se enumeran los pasos a seguir para el despliegue de una sonda de T-Pot personalizada en un VPS con el sistema operativo Debian 10 instalado:

1. Actualizar la lista de paquetes disponibles y sus versiones.

```
root@VPS:~# apt update
```

2. Instalar Git.

```
root@VPS:~# apt install git
```

3. Descargar el repositorio creado para este proyecto.

```
root@VPS:~# git clone https://github.com/0l4y4/helioslove
```

4. Acceder a la carpeta del repositorio descargado.

```
root@VPS:~# cd helioslove
```

5. Ejecutar el script helios.sh.[4].

```
root@VPS:~# ./helios.sh --user_elastic=usuario --pass_elastic=↩
    ↪ contraseña --pass_admin=contraseña
```

Como se puede observar en este último paso, para un correcto funcionamiento del script es necesario añadir los argumentos del usuario de la base de datos Elasticsearch desplegada en el nodo principal, su contraseña, así como la contraseña del usuario administrador local del sistema operativo donde se está ejecutando. Una vez terminada su ejecución, el sistema se reiniciará y se pondrá en ejecución.

5.2.4 Despliegue masivo

Siguiendo la filosofía de minimizar tiempos y esfuerzo, es conveniente tener la capacidad de desplegar varias sondas de forma simultánea. Para ello, se ha utilizado la herramienta **Ansible**.

Ansible, entre otras muchas funcionalidades, permite la ejecución de comandos en remoto mediante el protocolo SSH en varios dispositivos con una sola instrucción.

Para este escenario, se ha hecho uso de un ordenador externo a la infraestructura que actúa como controlador Ansible, en este caso, un ordenador portátil personal con el sistema operativo **Ubuntu 20.04 LTS**.

La instalación de la herramienta es sencilla, ejecutándose el siguiente comando en un terminal:

```
alumno@tfg:~$ sudo apt -y install ansible
```

Para facilitar el despliegue, se ha evitado la comprobación de la huella digital en la primera conexión SSH que se realice a cada sonda, configurando el fichero /etc/ansible/**ansible.cfg** con la siguiente linea:

```
[defaults]
host_key_checking = false
```

[4]Comprobar que el script tiene permisos de ejecución, en caso de que no los tenga dárselos con: chmod +x helios.sh

Ansible trabaja con **inventarios**. Un inventario es un fichero de configuración que contiene los distintos elementos que se quieren administrar. El fichero de configuración por defecto es /etc/ansible/**hosts**.

Dentro del inventario por defecto, se ha creado el grupo **[campaña_tfg]**. El nombre de los grupos se escribe entre corchetes y agrupa todas las direcciones IP o nombres DNS que tenga escritos debajo. Además, Ansible viene configurado por defecto para usar un par de claves SSH (pública/privada) en sus conexiones, sin embargo, se ha forzado para que lo haga mediante usuario y contraseña definiéndolas al lado de cada IP contenida en el inventario. A continuación, se puede ver cómo se ha creado el grupo y se ha añadido la IP de la sonda actual junto con su usuario y contraseña de acceso.

```
alumno@tfg:~$ cat /etc/ansible/hosts
[campaña_tfg]
5.254.118.157 ansible_ssh_user=root ansible_ssh_pass=contraseña
```

Además, se ha instalado el paquete **sshpass**, para evitar errores al usar usuario y contraseña en las conexiones SSH.

```
alumno@tfg:~$ sudo apt -y install sshpass
```

Una vez el servidor Ansible está preparado, se pueden ejecutar comandos directamente sobre las direcciones IP añadidas en el inventario. La sintaxis pare realizarlo es la siguiente:

```
alumno@tfg:~$ ansible nombre_grupo -a "comando"
```

Si la tarea a realizar requiere permisos de administrador será necesario añadir las opciones –**become**[5] y **-K**[6].

Una vez el servidor Ansible está preparado, resulta muy fácil desplegar cuantas sondas se desee, añadiendo las direcciones IP al inventario, el usuario y la contraseña, ejecutando las siguientes instrucciones:

1. Actualizar la lista de paquetes y sus versiones.

   ```
   alumno@tfg:~$ ansible campaña_tfg -a "apt update" --become -K
   ```

2. Instala Git.

[5]Ejecuta el comando con sudo.
[6]Solicita la contraseña de sudo.

```
alumno@tfg:~$ ansible campaña_tfg -a "apt -y install git" --↩
    ↪ become -K
```

3. Descargar el repositorio helioslove de GitHub, creado para este proyecto.

```
alumno@tfg:~$ ansible campaña_tfg -a "git clone https://github↩
    ↪ .com/Ol4y4/helioslove" --become -K
```

4. Ejecutar el script de despliegue de la sonda "helios.sh"[7].

```
alumno@tfg:~$ ansible campaña_tfg -a "sh /root/helioslove/↩
    ↪ helios.sh --user_elastic=usuario --pass_elastic=↩
    ↪ contraseña --pass_admin=contraseña" --become -K
```

Una vez termine la ejecución del script, las sondas se reiniciarán y empezarán a recibir ciberataques.

5.2.5 Investigación para la instalación de T-Pot en un VPS

Abrir una linea de investigación para conocer el funcionamiento de T-Pot y poder instalarlo en un VPS con el sistema operativo Debian 10 implementado por el proveedor, **ha sido una de las claves para llevar a cabo con éxito este proyecto**, ya que instalar las sondas desde una imagen ISO hubiera limitado mucho las opciones de despliegue de éstas, debido a los pocos proveedores de VPS que permiten importar imágenes de sistemas operativos.

Según la documentación de Deutsche Telekom en su repositorio de GitHub (*véase* `https://github.com/telekom-security/tpotce#postinstall`), se han de seguir los siguientes pasos para realizar la instalación en el VPS:

- Descargar el repositorio de GitHub.
- Acceder a la carperta donde se encuentra el ejecutable de instalación.
- Ejecutar el script de instalación y pasarle el parámetro –type=user.

Sin embargo, la ejecución del script falla y, por ende, la instalación no se realiza.

Para solucionar este problema, se ha realizado un análisis de la instalación de T-Pot mediante la imagen ISO proporcionada por los creadores.

[7] Comprobar que el script tiene permisos de ejecución, en caso de que no los tenga dárselos con ansible campaña_tfg -a "chmod +x /root/helioslove/helios.sh" –become -K

En primer lugar, se ha instalado T-Pot en una máquina virtual mediante la imagen ISO (*véase "Anexo II. Instalación de T-Pot mediante una imagen ISO"*) y se ha comprobado que se realiza correctamente.

En segundo lugar, se ha realizado un análisis exhaustivo del contenido de la imagen ISO.

Finalmente, se ha hecho una investigación del funcionamiento de la instalación mediante la imagen ISO.

5.2.5.1 Análisis del contenido de la imagen ISO de T-Pot

Por motivos de seguridad y transparencia, Deutsche Telekom permite, mediante un script programado en Bash llamado **makeiso.sh**, crear la imagen ISO de instalación de T-Pot. Éste script (*véase* `https://github.com/telekom-security/tpotce/blob/master/makeiso.sh`), realiza una serie de tareas clave para conseguir el objetivo, que se mostrarán a continuación.

La primera tarea relevante que realiza el script, es la descarga de la imagen ISO del sistema operativo Debian en su versión ***Netboot***[8].

Una vez descargada la imagen ISO Debian Netbook, el script modifica los siguientes archivos ubicados en la raíz de la imagen ISO:

- **initrd**: sistema de archivos temporal que carga junto con el Kernel para arrancar el sistema operativo. El script añade un fichero semilla llamado **preesed.cfg** en la raiz que el instalador del sistema operativo buscará durante el arranque haciendo posible que la instalación sea automática, ofreciendo un mecanismo para responder a preguntas realizadas durante la instalación sin tener que introducir manualmente las respuestas mientras éste se ejecuta. Además, crea los directorios **opt/**, **/opt/installer/** y dentro de éste último, los scripts **install.sh** y **wrapper.sh** y los archivos de configuración **rc.local.install**, **iso.conf.dist** y **tpot.conf.dist**.

- **txt.cfg**: indicará al instalador del sistema operativo que modifique las opciones de arranque para poder realizar una instalación desatendida. Su contenido se muestra a continuación:

```
default install
label install
 menu label ^T-Pot 20.06.2 (based on Debian Stable)
 menu default
kernel linux
```

[8]Contiene sólo la mínima cantidad de software para comenzar la instalación y obtener el resto de paquetes a través de Internet

```
        append vga=788 initrd=initrd.gz console-setup/ask_detect=true ↩
            ↪ --
```

Finalmente, el script crea la imagen ISO de T-Pot. En la siguiente imagen se muestra un mapa conceptual para un mejor entendimiento de los pasos realizados.

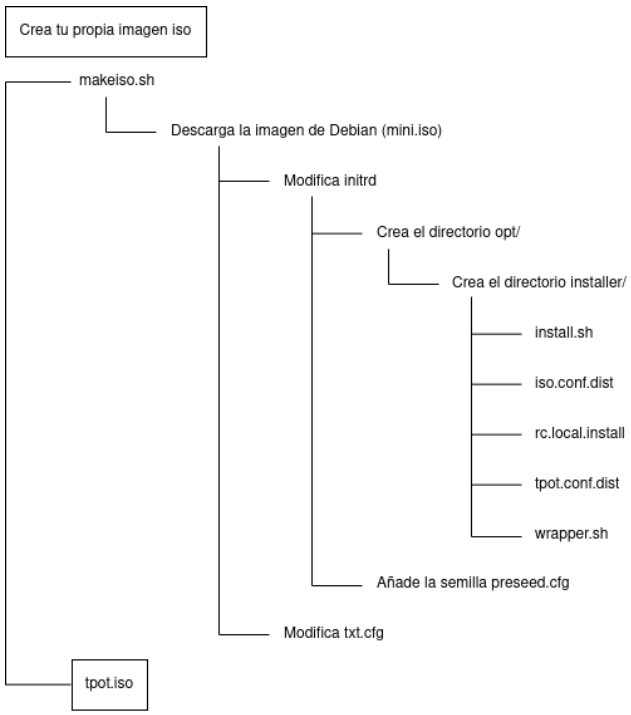


Figura 5.2: Despliegue de la infraestructura. Mapa conceptual del funcionamiento de makeiso.sh.

5.2.5.2 Funcionamiento de la imagen ISO de T-Pot

Una vez arranca el equipo desde la imagen ISO de T-Pot, el instalador busca automáticamente el fichero semilla **preseed.cfg**, lo carga y realiza las instrucciones descritas en su contenido (*véase* `https://github.com/telekom-security/tpotce/blob/master/iso/preseed/tpot.seed`), que se muestran a continuación:

- Definición del país de localización.
- Definición del idioma del teclado.
- Definición de la configuración de red.
- Definición de las particiones del sistema.
- Definición de usuarios de sistema y sus permisos.

- Definición del país del servidor de actualizaciones del sistema y configuración proxy.
- Definición de la configuración del GRUB.
- Definición de la zona horaria.
- Definición de paquetes a instalar.
- Definición de configuraciones post-instalación.
- Descarga del repositorio de T-Pot de GitHub.
- Copia del fichero de configuración rc.local.install del initrd a /etc/rc.local.
- Copia del directorio /opt/installer y su contenido a /root.

Las tareas clave que realiza el instalador son la **instalación de paquetes** y la copia del fichero de configuración **rc.local**.

Los paquetes que se instalan se exponen a continuación:

- **apache2-utils**, **cracklib-runtime**, **curl**, **dialog**, **figlet**, **git**, **grc**, **libcrack2**, **libpq-dev**, **lsb-release**, **net-tools**, **software-properties-common** y **toilet**.

El fichero **rc.local** permite que cualquier comando que se coloque o script al que se llame en dicho fichero se ejecute al final del arranque del sistema operativo.

Si se analiza el fichero rc.local, se obtiene el siguiente código:

Código 5.7: Bash. rc.local

```
#!/bin/bash
#plymouth --quit
openvt -f -w -s /root/installer/wrapper.sh
```

Una vez instalado el sistema operativo, éste arrancará y ejecutará en un terminal virtual el script **wrapper.sh**.

Código 5.8: Bash. wrapper.sh

```
#!/bin/bash
cd /root/installer
./install.sh --type=iso
```

El script wrapper.sh ejecuta el script **install.sh** y le pasa como argumento "–**type=iso**". El análisis de este script, debido a su gran tamaño y la gran cantidad de tareas que realiza, se explica en un apartado dedicado (*véase "5.2.5.3 Análisis del script de instalación install.sh*").

5.2.5.3 Análisis del script de instalación install.sh

Install.sh es un script escrito en **Bash** con alrededor de mil lineas de código (*véase* `https://github.com/telekom-security/tpotce/blob/master/iso/installer/install.sh`). Este script es el componente principal para desplegar el sistema constituido por Deutsche Telekom.

En una primera lectura, se pueden identificar los siguientes apartados:

- Declaración de variables globales.
- Declaración de funciones.
- Fase de pre-instalación.
- Preparación del entorno de instalación.
- Fase de interacción con el usuario.
- Fase de instalación.

Los dos primeros apartados son únicamente declaraciones de variables y funciones por lo que realmente el script empieza a trabajar a partir del tercer apartado (Fase de pre-instalación), realizando dos tareas:

- Consultar si el usuario que ha lanzado el script es el usuario "**root**".
- Comprobar si el sistema tiene instalados los paquetes necesarios, en caso de que falte alguno, realiza la instalación.

El cuarto apartado (Preparación del entorno de instalación) se encarga de realizar las siguientes tareas:

- Comprobar que el sistema operativo desde el que se ejecuta el script es **Debian 10**.
- Cargar el tipo de instalación "**iso**", "**user**" o "**auto**" dependiendo del argumento que se haya pasado al ejecutar el script. En caso de detectar el tipo "auto", guardará también el fichero de configuración que se debe pasar como segundo argumento al ejecutar el script.
- En caso de que la instalación sea del tipo "**user**" se muestra las conexiones activas en ese instante en el sistema y pregunta al usuario si quiere continuar, informando de que esas conexiones activas pueden generar un problema durante la instalación.

