Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Grado en Ciberseguridad

Penetración dinámica de redes locales con Skynet

Trabajo fin de estudio presentado por:	Pol Blanco Guillén
Tipo de trabajo:	Desarrollo de software
Director/a:	Pedro Diaz García
Fecha:	15/07/2026

Resumen

En la actualidad, se dispone de una amplia variedad de herramientas de pentest con capacidades avanzadas de análisis de redes, explotación de vulnerabilidades, simulación de ataques, así como plataformas para la orquestación de simulacros de red team. Sin embargo, buena parte de estas herramientas están orientadas a entornos estáticos, principalmente empresariales, donde no hay cambios constantes como conexiones y desconexiones de dispositivos, y existe una topología estable.

Este trabajo presenta Skynet, una propuesta de herramienta basada en una arquitectura modular con automatización parcial enfocada en redes LAN dinámicas y heterogéneas. La arquitectura se basa en un núcleo encargado de gestionar la información contextual, los eventos, y acciones en ejecución, y unos módulos complementarios de ataques y servicios, diseñados para facilitar la incorporación de nuevos ataques y funcionalidades. Los ataques incorporan metadatos alineados con el MITRE ATT&CK para proporcionar la información contextual durante la ejecución, que se puede combinar con la automatización asistida basada en LLMs.

Skynet se evalúa utilizando tres escenarios de laboratorio desarrollados en GNS3, cada uno representativo de un entorno distinto, con distintos niveles de complejidad y protección. Los resultados de cada escenario sirven para evaluar el comportamiento de la arquitectura propuesta, y permiten evaluar objetivamente el comportamiento de la arquitectura propuesta y determinar sus capacidades de integración, extensibilidad y automatización.

Palabras clave: pentesting, redes LAN, automatización, arquitectura modular, MITRE ATT&CK

Abstract

There is currently a wide variety of penetration testing tools available, offering advanced capabilities for network analysis, vulnerability exploitation and attack simulation, as well as platforms for orchestrating red team exercises. However, many of these tools are designed for static environments – primarily corporate ones – where there are no constant changes, such as devices being connected and disconnected, and where the topology remains stable.

This work presents Skynet, a proposed tool based on a modular architecture with partial automation, designed for dynamic and heterogeneous LANs. The architecture is based on a core responsible for managing contextual information, events and actions currently being executed, alongside complementary attack and service modules, designed to facilitate the incorporation of new attacks and functionalities. The attacks incorporate metadata aligned with the MITRE ATT&CK framework to provide contextual information during execution, which can be combined with LLM-based assisted automation.

Skynet is evaluated using three laboratory scenarios developed in GNS3, each representing a different environment, with varying levels of complexity and security. The results from each scenario serve to assess the behaviour of the proposed architecture, and enable an objective evaluation of its behaviour and a determination of its integration, extensibility and automation capabilities.

Keywords: penetration testing, LAN, Cyber Kill Chain, modular architecture, MITRE ATT&CK.

Índice de contenidos

Introducción....10
Motivación....10
Planteamiento del problema....12
Estructura del trabajo....12
Contexto y Estado del arte....14
Estado del arte defensivo....14
Ofensivo....22
Comparación de la propuesta con herramientas representativas del estado del arte....30
Conclusiones del estado del arte....32
Objetivos concretos y metodología de trabajo....33
Objetivo general....33
Objetivos específicos....34
Objetivos específicos de desarrollo....34
Objetivos específicos de evaluación....34
Metodología del trabajo....35
EVALUACIÓN Y ANÁLISIS DE LOS RESULTADOS....36
Limitaciones del estudio....37
Desarrollo específico de la contribución....38

Requisitos del software....38
Requisitos funcionales....38
Requisitos no funcionales....39
Restricciones de diseño....39
Descripción y funcionalidades de Skynet....40
Arquitectura....40
Ampliación de Skynet....42
Funcionalidades principales....43
Configuración e inicialización....46
Evaluación....47
Diseño de los escenarios de evaluación....47
Evaluación de Skynet....63
Conclusiones y trabajo futuro....81
Referencias bibliográficas....84

Índice de figuras

Figura 1: Diagrama representativo de la arquitectura.....40

Figura 2: Terminal de Skynet en ejecución de un ataque pasivo.....44

Figura 3: Terminal de Skynet operando servicios.....44

Figura 4: Skynet en modo "stealth".....46

Figura 5: Skynet visualizándose en ejecución en el terminal de Windows 11.....47

Figura 6: Escenario fácil cargado en GNS3.....48

Figura 7: Configuración del Router del escenario fácil.....49

Figura 8: Configuración del Router correspondiente al ISP, del escenario fácil.....50

Figura 9: Configuración del Router-1 del escenario fácil.....50

Figura 10: Configuración del Router-2 del escenario fácil.....50

Figura 11: Configuración de red del DNS del escenario fácil.....51

Figura 12: Archivo de zona del DNS del escenario fácil.....51

Figura 13: Test de conectividad desde el cliente Ubuntu.....52

Figura 14: Escenario medio cargado en GNS3.....53

Figura 15: Configuración del switch central del escenario medio.....54

Figura 16: Configuración del switch del departamento médico del escenario medio.....55

Figura 17: Configuración del switch del departamento de administración del escenario medio.....55

Figura 18: Configuración del switch del departamento de servidores del escenario medio 56

Figura 19: Configuración del Router del hospital.....57

Figura 20: Test de conectividad a la WAN y vista general del laboratorio medio.....58

Figura 21: Test de conectividad al internet simulado (imagen ampliada).....58

Figura 22: Test de segmentación de VLANs y vista general del laboratorio medio.....59

Figura 23: Test de conectividad entre la VLAN de médicos y administración.....60

Figura 24: Test de conectividad entre la VLAN de médicos y la base de datos.....60

Figura 25: Test de conectividad entre la VLAN de servidores y la VLAN de invitados.....61

Figura 26: Configuración de red de un cliente de la red de inivitados (UbuntuClient).....61

Figura 27: Cliente ubuntu conectado al captive portal.....62

Figura 28: Escenario difícil cargado en GNS3.....63

Figura 29: Skynet realizando un ataque DHCP starvation.....65

Figura 30: Skynet realizando un ataque DHCP starvation (imagen ampliada).....65

Figura 31: Leases activas en el DHCP del Router parte 1.....66

Figura 32: Leases activas en el DHCP del Router parte 2.....67

Figura 33: Skynet levantando el servicio de dhcpresponder.....67

Figura 34: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de dhcpresponder.....68

Figura 35: Skynet levantando el servicio de dnspresponder.....68

Figura 36: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de dnsresponder.....68

Figura 37: Skynet levantando el servicio de httpfileresponder.....68

Figura 38: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de httpfileresponder.....69

Figura 39: Estado actual de la red en el escenario fácil.....69

Figura 40: UbuntuClient ejecutando la bindshell involuntariamente.....70

Figura 41: Skynet testeando la conexión con la bindshell de UbuntuClient.....70

Figura 42: Skynet realizando un ataque de reconocimiento pasivo.....73

Figura 43: Terminal de Skynet iniciando servicios.....74

Figura 44: Event bus de Skynet mostrando los mensajes de los servicios iniciados.....74

Figura 45: Skynet intentando el ataque de DHCP starvation.....74

Figura 46: Inicio del servicio de DHCP malicioso.....75

Figura 47: Terminal de Skynet mostrando información de reconocimiento.....75

Figura 48: Terminal de ubuntu del atacante tratando de acceder a internet.....75

Figura 49: Mensajes del servicio de proxy de Skynet al ser iniciado.....76

Figura 50: Comando de inicio del proxy de Skynet.....76

Figura 51: Terminal de ubuntu del atacante accediendo al servidor web.....76

Figura 52: Intento de DHCP starvation mediante dhcp discover.....79

Índice de tablas

Tabla 1: Tabla comparativa del estado del arte ofensivo y la herramienta propuesta....31

Tabla 2: Métricas de evaluación de la arquitectura propuesta.....36

Tabla 3: Topología lógica de red del escenario de dificultad media.....53

Tabla 4: Resultados de Skynet en la prueba del escenario de seguridad baja.....70

Tabla 5: Resultados de Skynet en la prueba del escenario de seguridad media.....77

Tabla 6: Resultados de Skynet en la prueba del escenario de seguridad alta.....79

1. Introducción

El objetivo principal de este estudio es proponer una arquitectura orientada a la automatización parcial de tareas repetitivas durante la ejecución de ejercicios de pentesting, como la gestión del contexto entre las distintas fases y la coordinación y sinergia de ataques, que permita a la herramienta adaptarse a distintos contextos, y que puedan experimentar cambios continuos durante su funcionamiento.

Para lograr ese objetivo, se ha empleado un diseño modular, donde existe fundamentalmente un núcleo encargado de la inicialización de la herramienta y gestión del contexto, módulos destinados a la definición y extensión de ataques y servicios, un módulo para definir la interfaz gráfica, y otros módulos auxiliares para realizar tareas complementarias como logging o automatización.

Para lograr una gestión del contexto acorde a las técnicas de pentest actuales, se ha optado por utilizar metadatos en cada ataque alineados con el MITRE ATT&CK y la Cyber Kill Chain, lo que además favorece la trazabilidad y la reproducibilidad de las acciones ejecutadas, además de facilitar posibles desarrollos futuros alineados con las tendencias de la industria.

Para evaluar la efectividad de la propuesta, se le va a enfrentar a varios escenarios simulados, basados en el mundo real, que serán: un restaurante con Wi-Fi (seguridad básica), un hospital con captive portal (seguridad media), una planta industrial SCADA (seguridad alta). Con el fin de favorecer la reproducibilidad del trabajo, se han publicado los escenarios en este enlace, y el código en el siguiente repositorio público de GitHub: enlace al repositorio.

1.1. Motivación

La continua evolución de las infraestructuras tecnológicas incrementa la superficie de ataque y favorece la posibilidad de aparición de nuevas vulnerabilidades, lo que, añadido al constante flujo de aparición de nuevas amenazas, convierte la capacidad de adaptación en un aspecto relevante dentro de la ciberseguridad y la ciberresiliencia. En este contexto, disponer de información contextual y de una arquitectura que facilite la extensión de las

capacidades ofensivas puede mejorar la toma de decisiones informada y la eficiencia durante las evaluaciones de seguridad.

En los últimos años, se ha incrementado la incorporación de modelos de lenguaje (LLM) a herramientas y conjuntos de herramientas (mediante MCPs) de ciberseguridad. Sin embargo, la coordinación entre las distintas técnicas de ataque y herramientas, y el mantenimiento de un contexto compartido durante todo el ejercicio de pentesting, continúa planteando desafíos importantes, especialmente en entornos dinámicos donde los activos pueden variar durante la ejecución.