En el quinto apartado (Fase de interacción con el usuario):

- Se comprueba que las siguientes páginas web están disponibles:
  - `https://hub.docker.com`
  - `https://github.com`.
  - `https://pypi.python.org`.
  - `https://debian.org`.
  - `https://listbot.sicherheitstacho.eu`.
- Se busca el archivo /root/installer/**iso.conf** que contenga configuraciones personalizadas de proxy, certificados SSL/TLS y Network Time Protocol (NTP).
- Se solicita la edición de sonda que se quiere instalar: Standard, Sensor, Industrial, Collector, Netxgen o Medical.
- Si el tipo de instalación es "**iso**" se pide la contraseña para el usuario administrador que se creará con el nombre de "**tsec**".
- Pide y crea un usuario y una contraseña para acceder a la página web encargada de mostrar los paneles de acceso a las distintas herramientas como Kibana, Spiderfoot o Cyberchef.[9]

El último apartado (Fase de instalación) es el más extenso y también el más importante. A continuación se muestran las tareas más relevantes:

- Instala los paquetes necesarios, entre los que cabe destacar elasticdump, elasticsearch-curator y yq.
- Clona el repositorio de T-Pot `https://github.com/telekom-security/tpotce` en el directorio /opt/tpot.
- Deshabilita el acceso a **Cockpit** con el usuario "**root**".
- Crea el enlace simbólico /opt/tpot/etc/**tpot.yml** al docker-compose que corresponda a la edición de sonda que se haya elegido instalar.
- Descarga las imágenes necesarias de los contenedores configurados en el docker-compose correspondiente.
- Añade la configuración de "**Fail2ban**".
- Añade las siguientes tareas programadas:

[9] En caso de usar la opción de instalación "auto" se crearan mediante el archivo de configuración que se debe adjuntar al script.

  - Revisar a una hora aleatoria si hay imágenes de nuevos de contenedores y actualizarlas una vez al día.
  - Borrar todos los días a una hora aleatoria los índices de los logs de Elasticsearch que superen los 90 días.
  - Copia cada hora los binarios subidos al honeypot "**Dionaea**" en un volumen para ser accesibles desde el equipo anfitrión.
  - Reiniciar a una hora aleatoria la sonda todos los días de la semana excepto los domingos.
  - Actualizar el sistema a una hora aleatoria todos los domingos.

- Crea las carpetas y ficheros correspondientes a los volúmenes de los contenedores.
- Habilita el servicio "**tpot**" para que funcione en el arranque del equipo.
- Reinicia el sistema.

A continuación se muestra el diagrama de flujo resultante:

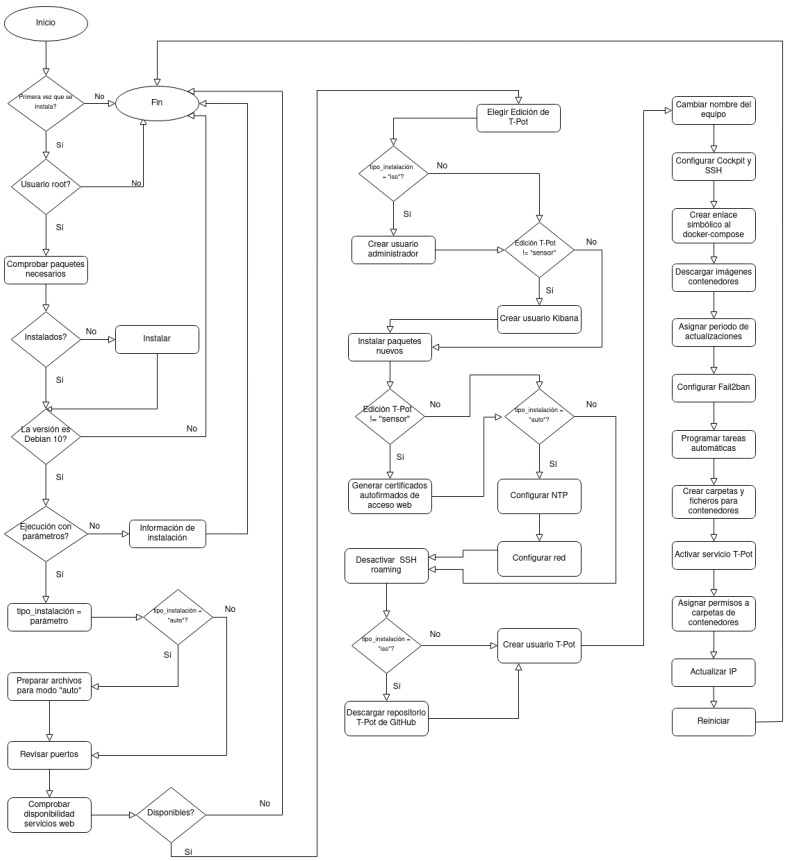


Figura 5.3: Despliegue de la infraestructura. Diagrama de flujo del script install.sh

6 Campañas de engaño

6.0.1 Diseño de campañas

Una campaña es un conjunto de activos de engaño que permite desarrollar una estrategia de contrainteligencia. A lo largo de este proyecto, se han implementado dos campañas que podrían servir como base a las que se determinen en el futuro. De alguna manera, se ha intentado crear una metodología, es decir, una forma de crear campañas que sea reutilizable, por supuesto, siendo lo más flexible posible para poderlas adaptar a los requisitos.

Crear una campaña de engaño efectiva requiere una planificación cuidadosa. Hay cuatro preguntas principales que deben estructurar la fase de diseño:

- ¿Por qué crear una campaña de engaño?
- ¿Quiénes son los adversarios?
- ¿Dónde se desplegarán los activos de la campaña?
- ¿Qué va a hacer la campaña?

Además, es importante tener siempre presente que las campañas son dinámicas y necesitarán adaptarse y evolucionar para mantener su efectividad. Para ello, habrá que analizar periódicamente los resultados y replantear nuevamente las cuatro preguntas previamente mencionadas.

6.0.1.1 ¿Por qué?

La primera pregunta que se debe plantear es: ¿Por qué crear una campaña de engaño? ¿Viene motivado como consecuencia de espionaje y manipulación de un adversario previamente identificado?, ¿o es quizás porque se quiere comprender mejor algún tipo de amenaza específica?

6.0.1.2 ¿Quién?

Para obtener un resultado óptimo de la campaña, se deberán adaptar las estrategias y tácticas a los posibles adversarios, que pueden ser:

- Estados nacionales.

- Grupos cibernéticos.
- Competidores.
- Insiders.
- Activistas.
- Ciber delincuentes.

6.0.1.3 ¿Dónde?

La respuesta a esta pregunta dependerá de los objetivos que se deseen alcanzar en la campaña. Por ejemplo, la recopilación de inteligencia en una red interna, requerirá el despliegue de las sondas en la red interna, mientras que campañas orientadas a abordar amenazas externas, requerirán el despliegue de las sondas en una DMZ o en Internet.

6.0.1.4 ¿Qué?

Es aquí donde se debe decidir que hará la campaña. Por ejemplo, recoger estadísticas de los ataques que se han realizado un mayor número de veces, generar un listado de direcciones IP que no estén asociadas a listados públicos de IPs con mala reputación, crear diccionarios de usuarios y contraseñas obtenidos, etc.

6.0.1.5 Roles

Cada campaña estará formada por varias personas con distintos perfiles: **Arquitecto**, **Administrador** y **Analista**.

El **arquitecto**, se encargarán de diseñar las campañas.

El **administrador**,se encargarán de implementar y desplegar los escenarios.

El **analista**, se encargarán de crear inteligencia basándose en la información obtenida en cada campaña.

6.1 Campaña I. Identificación de amenazas en busca de accesos a sistemas mediante el protocolo VNC

6.1.1 Objetivos

Se exponen a continuación los objetivos principales de la campaña:

- Detectar diariamente el número de intentos de acceso al servicio VNC de una sonda localizada en España.

- Capturar la contraseña en caso de ser introducida por el atacante.
- Identificar la localización de origen de donde proviene el intento de acceso a la sonda.

6.1.2 Análisis

6.1.2.1 Definición de requisitos

A continuación se definirán los requisitos de la campaña:

- La campaña debe previamente tener desplegada una sonda en Internet.
- La sonda debe estar desplegada en un VPS ubicado en España.
- La sonda debe tener desplegado el honeypot Heralding.
- Se debe poder acceder al honeypot Heralding mediante el protocolo VNC.
- El puerto de acceso al servicio VNC debe ser el 5900.
- El atacante podrá introducir una contraseña de acceso al servicio VNC.
- El honeypot devolverá el mensaje de "error de autenticación" para cualquier contraseña.
- La sonda debe crear logs de los intentos de acceso al servicio VNC.
- La sonda debe indexar los logs creados en la base de datos del nodo principal.
- El nodo principal deberá mostrar la información generada al analista.
- El analista podrá ver la dirección IP del equipo que haya accedido al servicio VNC.
- El analista podrá ver el Número de Sistema Autónomo (ASN) al que pertenece la dirección IP del equipo que ha accedido al servicio VNC.
- El analista podrá ver la contraseña introducida por el atacante para acceder servicio VNC.

6.1.2.2 Diagrama de casos de uso

A continuación, se muestra el diagrama de casos de uso:

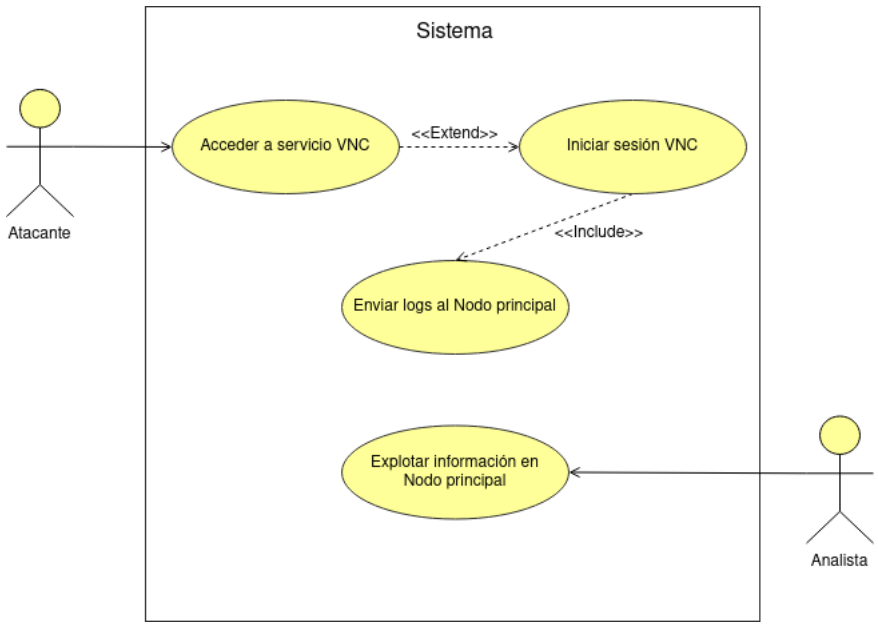


Figura 6.1: Campañas de engaño. Diagrama de casos de uso de identificación de amenazas de acceso por VNC.

6.1.2.3 Definición de actores

A continuación se definen los actores que interaccionarán con el sistema:

- **Analista**: Analista de seguridad que se encargará de recoger la información obtenida de ataques al servicio VNC para más tarde generar inteligencia.
- **Atacante**: atacante intentando acceder al servicio VNC.

6.1.2.4 Modelado

A continuación se mostrará el flujo de acciones que comprende el caso de uso.

Flujo básico:

1. El atacante conecta con el servicio VNC de la sonda.
2. El atacante introduce la contraseña de acceso al servicio VNC.
3. Se registra la acción del atacante en la sonda.
4. Se envía la acción del atacante al nodo principal.
5. Se almacena la acción del atacante en el nodo principal.
6. El analista explota la acción del atacante utilizando el nodo principal.

6.1.3 Diseño

6.1.3.1 Diagrama

A continuación se muestra el diagrama diseñado para la campaña actual.

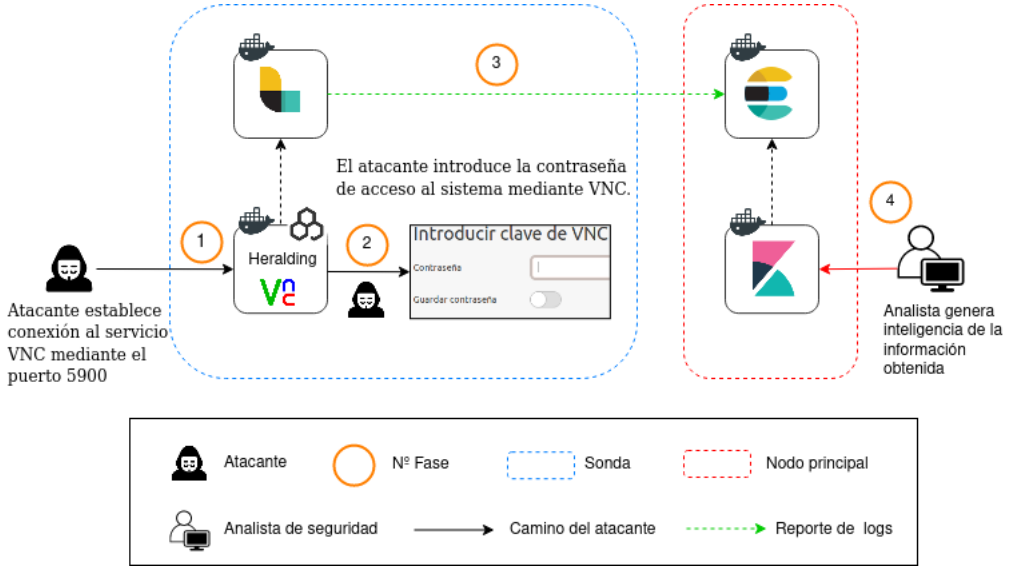


Figura 6.2: Campañas de engaño. Diagrama de una campaña de identificación de amenazas de acceso por VNC.

6.1.3.2 Modelado

El objetivo principal de esta campaña es detectar el número de intentos de acceso por VNC y capturar la contraseña introducida (*véase* 6.1.1 Objetivos).

El intento de acceso podrá provenir de atacantes externos, generalmente de ”bots” automatizados por ciberdelincuentes o grupos cibernéticos.

Para ello, la sonda, desplegada en un VPS localizado en España, levantará en un contenedor el honeypot **Heralding** que permitirá realizar un intento de conexión VNC a través del puerto 5900 (Fase 1).

Heralding, pedirá por pantalla la contraseña (Fase 2) que el atacante podrá introducir. En caso de intentar acceder con alguna contraseña, Heralding devolverá un error de autenticación.

Cada intento de acceso por VNC realizado creará un log que será indexado en la base de datos del nodo principal (Fase 3).

Finalmente, el analista explotará la información obtenida en el nodo principal (Fase 4).

6.1.4 Implementación

En este apartado la implementación ya está realizada, ya que **Heralding** es uno de los honeypots desplegados en un contenedor de la sonda diseñada en este proyecto (*véase* 5.2.1.1 Diseño a medida).

6.1.5 Pruebas y resultados

En esta sección se describen los datos que demuestran el cumplimiento de los objetivos y requisitos de la campaña implementada. Para ello, se han usado un equipo doméstico conectado a Internet con la dirección IP pública **95.62.22.98** y una sonda desplegada en un VPS con la dirección IP pública **5.254.118.157**.

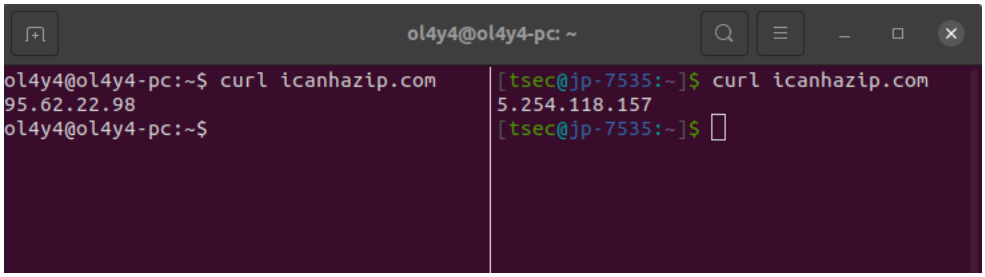


Figura 6.3: Campañas de engaño. Direcciones IP de la prueba realizada.

Desde el equipo doméstico y con la aplicación Remmina, se ha intentado acceder por VNC a la sonda.

Figura 6.4: Campañas de engaño. Conexión VNC a la sonda.

La sonda solicita una contraseña de acceso. Se ha introducido **123456** para intentar acceder,

obteniendo como resultado un error de autenticación.

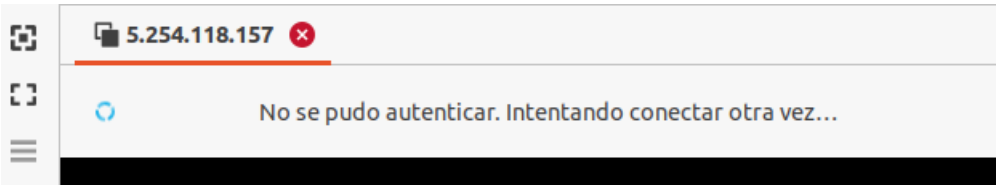


Figura 6.5: Campañas de engaño. Contraseña VNC errónea.

Se ha revisado en el nodo principal los intentos de acceso en los últimos dos minutos. El contador que ilustra el número de intentos de acceso es de 88 intentos con 2 IPs de origen.

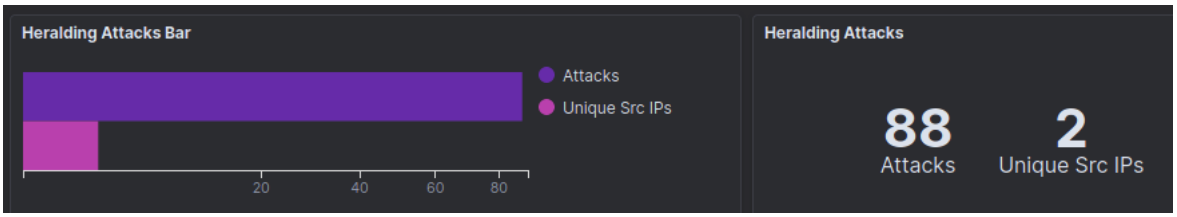


Figura 6.6: Campañas de engaño. Contador de ataques.

Como se puede observar en la siguiente imagen, los países de origen de las dos direcciones IP que han intentado acceder por VNC al puerto 5900 en los últimos dos minutos son España, representado con el color naranja, y Rusia, representado con el color azul.

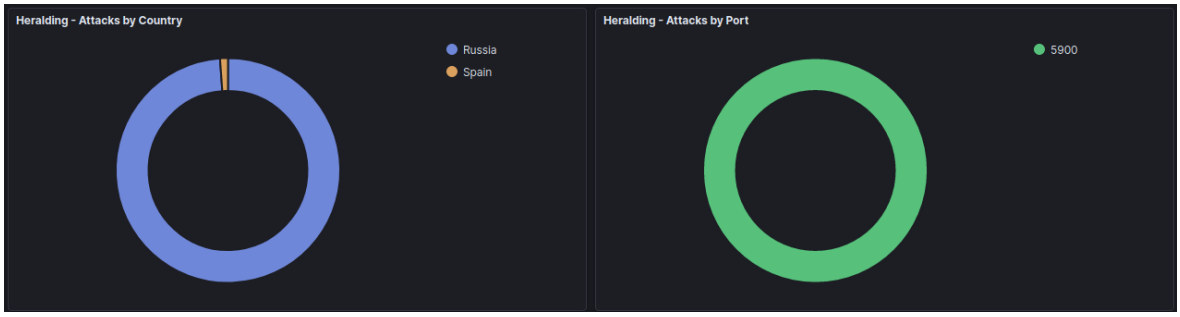


Figura 6.7: Campañas de engaño. Países desde los que se recibe el ataque.

Entre las contraseñas capturadas, se puede ver la introducida en el intento de acceso con el equipo doméstico (123456) entre otras.

Finalmente, aparecen las direcciones IP y ASN correspondientes al equipo doméstico usado.

Para ver la prueba de concepto y los resultados, véase `https://youtu.be/FC7kqzjSneA`.

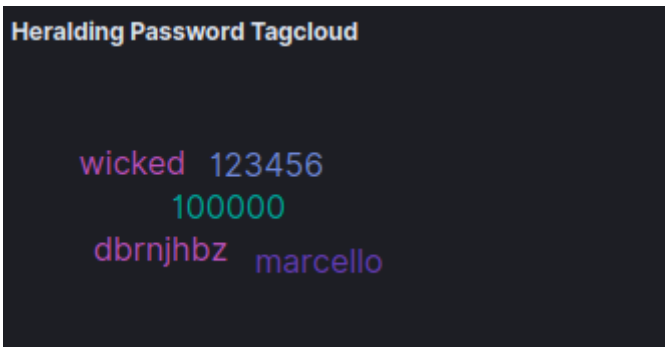


Figura 6.8: Campañas de engaño. Contraseñas capturadas.

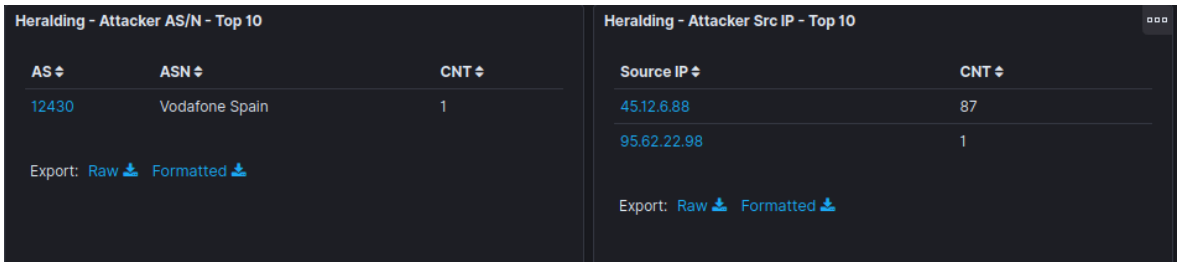


Figura 6.9: Campañas de engaño. IP de origen y ASN.

6.2 Campaña II. Identificación de amenazas sin catalogar

6.2.1 Objetivos

El objetivo principal de esta campaña es detectar en tiempo real las direcciones IP que estén realizando cualquier tipo de ataque a una sonda desplegada en Internet que no estén catalogadas como IPs con mala reputación (*véase "Anexo XIII. Catálogo de reputación de direcciones IP en T-Pot"*).

6.2.2 Análisis

6.2.2.1 Definición de requisitos

A continuación se definirán los requisitos funcionales y no funcionales de la campaña:

- La campaña debe previamente tener desplegada una sonda en Internet.
- La sonda debe crear logs de cualquier ataque recibido.
- La sonda debe indexar los logs creados en la base de datos del nodo principal.
- El nodo principal deberá mostrar toda la información generada al analista.
- El analista podrá aplicar filtros en los paneles de visualización.

- El analista podrá eliminar la visualización de IPs catalogadas como ”Malware”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Mala reputación”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Atacantes conocidos”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Redes anónimas”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Nodo de salida de Tor”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Abuso”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Correo no deseado”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Control de mando de una Botnet”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Comprometida”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”ransomware”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Escaneador masivo”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Bot, Crawler”.
- El analista podrá eliminar la visualización de IPs catalogadas como ”Minero de criptomonedas”.
- El analista podrá ver la información en tiempo real.

6.2.2.2 Diagrama de casos de uso

A continuación, se muestra el diagrama de casos de uso:

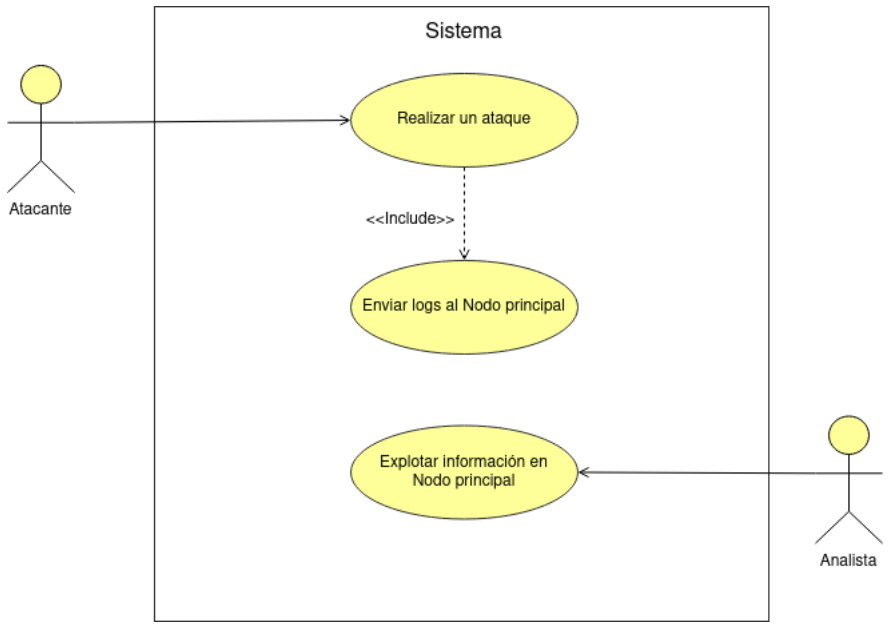


Figura 6.10: Campañas de engaño. Diagrama de casos de uso II.

6.2.2.3 Definición de actores

A continuación se definen los actores que interaccionarán con el sistema:

- **Analista**: Analista de seguridad que se encargará de recoger la información obtenida de los ataques recibidos para más tarde generar inteligencia.
- **Atacante**: atacante intentando vulnerar la sonda.

6.2.2.4 Modelado

A continuación se mostrará el flujo de acciones que comprende el caso de uso.

Flujo básico:

1. El atacante ataca una vulnerabilidad en la sonda.
2. Se registra la acción del atacante en la sonda.
3. Se envía la acción registrada al nodo principal.
4. Se almacena la acción del atacante en el nodo principal.
5. El analista explota la acción del atacante utilizando el nodo principal.

6.2.3 Diseño

6.2.3.1 Diagrama

A continuación se muestra el diagrama diseñado para la campaña actual.

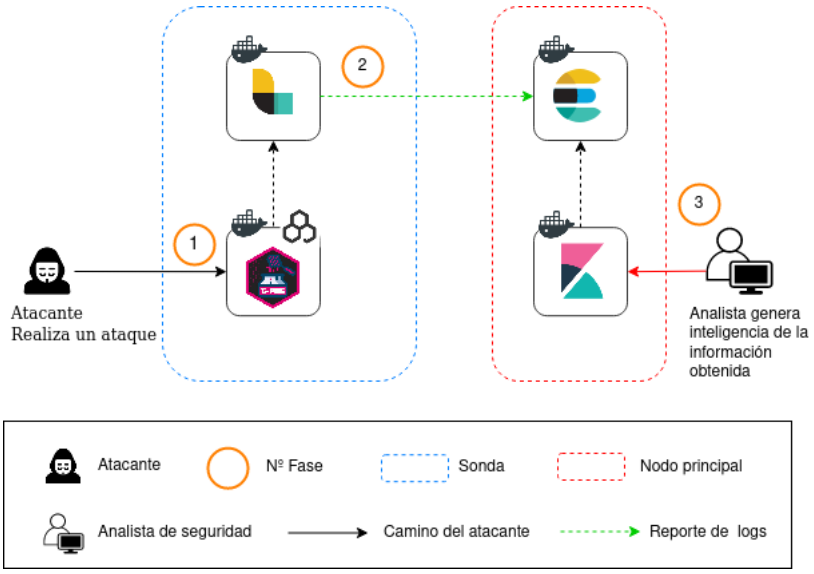


Figura 6.11: Campañas de engaño. Diagrama de una campaña de identificación de amenazas sin catalogar.

6.2.3.2 Modelado

El objetivo principal de esta campaña es identificar posibles atacantes desconocidos por los principales proveedores de IPs con mala reputación (*véase* 6.2.1 Objetivos).

El intento de acceso podrá provenir de atacantes externos, generalmente de ”bots” automatizados por ciberdelincuentes o grupos cibernéticos.

Un atacante podrá realizar un ataque a la sonda (Fase 1).

Cada ataque realizado creará un log que será indexado en la base de datos del nodo principal (Fase 2).

Finalmente, el analista explotará la información obtenida en el nodo principal (Fase 3).

6.2.4 Implementación

En este apartado la implementación es automática debido a que ya están implementados los honeypots en la sonda utilizada en este proyecto (*véase* 5.2.1.1 Diseño a medida).

6.2.5 Pruebas y resultados

En esta sección se describen los datos que demuestran el cumplimiento de los objetivos y requisitos de la campaña implementada. Para ello, se ha usado una sonda desplegada en un VPS con la dirección IP pública **5.254.118.157** y se ha observado la reputación de las IPs que han realizado ataques en los últimos quince minutos en el panel general de todos los ataques recibidos.

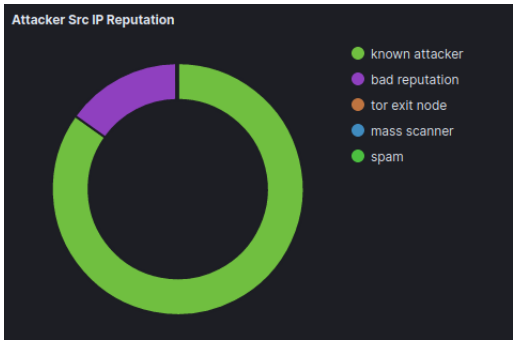


Figura 6.12: Campañas de engaño. Reputación de direcciones IP.