En ingeniería del software es una práctica ampliamente aceptada integrar los requisitos más importantes desde las primeras fases del ciclo de desarrollo. Tomando el ejemplo de los S-SDLC (ciclos de desarrollo del software seguro), se han convertido en el estándar de desarrollo en la industria por la demostración práctica de la veracidad de su principio: si se concibe la seguridad como un requisito desde el inicio, se reducen drásticamente las vulnerabilidades que presenta un software. El mismo principio es aplicable a la integración. Si la integración se concibe como un requisito ya desde las primeras fases de desarrollo, la arquitectura del software y su funcionalidad no presentará problemas para su futura integración con otras herramientas, mejorando la eficacia y coordinación de los ataques.

A partir de estas observaciones, el presente trabajo propone una arquitectura modular orientada a la integración de los módulos de ataque y servicio, y la automatización de determinadas tareas auxiliares del pentesting.

El objetivo principal es mantener información contextual durante toda la ejecución del pentest, simplificar la incorporación de nuevas capacidades ofensivas, y reducir parcialmente la complejidad operativa asociada a la coordinación entre herramientas.

1.2. Planteamiento del problema

A pesar de la gran variedad de herramientas de pentesting, tanto especializadas en determinadas tareas como las que abarcan varios aspectos del pentest, en muchos casos se sigue dependiendo de mecanismos específicos, no estandarizados, para acceder desde una herramienta al estado o información interna de otra si la última no fue diseñada teniendo en cuenta ese requisito.

Lo que puede llevar a numerosos problemas en escenarios complejos, ya sea a la hora de compartir el contexto de los ataques y decidir las próximas acciones ofensivas, evadir defensas camuflando el ataque con tráfico legítimo, o la falta de posibilidades de realizar una automatización sencilla, coherente y adaptativa.

En este trabajo se desarrolla un framework que aborda cada limitación con una solución concebida desde las primeras fases de su desarrollo, es decir, desde el propio diseño de la arquitectura: un núcleo encargado de la gestión del contexto compartido, un motor de ataques alineado con la Cyber Kill Chain y el MITRE ATT&CK, exposición de mecanismos para facilitar la integración con otras herramientas, plantillas para la inclusión de nuevos ataques y servicios, y soporte opcional para la automatización asistida mediante LLM.

1.3. Estructura del trabajo

El presente trabajo se organiza en cinco capítulos.

En el **capítulo 2** se presenta el estado del arte, estructurado en 2 perspectivas: defensiva y ofensiva. Donde a nivel defensivo se analizan los protocolos que forman la infraestructura de red TCP/IP, sus limitaciones y mitigaciones con herramientas de seguridad modernas, y por otra parte se trata su contrario, el estado del arte ofensivo, donde se analizan las principales herramientas, metodologías y tecnologías relacionadas con el pentesting automatizado, la gestión del contexto y la automatización en ciberseguridad.

El **capítulo 3** define los objetivos generales y específicos del trabajo, así como la metodología seguida para el desarrollo de la propuesta, las herramientas empleadas y el procedimiento de evaluación.

El **capítulo 4** describe la contribución desarrollada. En él se presentan los requisitos de diseño, la arquitectura de la herramienta, las principales decisiones de diseño e implementación y la evaluación experimental realizada mediante los escenarios de laboratorio definidos.

Finalmente, el **capítulo 5** recoge las conclusiones del trabajo, analizando el grado de cumplimiento de los objetivos planteados, las principales aportaciones de la propuesta, sus limitaciones detectadas y las posibles líneas de trabajo futuro.

2. Contexto y Estado del arte

El estado del arte se estructura en dos perspectivas complementarias. En primer lugar, se inicia analizando la evolución de los protocolos, herramientas y tecnologías defensivas que caracterizan a las infraestructuras de red actuales. Posteriormente se realiza un análisis sobre las metodologías, herramientas y marcos de referencia de ámbito ofensivo. Prestando siempre especial atención a las tendencias que están marcando la actualidad. Para finalizar, se presenta una comparación entre lo expuesto anteriormente y la propuesta desarrollada en este trabajo, contextualizando su alcance dentro del estado del arte actual.

2.1. ESTADO DEL ARTE DEFENSIVO

Como adecuadamente señalaron M. Ambrosin *et al* en su paper "Security and Privacy Analysis of NSF Future Internet Architectures", "IP was originally designed for a small and relatively amicable research community.", es decir, la base de internet se pensó para ser utilizada en entornos controlados como universidades o entes gubernamentales, no masivamente, y por ello la base de internet carece de seguridad.

Por su parte, CAIDA en su investigación sobre internet "The (un)Economic Internet?", nos explica por qué esto es así: "Since the network itself does not contain security support, the onus has fallen to the people managing individual computers connected to the Internet, to the network operators to protect Internet connected hosts and servers, and to the operators of Internet service providers to protect their routers and other infrastructure services. Since services such as user or end system authentication, data integrity verification and encryption were not built into the core Internet protocols, they are now layered on an infrastructure that is not intrinsically secure." Es decir, la seguridad ha venido después, cuando se han ido añadiendo extensiones y protocolos seguros sobre la base de internet, una vez este se ha expandido y muchas personas ya han sufrido las consecuencias.

Como consecuencia de esta evolución, el estado del arte defensivo está constituido por la combinación de dos grandes bloques tecnológicos. Por una parte, los protocolos y estándares de red correspondientes a la arquitectura TCP/IP que sirven como los cimientos

de toda red, y, por otra parte, las plataformas y herramientas cuya función es supervisar y defender continuamente la infraestructura, detectando comportamientos anómalos y coordinando la respuesta frente a incidentes. Ambos elementos están íntimamente interrelacionados constituyendo la base de las redes seguras en la actualidad.