Como se puede observar, la mayoría de las direcciones IP que han realizado algún ataque están etiquetadas. Para identificar las que no lo están, habrá que aplicar un filtro por cada etiqueta con las siguientes características:

- **Field**: ip_rep.keyword.
- **Operator**: is not.
- **Value**: cada una de las etiquetas.

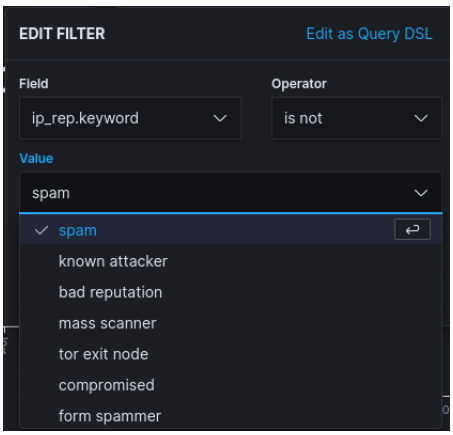


Figura 6.13: Campañas de engaño. Filtros para limpiar etiquetas de reputación.

Una vez aplicados todos los filtros, el panel de la reputación de las IPs aparecerá vacío y se podrán identificar las IPs sin categorizar.

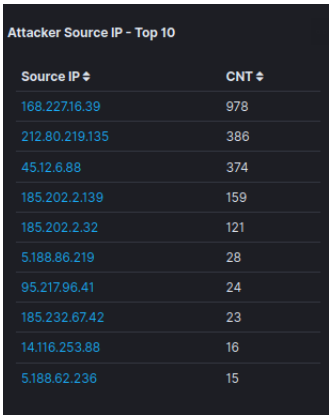


Figura 6.14: Campañas de engaño. IPs sin categorizar.

Para ver la prueba de concepto y los resultados, véase `https://youtu.be/XM_m2FEUB5c`.

A partir de este momento, el analista emprenderá las investigaciones que estime oportuno para categorizar dichas direcciones IP.

7 Conclusiones y lineas futuras

7.1 Conclusiones

En resumen, el auge de la ciberseguridad implica proporcionar a las empresas y organizaciones de herramientas capaces de mejorar la seguridad de sus redes y sistemas, por ello considero que este proyecto es una interesante aportación en este campo.

El despliegue de la infraestructura (sonda y nodo principal) ha sido un desafío importante, tales como el estudio, uso y modificaciones de la sonda de T-Pot, la búsqueda, uso y adaptación de imágenes Docker y la generación de campañas de engaño capaces de motivar e incrementar las hipótesis de los analistas de seguridad. El resultado satisfactorio a todas estas cuestiones y la implementación dada hacen pensar que este proyecto pudiera ser una buena referencia para la obtención de información de las técnicas, tácticas y procedimientos de los atacantes.

Se ha proporcionado una imagen de Docker con un servicio web capaz de actualizar sus certificados SSL/TLS de manera automática, de modo que puede ser reutilizada en futuros proyectos relacionados con este campo u otros.

Dada la complejidad técnica del proyecto, se decidió desde el principio que el producto final debería estar desplegado en sistemas en producción y no quedarse en un marco teórico.

El entorno de Elastic Stack (Elasticsearch, Logstash y Kibana) han superado las expectativas esperadas, siendo sorprendente la información que un analista de seguridad puede visualizar en cada panel, ocultando la complejidad de las tareas que realiza el sistema.

Cabe también una mención especial al entorno basado en contenedores Docker, no solo ha logrado cumplir con los objetivos del proyecto si no que lo ha conseguido ahorrando tiempos y esfuerzos, aportando una gran flexibilidad a la hora de trabajar.

Otra herramienta de indiscutible utilidad ha sido Ansible, permitiendo el despliegue de sonda de manera automática, algo imprescindible cuando el número de éstas empieze a crecer.

Se han aplicado los conocimientos estudiados durante la carrera consiguiendo los objetivos propuestos desde el punto de vista de la Ingeniería. Las herramientas utilizadas durante el desarrollo del proyecto han sido de software libre, pudiendo servir para otros ingenieros como ejemplos a la hora de desarrollar otros proyectos.

En general, considero que el proyecto ha sido muy completo, teniendo que utilizar una gran cantidad de conocimientos técnicos aprendidos durante la carrera, así como otros nuevos que

sin duda me serán de gran utilidad durante el resto de mi carrera profesional como Ingeniero.

7.2 Conocimientos adquiridos

Son muchos los conocimientos adquiridos durante el desarrollo del proyecto y muchos otros que han afianzado los adquiridos durante la carrera, de los cuales cabe destacar los siguientes:

- He adquirido conocimientos respecto a la elaboración de documentación con el formato LaTeX.
- He adquirido conocimientos respecto al lenguaje de serialización YAML.
- La elaboración de documentación me ha llevado a utilizar JabRef, hasta ahora desconocida, para gestionar la bibliografía.
- He afianzado los conocimientos de redes aprendidos durante la carrera, así como su aplicación en el uso de herramientas relacionadas en este campo, como por ejemplo en contenedores Docker.
- He afianzado los conocimientos de sistema operativos aprendidos durante la carrera, más concretamente respecto a la secuencia de arranque.
- He afianzado conocimientos respecto al uso de la herramienta Git, para el control de versiones de los distintos códigos desarrollados.
- A pesar de que antes de comenzar el proyecto tenía un buen nivel de conocimientos relacionados con programación shell, he afianzado algunos conceptos y aprendido otros nuevos, más concretamente respecto a los servicios de Systemd en sistemas Gnu/Linux.
- He utilizado por primera vez una semilla de preconfiguración para automatizar la instalación de un sistema operativo Debian GNU/Linux.
- He afianzado conocimientos desde el punto de vista de la gestión de proyectos.
- He adquirido profundos conocimientos del uso e implementación de honeypots de baja y media interacción.
- Tras una reflexión meticulosa, he conseguido implementar dos campañas de engaño, pudiéndose considerar la semilla para la escritura de una metodología en este campo.

7.3 Mejoras

Como en cualquier proyecto informático, aunque se haya conseguido un producto final que los usuarios puedan utilizar, siempre se puede mejorar. Las mejoras más importantes que se podrían abordar inmediatamente son:

- Aunque para el funcionamiento del sistema no es necesario, se debería implementar una página web en el dominio principal helioslove.com.
- Si bien es cierto que la información va cifrada entre el nodo principal y los componentes externos que acceden a éste, sería conveniente la implementación de un canal seguro entre los servicios de Elasticsearch y Kibana desplegados internamente en el nodo principal, ya que en caso de que un usuario no autorizado acceda al servidor, podrá monitorizar el tráfico y visualizar los datos en texto plano.
- Sería conveniente establecer de forma permanente en el nodo principal el valor 262144 para la memoria virtual del sistema operativo requerida por Elasticsearch para almacenar sus índices, ya que cuando el VPS se reinicia o se apaga hay que modificar el valor de nuevo.
- Se debería crear un playbook de Ansible para el despliegue masivo de las sondas.
- Una vez desplegadas las sondas con Ansible, sería conveniente configurar el acceso a éstas por SSH mediante certificados y no con usuario y contraseña, evitando así posibles ataques de fuerza bruta.
- Ansible dispone de un módulo llamado Ansible Vault que permite el cifrado de los ficheros con información sensible. Sería conveniente utilizarlo para los ficheros que contienen las contraseñas del usuario administrador de la sondas, pudiendo así compartirlos sin problema en GitHub.
- Cuando se importan los objetos de Kibana recogidos de la sonda de T-Pot para tener acceso a los dashboards, se realiza de manera manual. Sería importante conseguir automatizar la ingesta de estos objetos desde el docker-compose creado para levantar los distintos servicios del nodo principal.

7.4 Lineas futuras

El tema abordado incita a seguir dando pasos en futuros trabajos, como por ejemplo:

- Como ya se ha comentando en numerosas ocasiones, este trabajo se ha centrado en la creación de un prototipo. Aunque es funcional y podría ejecutarse en un entorno de

preproducción, habría que realizar un exhaustivo análisis, pudiendo ser objeto de otro proyecto por sí mismo, para desplegar la base de datos Elasticsearch de manera óptima.

- Sería interesante integrar los resultados obtenidos con un SIEM, combinando su capacidad de correlación con los datos obtenidos de las sondas. Esta integración proporcionaría alertas tempranas sobre ataques realizados tanto por atacantes conocidos como desconocidos utilizando técnicas usuales o nuevas, identificación de malware desconocido o exploits 0day, comportamientos maliciosos que estaban siendo clasificados como legítimos, etc.

- También sería interesante compartir los resultados obtenidos en un Malware Information Sharing Platform (MISP). Esta plataforma, desarrollada por las las Fuerzas Armadas Belgas, es usada para compartir, almacenar y correlacionar indicadores de compromiso[1] de manera colaborativa sobre amenazas existentes.

- Se debería estudiar el uso de sandboxes, de tal forma que el malware obtenido de las sondas pudiera ser analizado de forma efectiva.

- Sería de tremenda utilizad disponer de bases de datos relacionales y no relacionales que, con técnicas de Big Data y Machine Learning, pudieran predecir y analizar ciberataques.

- Tras el análisis de la sonda de T-Pot, se podrían modificar las configuraciones actuales de los honeypots en busca de otros objetivos o incluso se podrían añadir honeypots nuevos.

- Los paneles de Kibana a los que tiene acceso un analista de seguridad vienen marcados por los exportados de la sonda de T-Pot. Sin embargo, sería interesante crear paneles personalizados que una analista de seguridad pudiera necesitar durante el desarrollo de una campaña de engaño.

- Enlazando con el punto anterior y debido al potencial de Kibana a la hora de mostrar información, se debería valorar la posibilidad de importar datos externos que un analista de seguridad pudiera tener y correlacionarlos con los datos proporcionados por las sondas.

- Un aparato importante sería conseguir anonimizar los servicios de acceso a la base de datos y a los paneles de información.

- Para una mejor gestión del entorno, sería recomendable migrar el entorno de Docker-Compose a Kubernetes.

[1] Información relevante que describe cualquier incidente de ciberseguridad, actividad y/o artefacto malicioso.

- Finalmente, probar la infraestructura y conceptos presentados en este proyecto en la creación de nuevas campañas de engaño. En este sentido, pienso que he puesto la semilla para la escritura de una metodología para el análisis, diseño e implementación de este tipo de campañas.

Bibliografía

Aivaliotis, D. (2016). *Mastering nginx*. Packt Publishing Ltd.

Ansible. (s.f.). Descargado 2021-03-01, de `https://github.com/ansible`

Armijos, J. I. C., y Hecht, P. (s.f.). Honeypot como herramienta de prevención de ciberataques.

Barnum, S. (2012). Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation*, *11*, 1–22.

Barrell, R. (2018). Countercraft:: Plataforma de ciber-engaño para detección temprana de vulnerabilidades y creación de una base de inteligencia de amenazas en tiempo real. *Revista SIC: ciberseguridad, seguridad de la información y privacidad*, *27*(132), 154–156.

Blog, R. (s.f.). *What Is A Honeypot?* Descargado 2021-03-01, de `https://www.redlegg.com/blog/what-is-a-honeypot`

Certbot. (s.f.). Descargado 2021-03-01, de `https://certbot.eff.org/`

Chapendama, S. (2019, noviembre). *Analysing Honeypot Data using Kibana and Elasticsearch.* Descargado 2021-03-01, de `https://towardsdatascience.com/analysing-honeypot-data-using-kibana-and-elasticsearch-5e3d61eb2098`

CVE-2018-0101. (2018, enero). Descargado 2021-03-01, de `https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-0101`

Distribuyendo nuestra clave pública de SSH con Ansible. (2019, enero). Descargado 2021-03-01, de `https://www.elarraydejota.com/distribuyendo-nuestra-clave-publica-de-ssh-con-ansible/`

Docker Engine overview. (2021, febrero). Descargado 2021-03-01, de `https://docs.docker.com/engine/`

Elasticsearch & NGINX – Better Together. (2015, marzo). Descargado 2021-03-01, de `https://www.nginx.com/blog/nginx-elasticsearch-better-together/`

ELK Stack: Elasticsearch, Logstash, Kibana | Elastic. (s.f.). Descargado 2021-03-01, de https://www.elastic.co/es/what-is/elk-stack

Estrella Quijije, G. D. (2011). *Diseño del prototipo de una honeypot virtual que permita mejorar el esquema de seguridad en las redes de la carrera de ingeniería en sistemas computacionales y networking de la universidad de guayaquil.* (Tesis Doctoral no publicada). Universidad de Guayaquil. Facultad de Ciencias Matematicas y Fisicas ….

Farinango Endara, H. P. (s.f.). Detección de anomalías con elastic stack.

García, J. (2009). Pequeño pero matón: servicio web y proxy inverso con el veloz nginx. *Linux magazine*(55), 26–30.

G.B, S. (s.f.). *Como arranca Linux explicado paso a paso.* Descargado 2021-03-01, de https://www.sololinux.es/como-arranca-linux-explicado-paso-a-paso/

González Cortés, C. J., y cols. (2018). *Detectando honeypots: Estudio, análisis y mejora de la ofuscación de honeypots* (B.S. thesis).

Hochstein, L., y Moser, R. (2017). *Ansible: Up and running: Automating configuration management and deployment the easy way.* " O'Reilly Media, Inc.".

Introducción a Cyber Threat Intelligence. (2020, abril). Descargado 2021-03-01, de https://ciberseguridad.oesia.com/introduccion-a-cyber-threat-intelligence/

Jangla, K. (2018). *Accelerating development velocity using docker: Docker across microservices.* Apress.

Kong, C. (2017, noviembre). *Ansible Playbook: Deploy the public key to remote hosts.* Descargado 2021-03-01, de https://medium.com/@visualskyrim/ansible-playbook-deploy-the-public-key-to-remote-hosts-da3f3b4b5481

Kotenko, I., Kuleshov, A., y Ushakov, I. (2017). Aggregation of elastic stack instruments for collecting, storing and processing of security information and events. En *2017 ieee smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (smartworld/scalcom/uic/atc/cbdcom/iop/sci)* (pp. 1–8).

Los ciberdelincuentes van por delante porque invierten en talento | Economía 3. (2014, mayo). Descargado 2021-03-01, de https://economia3.com/los-ciberdelincuentes-van-siempre-por-delante-porque-invierten-en-tal/

Mairh, A., Barik, D., Verma, K., y Jena, D. (2011). Honeypot in network security: a survey. En *Proceedings of the 2011 international conference on communication, computing & security* (pp. 600–605).

Malware. (2021, febrero). Descargado 2021-03-01, de https://es.wikipedia.org/w/index.php?title=Malware&oldid=133120470 (Page Version ID: 133120470)

Martínez, D. R. (2019). Expertos en engañar a los ciberdelincuentes: Countercraft: el arte de poner 'trampas' para cazar a los hackers. *Emprendedores: las claves de la economía y el éxito profesional*(266), 40–41.

McKendrick, R., y Gallagher, S. (2017). *Mastering docker.* Packt Publishing Ltd.

MIR. (2019, septiembre). *Estudio sobre la cibercriminalidad en españa.* Descargado de http://www.interior.gob.es/documents/10180/9814700/EstudiosobrelaCibercriminalidadenEspaña2019.pdf/24bd3afb-5a8e-4767-9126-c6c3c256982b

Modificar la imagen de arranque initrd del instalador Debian. (s.f.). Descargado 2021-03-01, de https://gnulinuxvagos.es/topic/6276-modificar-la-imagen-de-arranque-initrd-del-instalador-debian/

Nedelcu, C. (2010). *Nginx http server: Adopt nginx for your web applications to make the most of your infrastructure and serve pages faster than ever.* Packt Publishing Ltd.

NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy. (s.f.). Descargado 2021-03-01, de https://www.nginx.com/

Overview of Docker Compose. (2021, febrero). Descargado 2021-03-01, de https://docs.docker.com/compose/

Plataforma Honeypot todo en uno. (2020, marzo). Descargado 2021-03-01, de https://gurudelainformatica.es/plataforma-honeypot-todo-en-uno

Ponsico Martin, P. (2017). *Tecnología de contenedores docker* (B.S. thesis). Universitat Politècnica de Catalunya.

Porras Suriaga, C. M. (2020). *Análisis y diseño de una arquitectura de red utilizando las herramientas honeypot, ids y firewall* (B.S. thesis). Machala: Universidad Tëcnica de Machala.

Public version. (s.f.). *Using honeypots to detect and analyze attack patterns on cloud infrastructures.* https://davidebove.com/files/thesis-bove-public.pdf. (Accessed: 2021-3-1)

Que es un insider. (2016, noviembre). Descargado 2021-03-01, de `https://comunidadblogger.net/que-es-un-insider/`

Raj, P., Chelladhurai, J. S., y Singh, V. (2015). *Learning docker.* Packt Publishing Ltd.

Reviglio, V., y Frias, P. (2018). Inteligencia local en un centro de operaciones de ciberseguridad. En *Xxi concurso de trabajos estudiantiles (est)-jaiio 47 (caba, 2018).*

Rohaut, S. (2015). *Linux: Preparación para la certificación lpic-1: Exámenes lpi 101 y lpi 102.* Ediciones Eni.

Sanz, V. A., y Lillo, R. V. (2015). Ciberamenazas emergentes: a qué nos enfrentamos y cómo las combatimos. *Revista SIC: ciberseguridad, seguridad de la información y privacidad*(114), 90–92.

Seifert, C., Welch, I., Komisarczuk, P., y cols. (2007). Honeyc-the low-interaction client honeypot. *Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand*, *6*.

Smaldone, J. (2004). *Introducción a secure shell.* Obtenido de http://es. tldp. org/Tutoriales/doc-ssh-intro/introduccion_ssh-0 ….

Smith, R. (2017). *Docker orchestration.* Packt Publishing Ltd.

s.r.o, B. S. (s.f.). *GanttProject: free project management tool for Windows, macOS and Linux.* Descargado 2021-03-01, de `https://www.ganttproject.biz`

telekom-security/tpotce. (2021, marzo). Descargado 2021-03-01, de `https://github.com/telekom-security/tpotce` (original-date: 2014-11-28T16:57:47Z)

Tiefenau, C., von Zezschwitz, E., Häring, M., Krombholz, K., y Smith, M. (2019). A usability evaluation of let's encrypt and certbot: Usable security done right. En *Proceedings of the 2019 acm sigsac conference on computer and communications security* (pp. 1971–1988).

tmux/tmux. (s.f.). Descargado 2021-03-01, de `https://github.com/tmux/tmux`

Toaquiza, Y., y Vicente, A. (2018). *Implementación del provisionamiento automático de configuraciones (network automation) en infraestructuras multivendor con ansible.* (B.S. thesis). Escuela Superior Politécnica de Chimborazo.

T-Pot: Una colmena de Honeypots para atraparlos a todos. (s.f.). Descargado 2021-03-01, de `https://www.elladodelmal.com/2017/07/t-pot-una-colmena-de-honeypots-para.html`

Ubuntu 18.04 LTS. (s.f.). Descargado 2021-03-01, de `https://www.ncsc.gov.uk/collection/end-user-device-security/platform-specific-guidance/ubuntu-18-04-lts`

Yoshida, N., Ata, S., Nakayama, H., y Hayashi, T. (2017). Automation of network operations by cooperation between anomaly detections and operation logs. En *Globecom 2017-2017 ieee global communications conference* (pp. 1–6).

¿Qué es Kibana? (s.f.). Descargado 2021-03-01, de `https://www.elastic.co/es/what-is/kibana`

Lista de Acrónimos y Abreviaturas

ADB	Android Debug Bridge.
ADC	Application Delivery Controller.
ASN	Número de Sistema Autónomo.
BITKOM	German Federal Association for Information Technology, Telecommunications and New Media.
BSI	German Federal Office for Information Security.
CEH	Certified Ethical Hacker.
COVID-19	enfermedad del coronavirus.
CPD	Centro de Proceso de Datos.
CTI	Cyber Thread Intelligence.
DDOS	denegación de servicio.
DICOM	Digital Imaging and Communication On Medicine.
DMZ	Zona Desmilitarizada.
EPMAP	Endpoint Mapper.
FHIR	Fast Healthcare Interoperability Resources.
FTP	File Transfer Protocol.
HL7	Health Level Seven.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
I+D	investigación y desarrollo.
IaaS	Infraestructuras como Servicio.
ICS	Sistemas de Control Industrial.
IDS	Sistemas de Detección de Intrusos.
IMAP	Internet Message Access Protocol.
IPS	Sistemas de Prevención de Intrusos.
LPIC-1	Linux Server Professional.
MIR	Ministerio del Interior.
MISP	Malware Information Sharing Platform.

MSSQL	Microsoft Structured Query Language Server.
NSM	Network Security Monitoring.
NTP	Network Time Protocol.
POP3	Post Office Protocol.
RDP	Remote Desktop Protocol.
RGPD	Reglamento General de Protección de Datos.
SAP	Systems Applications Products.
SIEM	Security Information and Event Management.
SIP	Session Initiation Protocol.
SMB	Server Message Block.
SMTP	Simple Mail Transfer Protocol.
SSH	Secure SHell.
SSL/TLS	Secure Sockets Layer / Transport Layer Security.
TFG	Trabajo Final de Grado.
TFTP	Trivial File Transfer Protocol.
TLD	Top Level Domain.
UFW	Uncomplicated Firewall.
VNC	Virtual Network Computing.
VPS	Virtual Private Server.
YAML	YAML Ain't Markup Language.

Anexo I. Programas y herramientas de T-Pot

- **Adbhoney**: honeypot de baja interacción que simula un dispositivo Android con el sistema Android Debug Bridge (ADB) expuesto en el puerto 5555.
- **Ciscoasa**: honeypot de baja interacción que simula un dispositivo Cisco ASA vulnerable en la funcionalidad VPN Secure Sockets Layer, permitiendo la ejecución remota de código y ataques DDOS.
- **Citrixhoneypot**: honeypot de baja interacción que detecta y registra saltos de directorio no permitidos en Citrix Application Delivery Controller (ADC).
- **Conpot**: honeypot de baja interacción que simula ICS.
- **Cowrie**: honeypot de media interacción que registra ataques de fuerza bruta a los protocolos Telnet y SSH y comandos utilizados por los atacantes en un terminal.
- **Dicompot**: honeypot de baja interacción que simula un servidor Digital Imaging and Communication On Medicine (DICOM)[1].
- **Dionaea**: honeypot de baja interacción capaz de capturar muestras de malware mediante los servicios Server Message Block (SMB), Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), MySQL, Microsoft Structured Query Language Server (MSSQL), Endpoint Mapper (EPMAP) o Session Initiation Protocol (SIP).
- **Elasticpot**: honeypot de baja interacción que simula un servidor Elasticsearch vulnerable.
- **Glutton**: honeypot de baja interacción que simula el acceso a un servidor SSH para capturar el nombre de usuario y la contraseña.
- **Heralding**: honeypot de baja interacción que recopila credenciales de acceso mediante los protocolos FTP, SSH, Telnet, HTTP, HTTPS, Post Office Protocol (POP3), Internet Message Access Protocol (IMAP), Simple Mail Transfer Protocol (SMTP), VNC, PostgreSQL, y Socks5.

[1] estándar de transmisión de imágenes médicas y datos entre hardware de propósito médico.

- **Honeysap**: honeypot de baja interacción que simula un sistema Systems Applications Products (SAP).
- **Ipphoney**: honeypot de baja interacción que simula una impresora expuesta en Internet.
- **Mailhoney**: honeypot de baja interacción que simula un servicio SMTP. Registra emails que se estén intentando enviar y credenciales de acceso.
- **Medpot**: honeypot de baja interacción que simula los estándares Fast Healthcare Interoperability Resources (FHIR) y Health Level Seven (HL7).
- **Rdpy**: honeypot de baja interacción que simula el servicio de Microsoft RDP.
- **Snare/Tanner**: Snare es un honeypot de baja interacción que simula una aplicación web vulnerable. Tanner se encarga de decirle a Snare como debe responderle a los atacantes.
- **Cockpit**: herramienta de administración de sistemas Linux mediante una interfaz web.
- **Cyberchef**: herramienta que permite analizar y decodificar datos mediante una interfaz web.
- **ELK Stack**: es el conjunto de herramientas formado por Elasticsearch, Logstash y Kibana.
- **Elasticsearch Head**: herramienta que permite gestionar bases de datos Elasticsearch mediante una interfaz web.
- **Fatt**: herramienta de monitorización de redes.
- **Spiderfoot**: herramienta para la obtención de información en fuentes abiertas.
- **Suricata**: herramienta de detección de amenazas. Tiene capacidad para ejercer de IDS, IPS y Network Security Monitoring (NSM).

Anexo II. Instalación de T-Pot mediante una imagen ISO

Para instalar T-Pot en su forma más básica (mediante una imagen ISO preconstruida por Deutsche Telekom) se han de seguir los siguientes pasos:

1. Descargar la imagen ISO haciendo click en el siguiente enlace `https://github.com/telekom-security/tpotce/releases/download/20.06.1/tpot.iso`.

2. Encender el equipo y arrancarlo con la imagen ISO descargada en el paso anterior (esto se puede hacer mediante un USB Bootable, un DVD o, en caso de usar algún entorno de virtualización, cargando la imagen ISO en el lector de DVD de la máquina virtual).

3. Iniciar el instalador de T-Pot.

Figura .1: Anexo II. Instalador de T-Pot.

4. Elegir el país de localización.

5. Elegir el idioma del teclado.

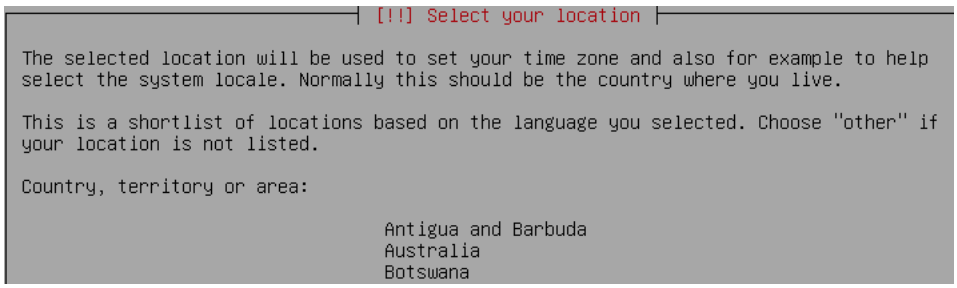


Figura .2: Anexo II. País de localización.

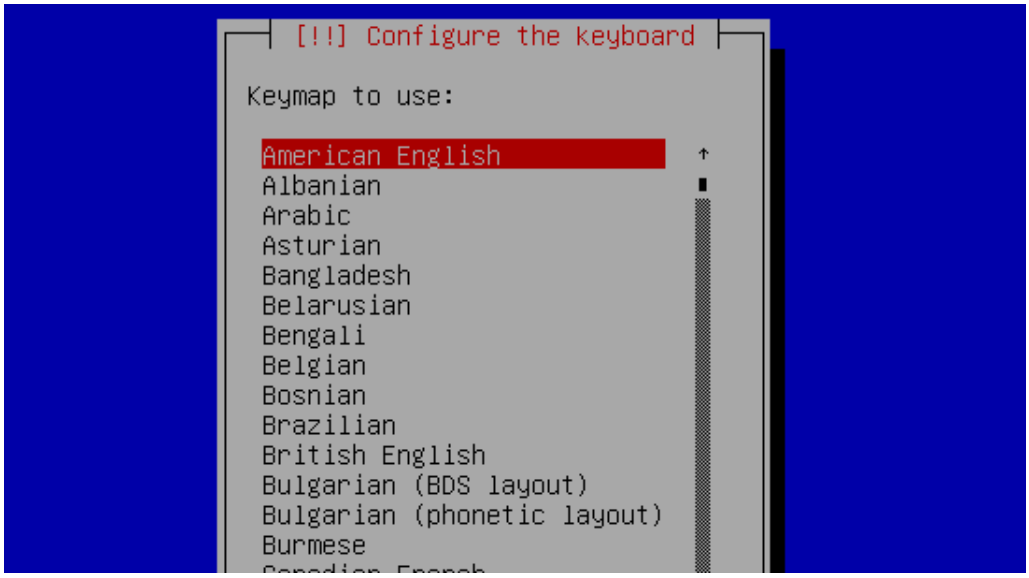


Figura .3: Anexo II. Idioma para el teclado.

6. En caso de encontrar un dirección IP mediante el protocolo DHCP, como es el caso, se configurarán los datos de red automáticamente, si no existe un servicio de DHCP en la red, habrá que configurarlo manualmente.

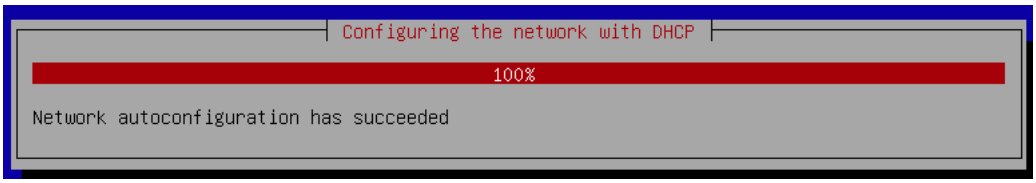


Figura .4: Anexo II. Configuración de red.