A continuación, se analizan los mecanismos de protección más elementales incorporados en cada capa del modelo TCP/IP. Pese a que cada protocolo resuelve unos problemas de seguridad específicos, se aprecia la tendencia a integrar de base la seguridad a nivel de protocolo, y no relegando la seguridad a implementaciones no estandarizadas:

- Capa de acceso a la red
    - IEEE 802.1X (RFC 3748): proporciona autenticación en la capa de acceso a la red.
    - DHCP Snooping: permite filtrar por DHCPs legítimos y evitar que se levanten DHCPs no autorizados. Esta función no tiene un RFC o autor determinados, pese a que fue popularizado por los fabricantes como Cisco en la década de los 2000.
    - Dynamic ARP Inspection (DAI): valida las respuestas ARP para filtrar el tráfico malicioso y evitar ataques MITM.
    - VLAN Security / Anti-trunking: hace referencia al fortalecimiento de los switches contra ataques cambios de puerto a trunk para obtener tráfico de otras VLANs.
    - VXLAN (o VLAN extensible, RFC 7348): proporciona la capacidad de extender VLANs sobre UDP. Muy utilizado en cloud. Mejora mucho la escalabilidad sobre redes de capa 2, y mejora el aislamiento y adaptabilidad a los cambios en redes, lo que aplicando microsegmentación y políticas dinámicas hace mejorar notablemente la seguridad en redes.
    - Ethernet VPN (RFC 7432): normalmente se utiliza para que haga de base a la VXLAN (juntos forman el RFC 8365), y de esta forma los paquetes viajen por

caminos mejor definidos y evitar el tráfico broadcast o unknown unicast flooding.

- Port Security: es una protección en la que solo se permite ciertas MACs determinadas en determinados puertos asegurando que solo los dispositivos autorizados se conecten a los puertos correspondientes.
- Protecciones STP
  - BPDU Guard: bloquea BPDUs en puertos de usuario, para prevenir que usuarios maliciosos se conviertan en el switch principal (root)
  - Root Guard: evita que un switch malicioso se pueda convertir en bridge
  - Loop Guard: evita bucles de tráfico
- En conjunto, los protocolos y tecnologías revisadas muestran una tendencia clara hacia la segmentación de red, autenticación de dispositivos basada en MAC, y protecciones básicas contra escalado de privilegios y denegaciones de servicio tanto intencionales como fortuitas.

- Capa de Internet
  - IPsec (RFC 6071): proporciona encriptación y autenticación sobre el protocolo IP (RFC 791)
  - IP Ingress/Egress filtering (RFC 2827 y RFC 3704): bloquea el tráfico IP que tiene direcciones no acordes a la red por la que viaja. Como por ejemplo un paquete con origen 8.8.8.8 que salga de una red 10.0.0.0/24.
  - Debido a que la mayoría de tráfico en internet está orientado a conexión, ya sea mediante TCP o UDP, la capa de internet, con su protocolo principal (IP), presenta unas funcionalidades de seguridad básicas como IP filtering, sin embargo, permite conseguir un nivel de seguridad más alto con IPsec,

cumpliendo completamente con la triada CIA (confidencialidad, integridad y autenticidad).

- Capa de transporte
    - TLS (RFC 9846): añade encriptación y autenticación sobre TCP (RFC 9293). Ha sido muy utilizado y es muy utilizado para crear la variante segura de muchos protocolos de capa de aplicación, como HTTPS. Actualmente lo último es el empleo de criptografía híbrida (moderna + post cuántica) para evitar ataques "harvest now, decrypt later".
    - TCP/UDP/SCTP hardening: se trata de configurar los parámetros de los protocolos en las distintas máquinas expuestas para mitigar sobre todo los ataques de denegación de servicio, ajustando rate limits, usando syn cookies, caché, proxies intermedios que controlen estos protocolos y no hagan gastar recursos al servidor hasta que sea una conexión legítima. TCP (RFC 4987), UDP y SCTP tienen hardenings posibles distintos.
    - Las tecnologías de defensa y funcionalidad de la capa de transporte constituyen una de las más importantes barreras defensivas de internet, pues es la capa sobre la que opera el protocolo de cifrado más utilizado en internet, TLS, muy utilizado para proteger los protocolos de la capa de aplicación. Además, es en esta capa donde se definen muchos campos utilizados por los firewalls con DPI, o los NIDS.
- Capa de aplicación
    - HSTS (RFC 6797): obliga al navegador web a conectarse siempre por HTTPS, evitando que, por descuido o acciones maliciosas de un tercero, una víctima se conecte solamente por HTTP (es decir, sin cifrado ni autenticación).
    - SSH (RFC 4251): permite conexiones de consola seguras, además de otras funcionalidades como reenvío de puertos con criptografía propia (independiente de TLS).

- Kerberos (RFC 4120): es un protocolo de autenticación centralizada que prescinde de la criptografía asimétrica, y en su lugar utiliza claves compartidas y tiques con criptografía simétrica.

- DNSSEC (RFC 4033, 4034, 4035): proporciona autenticidad e integridad a las respuestas dadas por el servidor DNS mediante firma digital, pero sin cifrar el tráfico.

- PGP / OpenPGP (RFC 9580): es un estándar criptográfico independiente de TLS, bastante utilizado en correo y en entornos locales.

- S/MIME (RFC 8551): es otro estándar criptográfico independiente de TLS muy enfocado en su uso en correos, pero a diferencia de PGP, sí requiere de una PKI.

- SPF (RFC 7208): es un mecanismo de autenticación basado en DNS para definir quién puede enviar correos en nombre de un determinado dominio.

- DKIM (RFC 6376): es un estándar de firma digital de correos electrónicos, que sirve para garantizar la integridad y autenticidad de los mensajes, pero sin cifrarlos.

- DMARC (RFC 9989): utiliza SPF y DKIM para establecer políticas de validación, respuesta y reporte. Enfocado a correo.

- Los protocolos y estándares revisados en esta capa indican la clara tendencia hacia la triada CIA. Dado que esta capa está íntimamente relacionada con el manejo de datos y la lógica de negocio de cada aplicación, es lógico que las protecciones de esta capa implementen mecanismo de cifrado para mantener la confidencialidad de las comunicaciones, hashes para asegurar su integridad, y firmas y otros mecanismos como cifrado autenticado para mantener la autenticidad.

Una vez vistos los protocolos, es el momento de describir el estado del arte del software y hardware defensivo que pretende asegurar esos protocolos:

- Capa de acceso a la red

    - NAC avanzado (Network Access Control, 802.1X): sirve para detectar cuando un dispositivo se conecta y aplicar las acciones correspondientes como bloquearlo, aislarlo, meterlo en una determinada VLAN (por ejemplo, como cuarentena). Los más avanzados (ZTNA) se basan en el principio de Zero Trust, y no son simples dispositivos semi-estáticos, sino que son capaces de detectar el estado de los dispositivos que monitorizan, establecer segmentación dinámica basada en el contexto, e integrarse con el XDR y el SIEM. Entre las soluciones comerciales más populares se encuentran Cisco Identity Services Engine (ISE), Aruba ClearPass y Forescout, todas ellas orientadas a la aplicación de políticas dinámicas de acceso, segmentación y autenticación basadas en el contexto.

    - Intent-based networking: los switches antiguos no disponen de un control de políticas centralizado, lo que dificulta la escalabilidad y la gobernanza centralizada, además de la visión global de ciberseguridad. Es por ello por lo que se ha desarrollado la Policy-based networking. Sus características son el establecimiento de políticas centralizadas con aplicación automática a switches, y una mayor cantidad de identidades a las cuales poder aplicar la configuración, como usuarios, dispositivos y otros elementos del contexto, facilitando la microsegmentación. Pero eso no es el estado del arte, el estado del arte es Intent-based networking, que traduce intenciones en lenguaje humano como por ejemplo “quiero una red de invitados” a reglas automáticas que minimizan la intervención humana y permiten una mayor abstracción. Entre las plataformas comerciales que implementan esta tecnología, destacan Cisco Catalyst Center, Juniper Apstra y VMware NSX. Cabe mencionar que la Policy-based networking y su evolución (Intent-based networking) se sustenta sobre SDN (Software Defined Networking), que proporciona su base técnica de arquitectura.

- Capa de Internet

    - Firewalls: los firewalls han avanzado mucho desde su nacimiento tanto a nivel hardware como software. Los firewalls de última generación (NGFW) son capaces de evaluar el estado de una conexión (y aplicar acciones al respecto), pues son stateful, además de establecer reglas muy específicas teniendo en cuenta la inspección profunda de paquetes (DPI), y tienen la posibilidad de integrarse con el resto de software de seguridad de la red para realizar respuestas rápidas y coordinadas ante incidentes.

    - IDS/IPS: son dispositivos de red intermedios, perimetrales, por los que se hace pasar todo el tráfico que se quiere controlar para que estos dispositivos, en función de sus reglas, comportamiento y circunstancias, generen alertas, bloqueen automáticamente ataques, y despleguen acciones con el resto de los dispositivos de seguridad como los firewalls.

    - Telemetría de red: en la actualidad existen distintas plataformas de observabilidad y telemetría de red. Las más representativas son Juniper Mist AI, Cisco ThousandEyes o Arista CloudVision. Cuyas capacidades principales son la monitorización contínua, correlación automática de eventos, análisis de comportamientos, y detección de anomalías basada en IA.

- Capa de aplicación:

    - SIEM (Security Information and Event Management) + UEBA (User and Entity Behavior Analytics) + SOAR (Security Orchestration Automation and Response): es el software que unifica todas las fuentes de datos relevantes a la hora de monitorizar en tiempo real la actividad maliciosa en una organización, y facilita la visión general de la ciberseguridad en la organización, además de reaccionar a tiempo asociándose con el SOAR, y mejorando la precisión de las acciones tomadas gracias al UEBA, con técnicas modernas de análisis de patrones y comportamiento basadas en reglas e IA.

- XDR (Extended detection and response): la evolución natural de los antiguos antivirus basados en firmas los llevó a implementar escudos para cubrir agujeros de seguridad visibles que no podían ser resueltos mediante el análisis basado en firmas, como por ejemplo el escudo basado en comportamiento y técnicas más modernas que implican el uso de IA, llegando así a los EDR (Endpoint detection and response). Los EDR por su parte han evolucionado a los XDR, donde ya no solo se concibe el software para proteger cada host individual, sino para extenderlo a toda la red facilitando la monitorización continua, más fuentes de datos de incidentes, y una mayor agilidad a la hora de responder ante amenazas. Los XDR han evolucionado hacia los MDR, lo que supone una externalización parcial o total de la gestión del XDR, pero se ha decidido no incluirlo dado que en un escenario simulado no tiene sentido externalizar por coste, dado que no se ocasiona ningún coste.