7. Elegir el país del repositorio de descarga de paquetes de Debian.

Figura .5: Anexo II. País del repositorio de descarga de paquetes Debian.

8. Elegir el repositorio de descarga de paquetes de Debian.

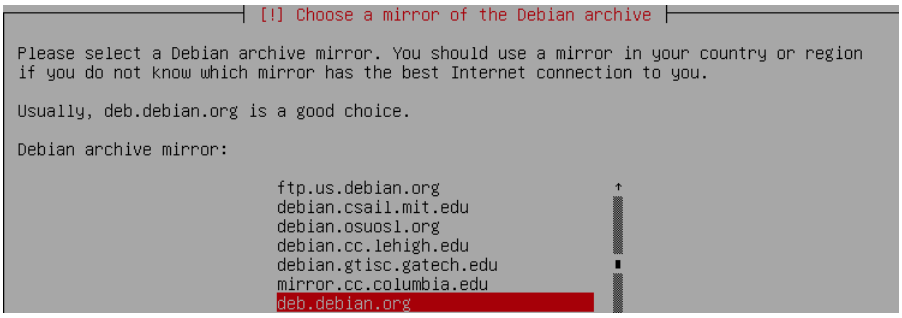


Figura .6: Anexo II. Repositorio de descarga de paquetes Debian.

9. Configurar un proxy, en este caso se omite este paso siguiendo adelante sin introducir nada.

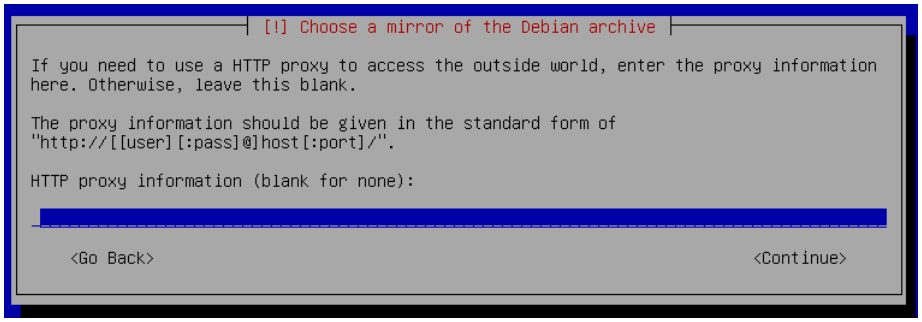


Figura .7: Anexo II. Configuración proxy.

10. El sistema operativo comenzará a instalarse hasta reiniciarse. Una vez arrancado el sistema operativo instalado (hay que modificar el arranque del sistema para que no lo haga desde la imagen ISO), habrá que elegir el tipo de sonda que se quiere desplegar, en este caso el modelo Standar.

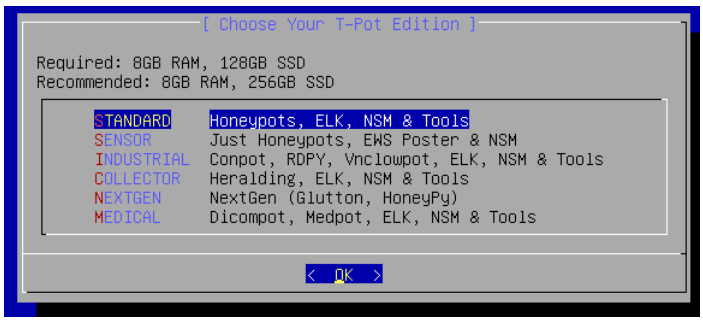


Figura .8: Anexo II. Elección del tipo de sonda.

11. Introducir la contraseña para el usuario administrador del sistema (t-sec).

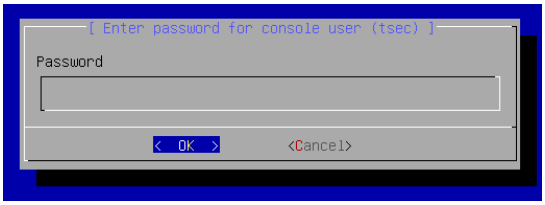


Figura .9: Anexo II. Contraseña del usuario administrador.

12. Introducir el nombre de usuario para acceder vía web a Kibana.

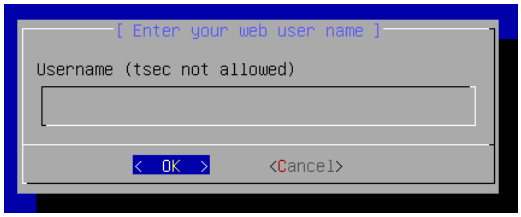


Figura .10: Anexo II. Usuario de acceso a Kibana.

13. Introducir la contraseña para el usuario creado en el paso anterior.

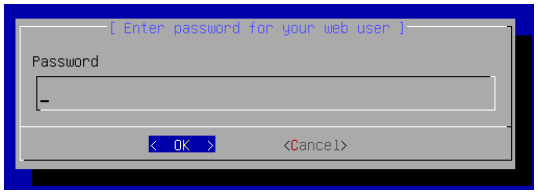


Figura .11: Anexo II. Contraseña de acceso a Kibana.

14. Comenzarán a instalarse las distintas aplicaciones, se reiniciará el equipo al terminar y

la sonda estará funcionando.

Figura .12: Anexo II. T-Pot instalado y preparado.

Anexo III. Instalación de Docker y Docker Compose

A continuación, se mostrará como instalar Docker y Docker-Compose en Ubuntu 20.04 LTS.

1. Actualizar la lista de paquetes disponibles y sus versiones.

```
root@host:~# apt update
```

2. Instalar paquetes necesarios.

```
root@host:~# apt update install apt-transport-https ca-↩
    ↪ certificates curl software-properties-common -y
```

3. Agregar la clave GPG del repositorio oficial.

```
root@host:~# apt update install apt-transport-https ca-↩
    ↪ certificates curl software-properties-common -y
```

4. Añadir el repositorio de Docker.

```
root@host:~# curl -fsSL https://download.docker.com/linux/ubuntu/↩
    ↪ gpg | sudo apt-key add -
```

5. Actualizar de nuevo la lista de paquetes disponibles y sus versiones.

```
root@host:~# apt update
```

6. Instalar Docker.

```
root@host:~# apt install docker-ce
```

7. Instalar Docker-Compose.

```
root@host:~# curl -L "https://github.com/docker/compose/releases/↩
    ↪ download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o ↩
    ↪ /usr/local/bin/docker-compose
```

8. Asignar permisos de ejecución.

```
root@host:~# chmod +x /usr/local/bin/docker-compose
```

Anexo IV. Archivos de configuración de Nginx personalizados

```
##############################################################################
                  Dominio - helioslove.com
##############################################################################
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name helioslove.com;

    ssl_certificate /etc/letsencrypt/live/helioslove.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/helioslove.com/privkey.pem↩
        ↪ ;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!MD5;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    location / {
           try_files $uri $uri/ =404;
```

```
    }

# ssl_session_timeout 1d;
# ssl_session_cache shared:MozSSL:10m; # about 40000 sessions
# ssl_session_tickets off;

    # modern configuration
# ssl_protocols TLSv1.3;
# ssl_prefer_server_ciphers off;

    # HSTS (ngx_http_headers_module is required) (63072000 seconds)
# add_header Strict-Transport-Security "max-age=63072000" always;
}
```

```
    ################################################################################
                  Subdominio - god.helioslove.com
    ################################################################################
    server {
        listen 80;
        listen [::]:80;
        server_name god.helioslove.com;
        return 301 https://god.helioslove.com$request_uri;
    }

    server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name god.helioslove.com;

        ssl_certificate /etc/letsencrypt/live/god.helioslove.com/↩
            ↪ fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/god.helioslove.com/↩
            ↪ privkey.pem;
        ssl_session_timeout 1d;
        ssl_session_cache shared:MozSSL:10m; # about 40000 sessions
        ssl_session_tickets off;
```

```
    # modern configuration
    ssl_protocols TLSv1.3;
    ssl_prefer_server_ciphers off;

    # HSTS (ngx_http_headers_module is required) (63072000 seconds)
    add_header Strict-Transport-Security "max-age=63072000" always;

    location / {
           proxy_pass http://10.10.10.3:5601;
    }
}
```

```
##################################################################################
              Subdominio - sensors.helioslove.com
##################################################################################
server {
    listen 80;
    listen [::]:80;
    server_name sensors.helioslove.com;
    return 301 https://sensors.helioslove.com$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name sensors.helioslove.com;

    ssl_certificate /etc/letsencrypt/live/sensors.helioslove.com/↩
        ↪ fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/sensors.helioslove.com↩
        ↪ /privkey.pem;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m; # about 40000 sessions
    ssl_session_tickets off;
```

```
        # modern configuration
        ssl_protocols TLSv1.3;
        ssl_prefer_server_ciphers off;

        # HSTS (ngx_http_headers_module is required) (63072000 seconds)
        add_header Strict-Transport-Security "max-age=63072000" always;

        location / {
               proxy_pass http://10.10.10.2:9200;
        }
    }
```

Anexo V. Scripts necesarios para la imagen Docker

Código 1: Dockerfile. Script de renovación de certificados

```
#########################################################################
                        renew.sh
#########################################################################
#!/bin/bash

## Para el servicio web
service nginx stop
## Renueva certificados Let's Encrypt
certbot renew
## Arranca el servicio web
service nginx start
```

Código 2: Dockerfile. Script para mantener el contenedor activo

```
#########################################################################
                        run.sh
#########################################################################
#!/bin/bash

## Inicia el servicio web
service nginx start
## bucle infinito
while true
do
    sleep 10000
done
```

Anexo VI. Imagen Docker desarrollada

Código 3: Dockerfile. Imagen personalizada de Nginx

```
FROM ubuntu:18.04

#Se configura el horario del sistema para que no lo pida al instalar ↩
    ↪ certbot
ENV TZ=Europe/Madrid
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/↩
    ↪ timezone

#Instalación de NGINX y Certbot
RUN apt update && apt install -y nginx certbot

#Se borra la configuración por defecto de NGINX
RUN rm /etc/nginx/sites-available/default
RUN rm /etc/nginx/sites-enabled/default

#Se crea un directorio para la página web del dominio
RUN mkdir /var/www/helioslove.com

#Se copia la configuración de NGINX para el dominio y los subdominios
COPY helioslove /etc/nginx/sites-available/
COPY god.helioslove /etc/nginx/sites-available/
COPY sensors.helioslove /etc/nginx/sites-available/

#Se crean los enlaces simbólicos a los ficheros de configuración de ↩
    ↪ cada dominio y subdominio que lee NGINX
RUN ln -s /etc/nginx/sites-available/helioslove /etc/nginx/sites-↩
    ↪ enabled/
RUN ln -s /etc/nginx/sites-available/god.helioslove /etc/nginx/sites-↩
```

```
    ↪ enabled/
RUN ln -s /etc/nginx/sites-available/sensors.helioslove /etc/nginx/↩
    ↪ sites-enabled/

#Script para renovar los certificados de Let's encrypt
COPY renew.sh /renew.sh
RUN chmod +x /renew.sh
#Tarea programada para que se ejecute todos los lunes a las 7.15h
RUN echo "15 7 * * 1 root /renew.sh" >> /etc/crontab

#Script para levantar el servicio de NGINX y mantener el docker vivo
COPY run.sh /run.sh
RUN chmod +x /run.sh

CMD sh /run.sh
```

Anexo VII. Fichero de configuración docker-compose

Código 4: Docker Compose. docker-compose.yml

```
version: '3.7'
services:
  elasticsearch:
    image: elasticsearch:7.10.1
    container_name: elastic
    environment:
      - node.name=es01
      - cluster.name=tfg-cluster
      - cluster.initial_master_nodes=es01
      - bootstrap.memory_lock=true
# Asignación de 4GB de RAM a la vm de JAVA
      - "ES_JAVA_OPTS=-Xms4g -Xmx4g"
      - ELASTIC_PASSWORD=1234567890
      - xpack.license.self_generated.type=basic
      - xpack.security.enabled=true
      - xpack.monitoring.collection.enabled=true
      - xpack.security.transport.ssl.enabled=true
      - xpack.security.audit.enabled=true
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - elastic-data:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
```

```
    networks:
      app_net:
        ipv4_address: 10.10.10.2

  kibana:
    image: kibana:7.10.1
    container_name: kibana
    ports:
      - 5601:5601
    environment:
     - ELASTICSEARCH_USERNAME=changeme
     - ELASTICSEARCH_PASSWORD="changeme"
    networks:
      app_net:
        ipv4_address: 10.10.10.3

  nginx:
    image: nginx-nodo-principal:2.0
    build: .
    container_name: nginx
    volumes:
      - nginx-web:/var/www/helioslove.com
      - nginx-certs:/etc/letsencrypt/archive
      - nginx-certs-enlaces:/etc/letsencrypt/live

    ports:
      - 80:80
      - 443:443
    networks:
      app_net:
        ipv4_address: 10.10.10.4

volumes:
  elastic-data:
    driver: local
  nginx-web:
    driver: local
  nginx-certs:
```

```
    driver: local
  nginx-certs-enlaces:
    driver: local

networks:
  app_net:
    ipam:
      driver: default
      config:
        - subnet: "10.10.10.0/24"
```

Anexo VIII. Script para el despliegue automatizado de sondas

Código 5: Bash. helios.sh

```
#!/bin/bash

myUSER_ELASTIC="${1#*=}"
myPASS_ELASTIC="${2#*=}"
myADMIN_PASS="${3#*=}"

apt update
apt -y install apache2-utils cracklib-runtime curl dialog figlet git ↩
    ↪ grc libcrack2 libpq-dev lsb-release net-tools software-properties↩
    ↪ -common toilet sudo
apt -y install grub-pc
git clone --depth=1 https://github.com/telekom-security/tpotce /opt/↩
    ↪ tpot
git clone https://github.com/0l4y4/helioslove /opt/helioslove
sed -i "s/changeme_user/$myUSER_ELASTIC/g" /opt/helioslove/logstash.↩
    ↪ conf
sed -i "s/changeme_password/$myPASS_ELASTIC/g" /opt/helioslove/logstash↩
    ↪ .conf
mkdir -p /data/elk
cp /opt/helioslove/logstash.conf /data/elk
cp /opt/helioslove/standard.yml /opt/tpot/etc/compose
cp /opt/helioslove/install.sh /opt/tpot/iso/installer/install.sh
sed -i 's/allow-hotplug/auto/g' /etc/network/interfaces
apt -y remove exim4-base
apt -y autoremove
adduser tsec --gecos ",,," --disabled-password
```

```
adduser tsec sudo
cd /opt/tpot/iso/installer
chmod +x install.sh
./install.sh $myADMIN_PASS
```