- Reverse proxies: son dispositivos intermediarios que se sitúan entre el cliente y servidor, abstrayendo al cliente de su interacción con el servidor. Gracias a eso, se reduce la superficie de ataque al no interactuar el cliente directamente con el servidor, evitando enumeración de servicios o escaneos directos de infraestructura y actuando como bastión; al ser un punto central de control de acceso se puede utilizar como balanceador de carga y autenticación centralizada previa al backend; puede inspeccionar el TLS, descifrar el tráfico para analizarlo y posteriormente volverlo a encriptar hacia el servidor; puede inspeccionar el tráfico a nivel de aplicación integrándose por ejemplo con un WAF; y se pueden establecer protecciones como rate limiting y protección contra DoS. Un ejemplo sería Envoy Proxy.

- API gateways: una API gateway tiene ciertas funciones que también podría hacer un reverse proxy (como inspección TLS, rate limiting y autenticación), aunque su función no es idéntica. La principal función de las API gateways son servir como punto de acceso único a APIs (sobre todo microservicios), donde siguiendo el principio Zero Trust, se centraliza la autenticación y autorización reduciendo considerablemente la superficie de ataque y mejorando la

observabilidad. Además las API gateways tienen mejor inspección de tráfico a nivel de API. Un ejemplo sería Kong Gateway.

- Service mesh: un service mesh realiza la misma función que un API gateway pero solo para la comunicación entre microservicios, mientras que las API Gateways están pensadas para el tráfico exterior a los microservicios. Un ejemplo sería Envoy Proxy, ya que es bastante versátil y puede hacer también de service mesh.

- Firewalls de aplicación (como WAF): son firewalls específicos que filtran un determinado tipo de tráfico (como HTTP) y por tanto no se ponen donde pase todo el tráfico de la red, sino solamente las peticiones que van al servidor que está protegiendo. Son capaces de detectar y bloquear ataques como inyecciones SQL, o intentos de explotación de vulnerabilidades. Un ejemplo es ModSecurity.

2.2. OFENSIVO

Contrariamente a la estructura del estado del arte defensivo, el estado del arte ofensivo se estructura alineándose con el marco de referencia MITRE ATT&CK. Ello se debe a que las defensas se pueden implementar aisladamente y teniendo en cuenta una única capa del modelo TCP/IP, pese a no ser lo recomendado. Sin embargo lo más común en los ataques es que involucren varias o incluso todas las capas del modelo TCP/IP, no sirviendo este modelo como clasificación coherente del estado del arte ofensivo. En su lugar, el marco de referencia adoptado nos permite analizar y clasificar correctamente las técnicas utilizadas en los ataques, pudiendo apreciar la evolución y tendencias como se muestra de forma detallada a continuación:

- Reconocimiento: la fase de reconocimiento consiste en mapear la superficie de ataque del objetivo e identificar la máxima información posible acerca de los objetivos. Tradicionalmente esta fase se basaba en el uso de técnicas OSINT, escaneos de puertos, y sniffing pasivo.

El estado del arte ha evolucionado bastante desde entonces, sistematizando el OSINT mediante buscadores específicos como SHODAN, y herramientas unificadas avanzadas como Maltego, que permiten realizar investigaciones profundas de un objetivo o varios, tengan el tipo de superficie de ataque que tengan.

También forman parte del estado del arte las tecnologías de Attack Surface Management (ASM), que monitoriza continuamente la superficie de ataque aportando dinamismo e información en tiempo real; han emergido soluciones basadas en el cloud reconnaissance, como consecuencia de las tendencias de migración parcial o total de la infraestructura de red de las organizaciones a la nube; el identity reconnaissance, donde los atacantes no buscan solamente reconocer un host en particular o una infraestructura física determinada, sino mapear las relaciones de identidad lógicas en todo el dominio; automatización masiva (ya sea con Kali MCP o con las herramientas citadas como SHODAN); y fingerprinting pasivo (como por ejemplo utilizando herramientas como p0f, Zeek o JA3).

La tendencia observada en esta fase es a la automatización del descubrimiento y gestión de activos, correlación de información contextual de la superficie de ataque extraída de diversas fuentes, y monitorización en tiempo real.

- Desarrollo de recursos: en esta fase, el atacante prepara todo lo necesario para realizar el ataque, es decir, esta fase puede involucrar la compra de dominios y malware (o su desarrollo en su defecto), creación de servidores C2 (comando y control), creación de certificados, cuentas de correos, entre otras herramientas. Esta es una de las fases que más ha evolucionado, donde la complejidad de los ataques ha ido forzando un mayor desarrollo de las herramientas relativas a esta fase, con tecnologías como el malware polimórfico (capaz de mutar para evadir detección), infraestructura cloud efímera, dominios rotatorios, certificados TLS válidos (para levantar menos sospechas), e infraestructura distribuida.

  En conclusión, la simplicidad de los ataques en las épocas más embrionarias de la informática, debido a la menor complejidad de internet y sus defensas, y a la ausencia de una PKI universal, no forzaban a realizar un desarrollo de recursos previo al ataque como los que hoy en día son el día a día de las APTs y otras amenazas.

- Acceso inicial: esta fase supone el punto de entrada del actor malicioso a los sistemas objetivo. Es decir, es el momento en el cual el atacante practica la primera intrusión a los sistemas objetivos. Esta fase ha sido en especial dependiente de los cambios en la industria, pues los actores maliciosos siempre buscan atacar aquellos sistemas que les vayan a comportar una recompensa al final de los ataques, lo que generalmente suele comportar que los objetivos sean o utilicen tecnologías con adopción masiva. Y pese a la relativa estabilización de todo tipo de soluciones tecnológicas populares, desde navegadores hasta bases de datos, pasando por todo tipo de software, la industria y el mercado están en constante cambio, es por ello a su vez cambian los principales objetivos de los ciberdelincuentes. En esta fase, lógicamente, los ataques también se han modernizado bastante, mientras en el pasado fueron tendencia las explotaciones de errores de configuración simples, autenticaciones débiles por falta de contraseñas robustas, y vulnerabilidades en el software por falta de S-SDLCs, hoy en día la tendencia es hacia el MFA fatigue, AiTM (Adversary in the Middle, evolución natural del MITM), robo de sesiones/tokens, OAuth/SAML abuse, ataques a la cadena de suministro, explotación de CVEs públicos, explotación de vulnerabilidades zeroday, entre otros abusos donde median las APIs y clouds, ya que cada vez hay un mayor uso de estas tecnologías. En breve resumen, los principios y fundamentos de los ataques siguen siendo los mismos, pero se han modernizado por la necesidad de esquivar defensas y adaptarse a las nuevas tecnologías.

- Ejecución: en este punto el atacante empieza a ejecutar código malicioso o determinados comandos y scripts con el objetivo de ir interaccionando con los sistemas objetivo que se ha conseguido penetrar en la anterior fase. Históricamente, la operativa de esta fase se basaba en el uso y ejecución de scripts y binarios directamente en el sistema objetivo, mientras, el estado del arte en esta fase apunta hacia las siguientes técnicas: uso de malware fileless (es decir se ejecuta completamente en memoria), reflective loading (cargar DLLs directamente en memoria), LOLBins (Living Off The Land Binaries) que son herramientas ejecutables normales del sistema a las que se le puede dar un uso malicioso. En definitiva, esas técnicas aportan sigilo, reducir al máximo el número de indicadores de compromiso

generados para dificultar la detección por parte de las defensas como las soluciones XDR.

- Persistencia: esta fase tiene un objetivo muy claro: mantener el acceso. Los sistemas objetivos pueden sufrir una serie de cambios naturales en el transcurso de sus operaciones normales, como reinicios de sistema o de servicios, cierres de sesión, cambios de configuración, o incluso intentos de expulsar a los actores maliciosos de los sistemas infectados, por tanto, estos tratan de mantener el acceso de diversas maneras con la intención de pasar desapercibidos, como históricamente ha ocurrido con la creación de nuevas cuentas, establecimiento de puertas traseras, modificación de claves del registro, etc. Lo último en persistencia comprende las tecnologías de persistencia cloud, abuso IAM, OAuth persistence, web shells modernas y persistencia fileless (que se ayuda de los métodos previamente descritos). La tendencia nuevamente apunta hacia la minimización de indicadores de compromiso, automatización y nuevas tecnologías como objetivo de ataque.

- Escalada de privilegios: en este punto el atacante busca obtener más permisos dentro de la infraestructura objetivo, con la intención de expandir sus capacidades ofensivas. Tradicionalmente se han empleado muchas técnicas para ese fin, como explotar vulnerabilidades locales, fallos de configuración, o incluso tokens/credenciales mal protegidas.

  Sin embargo, la tendencia de la industria se ha movido más hacia técnicas menos manuales y más automatizadas, capaces de analizar y probar muchos más *attack paths*. Paralelamente a esa tendencia surge también la estrategia de basar esta fase en el abuso de relaciones de confianza, permisos e identidades existentes, que en depender únicamente de la explotación de vulnerabilidades.

  Las técnicas modernas más destacadas de escalada de privilegio son el kerberoasting, abuso de Active Directory (por su gran popularidad), token impersonation, escalado de privilegios en cloud IAM, y grafos automáticos de privilegios. En el mapeo automático de privilegios mediante grafos destacan herramientas como BloodHound, que son capaces de analizar e identificar todas las relaciones de grupos, usuarios,

permisos entre otras entidades dentro de un dominio para mostrar los posibles caminos de escalado de privilegios.

- Evasión de defensas: esta fase consiste en ocultarse a los ojos de las defensas del objetivo, con el objetivo de evitar o dificultar la detección por parte de los mecanismos de seguridad y monitorización del objetivo. Es decir, emplear técnicas como ofuscación de malware, cifrado de payloads, uso de procesos legítimos del sistema, fraccionamiento de paquetes de red, entre otras para ocultar la actividad maliciosa y tratar de pasar desapercibido.

  El estado del arte de esta fase es uno de los que más se han visto forzados a evolucionar, con el notable incremento de las capacidades de monitorización, observabilidad y detección por parte de las ciberdefensas. Mientras que años atrás una simple ofuscación o cifrado del código de un malware ya imposibilitaba su detección temprana dado que los antivirus generalmente se basaban principalmente en firmas, hoy en día un malware puede ser detectado mediante análisis de patrones y comportamiento, ausencia de ejecutables firmados, aislado mediante sandboxing, y analizado en profundidad con las últimas técnicas de análisis de malware, tanto estático como dinámico.

  El conjunto de técnicas que más destacan hoy en día comprende fundamentalmente evasión de XDR, AMSI bypass (evitar que el antimalware pueda detectar actividad maliciosa a través de la Antimalware Scan Interface (AMSI) cuando el atacante utilice herramientas como powershell), ETW bypass (evitar que windows registre los eventos maliciosos), syscalls directas, evasión de sandbox, y uso de tráfico cifrado y esteganografía.