Anexo IX. Script Install.sh personalizado

Código 6: Bash. install.sh

```
#!/bin/bash
# T-Pot Universal Installer

# Installer can only be executed once.
myTPOT_INSTALL_LOG="/install.log"
if [ -s "$myTPOT_INSTALL_LOG" ];
  then
    echo "Aborting. Installer can only be executed once."
    exit
fi

###################
# I. Global vars #
###################

myBACKTITLE="T-Pot-Installer"
myLSB_STABLE_SUPPORTED="stretch buster"
myLSB_TESTING_SUPPORTED="stable"
myTPOTCOMPOSE="/opt/tpot/etc/tpot.yml"
myREMOTESITES="https://hub.docker.com https://github.com https://pypi.↩
    ↪ python.org https://debian.org https://listbot.sicherheitstacho.eu↩
    ↪ "
myINSTALLPACKAGES="aria2 apache2-utils apparmor apt-transport-https ↩
    ↪ aufs-tools bash-completion build-essential ca-certificates ↩
    ↪ cgroupfs-mount console-setup console-setup-linux cracklib-runtime↩
    ↪  curl debconf-utils dialog dnsutils docker.io docker-compose ↩
    ↪ ethtool fail2ban figlet genisoimage git glances grc haveged ↩
    ↪ html2text htop iptables iw jq kbd libcrack2 libltdl7 libpam-↩
```

```
    ↪ google-authenticator man mosh multitail net-tools npm ntp openssh↩
    ↪ -server openssl pass pigz prips software-properties-common ↩
    ↪ syslinux psmisc pv python3-pip toilet unattended-upgrades unzip ↩
    ↪ vim wget wireless-tools wpasupplicant"
myPREINSTALLPACKAGES="aria2 apache2-utils cracklib-runtime curl dialog ↩
    ↪ figlet fuse grc libcrack2 libpq-dev lsb-release net-tools ↩
    ↪ software-properties-common toilet"
mySITES="https://ghcr.io https://github.com https://pypi.python.org ↩
    ↪ https://debian.org"

myUPDATECHECK="APT::Periodic::Update-Package-Lists \"1\";
APT::Periodic::Download-Upgradeable-Packages \"0\";
APT::Periodic::AutocleanInterval \"7\";
"

mySYSCTLCONF="
# Reboot after kernel panic, check via /proc/sys/kernel/panic[_on_oops]
# Set required map count for ELK
kernel.panic = 1
kernel.panic_on_oops = 1
vm.max_map_count = 262144
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
"
myFAIL2BANCONF="[DEFAULT]
ignore-ip = 127.0.0.1/8
bantime = 3600
findtime = 600
maxretry = 5

[pam-generic]
enabled = true
port = 64294
filter = pam-generic
logpath = /var/log/auth.log

[sshd]
```

```
enabled = true
port = 64295
filter = sshd
logpath = /var/log/auth.log
"

mySYSTEMDFIX="[Link]
NamePolicy=kernel database onboard slot path
MACAddressPolicy=none
"

mySHELLCHECK='[[ $- == *i* ]] || return'
myROOTPROMPT='PS1="\[\033[38;5;8m\][\[$(tput sgr0)\]\[\033[38;5;1m\]\u↩
    ↪ \[$(tput sgr0)\]\[\033[38;5;6m\]@\[$(tput sgr0)\]\[\033[38;5;4m↩
    ↪ \]\h\[$(tput sgr0)\]\[\033[38;5;6m\]:\[$(tput sgr0)↩
    ↪ \]\[\033[38;5;5m\]\w\[$(tput sgr0)\]\[\033[38;5;8m\]]\[$(tput ↩
    ↪ sgr0)\]\[\033[38;5;1m\]\$\[$(tput sgr0)\]\[\033[38;5;15m\] \[$(↩
    ↪ tput sgr0)\]"'
myUSERPROMPT='PS1="\[\033[38;5;8m\][\[$(tput sgr0)\]\[\033[38;5;2m\]\u↩
    ↪ \[$(tput sgr0)\]\[\033[38;5;6m\]@\[$(tput sgr0)\]\[\033[38;5;4m↩
    ↪ \]\h\[$(tput sgr0)\]\[\033[38;5;6m\]:\[$(tput sgr0)↩
    ↪ \]\[\033[38;5;5m\]\w\[$(tput sgr0)\]\[\033[38;5;8m\]]\[$(tput ↩
    ↪ sgr0)\]\[\033[38;5;2m\]\$\[$(tput sgr0)\]\[\033[38;5;15m\] \[$(↩
    ↪ tput sgr0)\]"'
myROOTCOLORS="export LS_OPTIONS='--color=auto'

myRANDOM_HOUR=$(shuf -i 2-22 -n 1)
myRANDOM_MINUTE=$(shuf -i 0-59 -n 1)
myDEL_HOUR=$(($myRANDOM_HOUR+1))
myPULL_HOUR=$(($myRANDOM_HOUR-2))
myCRONJOBS="
# Check if updated images are available and download them
$myRANDOM_MINUTE $myPULL_HOUR * * * root docker-compose -f /opt/tpot/↩
    ↪ etc/tpot.yml pull

# Uploaded binaries are not supposed to be downloaded
*/1 * * * * root mv --backup=numbered /data/dionaea/roots/ftp/* /data/↩
    ↪ dionaea/binaries/
```

```
# Daily reboot
$myRANDOM_MINUTE $myRANDOM_HOUR * * 1-6 root systemctl stop tpot && ↩
    ↪ docker stop \$(docker ps -aq) || docker rm \$(docker ps -aq) || ↩
    ↪ reboot

# Check for updated packages every sunday, upgrade and reboot
$myRANDOM_MINUTE $myRANDOM_HOUR * * 0 root apt-fast autoclean -y && apt↩
    ↪ -fast autoremove -y && apt-fast update -y && apt-fast upgrade -y ↩
    ↪ && sleep 10 && reboot
"

mySSHPORT="
Port 64295
"

myINFO="\
##############################################
### T-Pot Installer for Debian (Stable) ###
##############################################

Disclaimer:
This script will install T-Pot on this system.
By running the script you know what you are doing:
1. SSH will be reconfigured to tcp/64295.
2. Please ensure other means of access to this system in case something↩
    ↪  goes wrong.
3. At best this script will be executed on the console instead through ↩
    ↪ a SSH session.

########################################

Usage:
       $0 --help - Help.

Example:
       $0 --type=user - Best option for most users."
```

```
##################
# II. Functions #
##################

# Create banners
function fuBANNER {
  toilet -f ivrit "$1"
}

# Do we have root?
function fuGOT_ROOT {
echo
echo -n "### Checking for root: "
if [ "$(whoami)" != "root" ];
  then
    echo "[ NOT OK ]"
    echo "### Please run as root."
    echo "### Example: sudo $0"
    exit
  else
    echo "[ OK ]"
fi
}

# Check for pre-installer package requirements.
# If not present install them
function fuCHECKPACKAGES {
  export DEBIAN_FRONTEND=noninteractive
  # Make sure dependencies for apt-fast are installed
  myCURL=$(which curl)
  myWGET=$(which wget)
  mySUDO=$(which sudo)
  if [ "$myCURL" == "" ] || [ "$myWGET" == "" ] || [ "$mySUDO" == "" ]
    then
      echo "### Installing deps for apt-fast"
      apt-get -y update
```

```
      apt-get -y install curl wget sudo
  fi
  echo "### Installing apt-fast"
  /bin/bash -c "$(curl -sL https://raw.githubusercontent.com/ilikenwf/↩
      ↪ apt-fast/master/quick-install.sh)"
  echo -n "### Checking for installer dependencies: "
  local myPACKAGES="$1"
  for myDEPS in $myPACKAGES;
    do
      myOK=$(dpkg -s $myDEPS 2>&1 | grep -w ok | awk '{ print $3 }' | ↩
          ↪ head -n 1)
      if [ "$myOK" != "ok" ];
        then
          echo "[ NOW INSTALLING ]"
          apt-fast update -y
          apt-fast install -y $myPACKAGES
          break
      fi
  done
  if [ "$myOK" = "ok" ];
    then
      echo "[ OK ]"
  fi
}

# Check for other services
function fuCHECK_PORTS {
    echo
    echo "### Checking for active services."
    echo
    grc netstat -tulpen
    echo
    echo "### Please review your running services."
    echo "### We will take care of SSH (22), but other services i.e. FTP↩
        ↪  (21), TELNET (23), SMTP (25), HTTP (80), HTTPS (443), etc."
    echo "### might collide with T-Pot's honeypots and prevent T-Pot ↩
        ↪ from starting successfully."
    echo
```

```
    while [ 1 != 2 ]
      do
        read -s -n 1 -p "Continue [y/n]? " mySELECT
    echo
        case "$mySELECT" in
          [y,Y])
            break
            ;;
          [n,N])
            exit
            ;;
        esac
      done
}

# Check if remote sites are available
function fuCHECKNET {
  local mySITES="$1"
  mySITESCOUNT=$(echo $mySITES | wc -w)
  j=0
  for i in $mySITES;
    do
      echo $(expr 100 \* $j / $mySITESCOUNT) | dialog --title "[ ↩
          ↪ Availability check ]" --backtitle "$myBACKTITLE" --gauge "\n↩
          ↪  Now checking: $i\n" 8 80
      curl --connect-timeout 30 -IsS $i 2>&1>/dev/null
      if [ $? -ne 0 ];
        then
          dialog --keep-window --backtitle "$myBACKTITLE" --title "[ ↩
              ↪ Continue? ]" --yesno "\nAvailability check failed. You ↩
              ↪ can continue, but the installation might fail." 10 50
          if [ $? = 1 ];
            then
              dialog --keep-window --backtitle "$myBACKTITLE" --title "[ ↩
                  ↪ Abort ]" --msgbox "\nInstallation aborted. Exiting ↩
                  ↪ the installer." 7 50
              exit
            else
```

```
            break;
          fi;
      fi;
    let j+=1
    echo $(expr 100 \* $j / $mySITESCOUNT) | dialog --keep-window --↩
        ↪ title "[ Availability check ]" --backtitle "$myBACKTITLE" --↩
        ↪ gauge "\n Now checking: $i\n" 8 80
  done;
}

# Let's load docker images
function fuPULLIMAGES {
for name in $(cat $myTPOTCOMPOSE | grep -v '#' | grep image | cut -d'"'↩
    ↪  -f2 | uniq)
  do
    docker pull $name
done
}

# Install T-Pot dependencies
function fuGET_DEPS {
  export DEBIAN_FRONTEND=noninteractive
  echo
  echo "### Getting update information."
  echo
  apt-fast -y update
  echo
  echo "### Upgrading packages."
  echo
  # Downlaod and upgrade packages, but silently keep existing configs
  echo "docker.io docker.io/restart boolean true" | debconf-set-↩
      ↪ selections -v
  echo "debconf debconf/frontend select noninteractive" | debconf-set-↩
      ↪ selections -v
  apt-fast -y dist-upgrade -o Dpkg::Options::="--force-confdef" -o Dpkg↩
      ↪ ::Options::="--force-confold" --force-yes
  echo
  echo "### Installing T-Pot dependencies."
```

```
  echo
  apt-fast -y install $myINSTALLPACKAGES
  # Remove exim4
  echo "### Removing and holding back problematic packages ..."
  apt-fast -y purge exim4-base mailutils pcp cockpit-pcp elasticsearch-curator
  apt-fast -y autoremove
  apt-mark hold exim4-base mailutils pcp cockpit-pcp elasticsearch-curator
}

############################
# III. Pre-Installer phase #
############################
fuGOT_ROOT
fuCHECKPACKAGES "$myPREINSTALLPACKAGES"

####################################
# IV. Prepare installer environment #
####################################

# Check for Debian release and extract command line arguments
myLSB=$(lsb_release -c | awk '{ print $2 }')
myVERSIONS="$myLSB_STABLE_SUPPORTED $myLSB_TESTING_SUPPORTED"
mySUPPORT="FALSE"
for i in $myVERSIONS
  do
    if [ "$myLSB" = "$i" ];
      then
        mySUPPORT="TRUE"
    fi
done
if [ "$mySUPPORT" = "FALSE" ];
  then
    echo "Aborting. Debian $myLSB is not supported."
    exit
fi
if [ "$1" == "" ];
```

```
  then
    echo "$myINFO"
    exit
fi


# Prepare running the installer
echo "$myINFO" | head -n 3
fuCHECK_PORTS


########################################
# V. Installer user interaction phase #
########################################

# Set TERM
export TERM=linux

# Check if remote sites are available
fuCHECKNET "$myREMOTESITES"

# Let's ask for a secure tsec password if installation type is iso
myCONF_TPOT_USER="tsec"
myPASS1="pass1"
myPASS2="pass2"
mySECURE="0"
while [ "$myPASS1" != "$myPASS2" ] && [ "$mySECURE" == "0" ]
  do
    while [ "$myPASS1" == "pass1" ] || [ "$myPASS1" == "" ]
      do
        myPASS1=$(dialog --keep-window --insecure --backtitle "↩
            ↪ $myBACKTITLE" \
                        --title "[ Enter password for console user (tsec) ↩
                            ↪ ]" \
                        --passwordbox "\nPassword" 9 60 3>&1 1>&2 2>&3 ↩
                            ↪ 3>&-)
      done
        myPASS2=$(dialog --keep-window --insecure --backtitle "↩
```

```
            ↪ $myBACKTITLE" \
                        --title "[ Repeat password for console user (tsec)↩
                            ↪  ]" \
                        --passwordbox "\nPassword" 9 60 3>&1 1>&2 2>&3 ↩
                            ↪ 3>&-)
    if [ "$myPASS1" != "$myPASS2" ];
      then
        dialog --keep-window --backtitle "$myBACKTITLE" --title "[ ↩
            ↪ Passwords do not match. ]" \
              --msgbox "\nPlease re-enter your password." 7 60
        myPASS1="pass1"
        myPASS2="pass2"
    fi
    mySECURE=$(printf "%s" "$myPASS1" | cracklib-check | grep -c "OK")
    if [ "$mySECURE" == "0" ] && [ "$myPASS1" == "$myPASS2" ];
      then
        dialog --keep-window --backtitle "$myBACKTITLE" --title "[ ↩
            ↪ Password is not secure ]" --defaultno --yesno "\nKeep ↩
            ↪ insecure password?" 7 50
        myOK=$?
        if [ "$myOK" == "1" ];
          then
            myPASS1="pass1"
            myPASS2="pass2"
        fi
    fi
  done
printf "%s" "$myCONF_TPOT_USER:$myPASS1" | chpasswd

dialog --clear

############################
# VI. Installation phase #
############################

exec 2> >(tee "/install.err")
exec > >(tee "/install.log")
```

```
fuBANNER "Installing ..."

fuGET_DEPS

# Let's make sure SSH roaming is turned off (CVE-2016-0777, CVE-2016-0778)
fuBANNER "SSH roaming off"
echo "UseRoaming no" | tee -a /etc/ssh/ssh_config

# Let's create the T-Pot user
fuBANNER "Create user"
addgroup --gid 2000 tpot
adduser --system --no-create-home --uid 2000 --disabled-password --disabled-login --gid 2000 tpot

# Let's set the hostname
a=$(fuRANDOMWORD /opt/tpot/host/usr/share/dict/a.txt)
n=$(fuRANDOMWORD /opt/tpot/host/usr/share/dict/n.txt)
myHOST=$a$n
fuBANNER "Set hostname"
hostnamectl set-hostname $myHOST
sed -i 's#127.0.1.1.*#127.0.1.1\t'"$myHOST"'#g' /etc/hosts

# Let's patch sshd_config
fuBANNER "Adjust ports"
sed -i '/^port/Id' /etc/ssh/sshd_config
echo "$mySSHPORT" | tee -a /etc/ssh/sshd_config

# Let's download images and started
fuBANNER "HELIOSLOVE"
ln -s /opt/tpot/etc/compose/standard.yml $myTPOTCOMPOSE


# Let's load docker images
fuBANNER "Pull images"
fuPULLIMAGES

# Let's add the daily update check with a weekly clean interval
```

```
fuBANNER "Modify checks"
echo "$myUPDATECHECK" | tee /etc/apt/apt.conf.d/10periodic

# Let's make sure to reboot the system after a kernel panic
fuBANNER "Tweak sysctl"
echo "$mySYSCTLCONF" | tee -a /etc/sysctl.conf

# Let's setup fail2ban config
fuBANNER "Setup fail2ban"
echo "$myFAIL2BANCONF" | tee /etc/fail2ban/jail.d/tpot.conf

# Fix systemd error https://github.com/systemd/systemd/issues/3374
fuBANNER "Systemd fix"
echo "$mySYSTEMDFIX" | tee /etc/systemd/network/99-default.link

# Let's add some cronjobs
fuBANNER "Add cronjobs"
echo "$myCRONJOBS" | tee -a /etc/crontab

# Let's create some files and folders
fuBANNER "Files & folders"
mkdir -vp /data/adbhoney/{downloads,log} \
        /data/ciscoasa/log \
        /data/conpot/log \
        /data/citrixhoneypot/logs \
        /data/cowrie/{downloads,keys,misc,log,log/tty} \
          /data/dicompot/{images,log} \
        /data/dionaea/{log,bistreams,binaries,rtp,roots,roots/ftp,roots/↩
            ↪ tftp,roots/www,roots/upnp} \
        /data/elasticpot/log \
        /data/elk/{data,log} \
        /data/fatt/log \
        /data/honeytrap/{log,attacks,downloads} \
        /data/glutton/log \
        /data/heralding/log \
        /data/honeypy/log \
        /data/honeysap/log \
          /data/ipphoney/log \
```

```
        /data/mailoney/log \
        /data/medpot/log \
        /data/nginx/{log,heimdall} \
        /data/emobility/log \
        /data/ews/conf \
        /data/rdpy/log \
        /data/spiderfoot \
        /data/suricata/log \
        /data/tanner/{log,files} \
        /data/p0f/log \
        /home/tsec/.ssh/

# Let's copy some files
fuBANNER "Copy configs"
cp /opt/tpot/host/etc/systemd/* /etc/systemd/system/
systemctl enable tpot

# Let's take care of some files and permissions
fuBANNER "Permissions"
chmod 770 -R /data
usermod -a -G tpot tsec
chown tsec:tsec -R /home/tsec/.ssh
chown tpot:tpot -R /data

# Let's replace "quiet splash" options, set a console font for more ↩
    ↪ screen canvas and update grub
fuBANNER "Options"
sed -i 's#GRUB_CMDLINE_LINUX_DEFAULT="quiet"#GRUB_CMDLINE_LINUX_DEFAULT↩
    ↪ ="quiet consoleblank=0"#' /etc/default/grub
sed -i 's#GRUB_CMDLINE_LINUX=""#GRUB_CMDLINE_LINUX="cgroup_enable=↩
    ↪ memory swapaccount=1"#' /etc/default/grub
update-grub

fuBANNER "Setup console"
cp /usr/share/consolefonts/Uni2-Terminus12x6.psf.gz /etc/console-setup/
gunzip /etc/console-setup/Uni2-Terminus12x6.psf.gz
sed -i 's#FONTFACE=".*#FONTFACE="Terminus"#' /etc/default/console-setup
sed -i 's#FONTSIZE=".*#FONTSIZE="12x6"#' /etc/default/console-setup
```

```
update-initramfs -u
sed -i 's#After=.*#After=systemd-tmpfiles-setup.service console-screen.↩
    ↪ service kbd.service local-fs.target#' /etc/systemd/system/multi-↩
    ↪ user.target.wants/console-setup.service

# Let's enable a color prompt and add /opt/tpot/bin to path
fuBANNER "Setup prompt"
tee -a /root/.bashrc <<EOF
$mySHELLCHECK
$myROOTPROMPT
$myROOTCOLORS
PATH="$PATH:/opt/tpot/bin"
EOF
for i in $(ls -d /home/*/)
  do
tee -a $i.bashrc <<EOF
$mySHELLCHECK
$myUSERPROMPT
PATH="$PATH:/opt/tpot/bin"
EOF
done

# Let's create ews.ip before reboot and prevent race condition for ↩
    ↪ first start
fuBANNER "Update IP"
/opt/tpot/bin/updateip.sh

# Let's clean up apt
fuBANNER "Clean up"
apt-fast autoclean -y
apt-fast autoremove -y

# Final steps
cp /opt/tpot/host/etc/rc.local /etc/rc.local && \
rm -rf /root/installer && \
rm -rf /etc/issue.d/cockpit.issue && \
rm -rf /etc/motd.d/cockpit && \
rm -rf /etc/issue.net && \
```

```
rm -rf /etc/motd && \
systemctl restart console-setup.service
fuBANNER "Rebooting ..."
sleep 2
reboot
```

Anexo X. Servicio de T-Pot

```
[Unit]
Description=tpot
Requires=docker.service
After=docker.service

[Service]
Restart=always
RestartSec=5
TimeoutSec=infinity

# Get and set internal, external IP infos, but ignore errors
ExecStartPre=-/opt/tpot/bin/updateip.sh

# Clear state or if persistence is enabled rotate and compress logs ↩
    ↪ from /data
ExecStartPre=-/bin/bash -c '/opt/tpot/bin/clean.sh on'

# Remove old containers, images and volumes
ExecStartPre=-/usr/bin/docker-compose -f /opt/tpot/etc/tpot.yml down -v
ExecStartPre=-/usr/bin/docker-compose -f /opt/tpot/etc/tpot.yml rm -v
ExecStartPre=-/bin/bash -c 'docker network rm $(docker network ls -q)'
ExecStartPre=-/bin/bash -c 'docker volume rm $(docker volume ls -q)'
ExecStartPre=-/bin/bash -c 'docker rm -v $(docker ps -aq)'
ExecStartPre=-/bin/bash -c 'docker rmi $(docker images | grep "<none>" ↩
    ↪ | awk \'{print $3}\')'

# Get IF, disable offloading, enable promiscious mode for p0f and ↩
    ↪ suricata
ExecStartPre=-/bin/bash -c '/sbin/ethtool --offload $(/sbin/ip address ↩
```

```
    ↪ | grep "^2: " | awk \'{ print $2 }\' | tr -d [:punct:]) rx off tx↩
    ↪  off'
ExecStartPre=/bin/bash -c '/sbin/ethtool -K $(/sbin/ip address | grep ↩
    ↪ "^2: " | awk \'{ print $2 }\' | tr -d [:punct:]) gso off gro off'
ExecStartPre=/bin/bash -c '/sbin/ip link set $(/sbin/ip address | grep ↩
    ↪ "^2: " | awk \'{ print $2 }\' | tr -d [:punct:]) promisc on'

# Set iptables accept rules to avoid forwarding to honeytrap / NFQUEUE
# Forward all other connections to honeytrap / NFQUEUE
ExecStartPre=/opt/tpot/bin/rules.sh /opt/tpot/etc/tpot.yml set

# Compose T-Pot up
ExecStart=/usr/bin/docker-compose -f /opt/tpot/etc/tpot.yml up --no-↩
    ↪ color

# Compose T-Pot down, remove containers and volumes
ExecStop=/usr/bin/docker-compose -f /opt/tpot/etc/tpot.yml down -v

# Remove only previously set iptables rules
ExecStopPost=/opt/tpot/bin/rules.sh /opt/tpot/etc/tpot.yml unset

[Install]
WantedBy=multi-user.target
```

Anexo XI. Configuración para el contenedor de Logstash

Código 7: YAML. standard.yml - Logstash

```
## Logstash service

  logstash:
    container_name: logstash
    restart: always
    environment:
     - LS_JAVA_OPTS=-Xms2048m -Xmx2048m
    env_file:
     - /opt/tpot/etc/compose/elk_environment
    image: "dtagdevsec/logstash:2006"
    build:
      context: /opt/tpot/docker/elk/logstash
      dockerfile: /opt/tpot/docker/elk/logstash/Dockerfile
    volumes:
     - /data:/data
     - /data/elk/logstash.conf:/etc/logstash/conf.d/logstash.conf
```

Anexo XII. Instalación de T-Pot en un VPS

1. Actualización de la lista de paquetes disponibles y sus versiones.

```
helios@VPS:~# sudo apt update
```

2. Instalación de los paquetes apache2-utils, cracklib-runtime, curl, dialog, figlet, git, grc, libcrack2, libpq-dev, lsb-release, net-tools, software-properties-common, grub-pc y toilet.

```
helios@VPS:~# sudo apt install apache2-utils cracklib-runtime ↩
    ↪ curl dialog figlet git grc libcrack2 libpq-dev lsb-↩
    ↪ release net-tools software-properties-common grub-pc ↩
    ↪ toilet
```

3. Actualización del GRUB.

```
helios@VPS:~# sudo update-grub
```

4. Cambio de los nombre lógicos de las interfaces de red.

```
helios@VPS:~# sudo sed -i 's/allow-hotplug/auto/g' /etc/↩
    ↪ network/interfaces
```

5. Descarga del repositorio de GitHub en /opt/tpot.

```
helios@VPS:~# sudo git clone --depth=1 https://github.com/↩
    ↪ telekom-security/tpotce /opt/tpot
```

6. Acceso a la carpeta donde se encuentra el ejecutable de instalación /opt/tpot/iso/installer.

```
helios@VPS:~# cd /opt/tpot/iso/installer
```

7. Ejecución del script de instalación y paso de parámetro *–type=user*.

```
helios@VPS:~# sudo ./install.sh --type=user
```

Anexo XIII. Catálogo de reputación de direcciones IP en T-Pot

T-Pot consulta un fichero YAML, descargado cada 24 horas en el Docker Logstash de `https://listbot.sicherheitstacho.eu/iprep.yaml.bz2`, para catalogar la reputación de cada dirección IP. El fichero es generado y actualizado por una herramienta llamada **Listbot** (desarrollada también por Deutsche Telekom) que consulta bases de datos de fuentes relevantes, como las que se muestran a continuación:

- `https://reputation.alienvault.com/reputation.generic`
- `https://raw.githubusercontent.com/Neo23x0/signature-base/39787aaefa6b70b0be6e7dcdc425 iocs/otx-c2-iocs.txt`
- `https://www.badips.com/get/list/any/2?age=90d`
- `http://osint.bambenekconsulting.com/feeds/c2-ipmasterlist.txt`
- `https://lists.blocklist.de/lists/all.txt`
- `https://iplists.firehol.org/files/bitcoin_nodes_30d.ipset`
- `https://iplists.firehol.org/files/botscout_30d.ipset`
- `https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/cruzit _web_attacks.ipset`
- `https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/malwaredomainlist .ipset`
- `https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/proxylists _30d.ipset`
- `https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/proxyrss _30d.ipset`
- `https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/proxyspy _30d.ipset`

- https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/ri_web_proxies_30d.ipset
- https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/socks_proxy_30d.ipset
- https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/sslproxies_30d.ipset
- https://iplists.firehol.org/files/cleantalk_30d.ipset
- https://iplists.firehol.org/files/dshield_30d.netset
- https://iplists.firehol.org/files/darklist_de.netset
- https://iplists.firehol.org/files/dm_tor.ipset
- http://danger.rulez.sk/projects/bruteforceblocker/blist.php
- http://cinsscore.com/list/ci-badguys.txt
- https://feodotracker.abuse.ch/blocklist/?download=ipblocklist
- https://rules.emergingthreats.net/open/suricata/rules/compromised-ips.txt
- http://blocklist.greensnow.co/greensnow.txt
- http://www.nothink.org/blacklist/blacklist_malware_irc.txt
- http://spys.me/proxy.txt
- http://ransomwaretracker.abuse.ch/downloads/RW_IPBL.txt
- https://report.cs.rutgers.edu/DROP/attackers
- http://sblam.com/blacklist.txt
- https://sslbl.abuse.ch/blacklist/sslipblacklist.csv
- http://www.talosintelligence.com/feeds/ip-filter.blf
- https://check.torproject.org/exit-addresses
- https://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv
- https://www.turris.cz/greylist-data/greylist-latest.csv
- https://zeustracker.abuse.ch/blocklist.php?download=badips

- https://raw.githubusercontent.com/stamparm/maltrail/master/trails/static/mass_scanner.txt
- https://myip.ms/files/blacklist/general/full_blacklist_database.zip
- http://www.dnsbl.manitu.net/download/nixspam-ip.dump.gz
- http://www.urlvir.com/export-ip-addresses/
- https://threatintel.stdominics.sa.edu.au/droplist_high_confidence.txt
- https://sslbl.abuse.ch/blacklist/dyre_sslipblacklist_aggressive.csv
- http://charles.the-haleys.org/ssh_dico_attack_hdeny_format.php/hostsdeny.txt
- https://zerodot1.gitlab.io/CoinBlockerLists/MiningServerIPList.txt
- http://www.botvrij.eu/data/ioclist.ip-dst.raw
- http://www.ipspamlist.com/public_feeds.csv