  La tendencia es clara: sofisticación de la interacción del malware con las APIs del sistema operativo, utilización de herramientas y mecanismos en los que confía el sistema operativo de la víctima, y enfoques en el cambio y sigilo del comportamiento.

- Acceso a credenciales: en esta fase, el actor malicioso busca prepararse para movimientos posteriores obteniendo credenciales e identidades válidas de los

sistemas vulnerables a los que ha logrado acceder para poder reutilizarlas en otras máquinas.

A lo largo del desarrollo de la ciberseguridad, han existido gran variedad de técnicas aún hoy en día utilizadas como ataques Pass-theHash, Pass-the-Ticket, keylogging, session hijacking, cookie theft, abuso Kerberos, dumping de memoria LSASS, identity theft, etc.

Sin embargo, dado que la informática ha ido evolucionando desde la autenticación local mediante el sistema operativo, seguido de los servidores de autenticación centralizada local, para finalmente dar paso a las grandes infraestructuras federadas de IAM, secret managers (como Azure Key Vault, AWS Secrets Manager o HashiCorp Vault), y a los grandes proveedores de identidad, esta fase se ha ido adaptando a los protocolos y estándares de identidad de cada época, pasando del simple robo y cracking de contraseñas, al compromiso de tokens, sesiones y mecanismos de autenticación distribuidos.

- Descubrimiento: una vez obtenido el acceso inicial, el objetivo en el que consiste esta fase es el de obtener la máxima infraestructura objetivo desde su interior, analizando objetivos que no eran visibles desde el exterior en la fase de reconocimiento.

  Tradicionalmente, esta tarea ha estado marcada por las técnicas de enumeración de red tanto pasivas como activas, listado de usuarios de algunos servicios vulnerables, y consultas a servicios de directorio activo. En la actualidad, el descubrimiento se orienta en torno a las técnicas de descubrimiento de recursos en entornos cloud, enumeración de clústeres Kubernetes, análisis de relaciones de privilegios, identidades y dominios mediante grafos, el mapeo de servidores de directorio activo utilizando herramientas como BloodHound, y el reconocimiento automatizado de APIs internas y servicios expuestos siguiendo marcos de referencia como el CTEM.

  Las tendencias que están marcando el pentest en esta fase se definen por el reconocimiento de entornos más amplios, variados y complejos, la automatización y el manejo de los datos a la hora de procesarlos y mostrarlos al atacante.

- Movimiento lateral: esta fase consiste en pivotar desde el host inicialmente comprometido hacia otros equipos de la red, con el objetivo de aumentar la severidad del ataque. El movimiento lateral, en su desarrollo más embrionario incorporaba técnicas básicas como el uso de exploits o mecanismos de control remoto como RDP para pivotar.

  Mientras que lo más moderno en movimiento lateral apunta hacia técnicas como el pivoting en entornos cloud y distribuidos, los movimientos laterales en Kubernetes y otros orquestadores de clústeres, los abusos del service mesh, movimiento lateral mediante APIs, y uso de credenciales e identidades válidas sin claros desvíos del comportamiento normal de esas identidades para no levantar sospechas.

  Por lo que se aprecia que la evolución en los métodos de movimiento lateral son consecuencia principalmente de las evoluciones en los sistemas de control remoto internos, gestores de permisos e identidad, y las defensas.

- Colección: en esta etapa, el actor malicioso recolecta toda aquella información sensible que tenga intención de exfiltrar. Pueden ser correos electrónicos, bases de datos, documentos, o incluso elementos de propiedad intelectual como software.

  Esta fase ha evolucionado mucho, ya que en épocas donde la ciberdefensa no estaba tan desarrollada, ni siquiera se contemplaban mecanismos para detectar y evitar fugas de datos, por tanto, un actor malicioso únicamente debía garantizar su anonimato, sin temor a que fuese detectado explícitamente en el momento del acceso a la información sensible.

  En la actualidad, el atacante comprime y organiza estos archivos sensibles para facilitar la siguiente fase. El estado del arte de esta fase comprende y se desarrolla hacia los secretos cloud, el abuso de APIs, robo de información en repositorios Git, pipelines CI/CD, robo de telemetría interna y uso de selección inteligente de datos, priorizando la posterior exfiltración de los más interesantes.

- Exfiltración: esta fase es la continuación directa de la anterior y consiste en mover la información robada a infraestructura controlada por el actor malicioso evitando ser

detectado en todo momento. Se suelen emplear técnicas como el envío mediante protocolos comunes (como covert channels en ocasiones) como SMTPS, HTTPS, DNS, TCP, ICMP, UDP, túneles SSH o incluso canales tradicionales encubiertos, como podría ser algún chat. Se emplean muchas veces algún tipo de ocultación, como túneles TLS, esteganografía o encriptación basada en contraseña. Además de las técnicas mencionadas, el estado del arte incorpora el uso de cloud storage legítimo y las técnicas mencionadas en la fase de desarrollo de recursos, además de mezclar tráfico normal con tráfico de exfiltraciones.

La tendencia ha evolucionado desde el uso de métodos y canales que proporcionasen facilidades por encima de todo al atacante, a priorizar ante todo el sigilo y camuflarse cada vez más con el tráfico legítimo.

- Impacto: esta fase supone la finalización del ataque y la culminación de sus objetivos. Las técnicas pueden variar bastante en función de los objetivos de los actores maliciosos. Se pueden presentar en esta fase acciones como simples o complejos sabotajes como bombas del tiempo o borrado de datos, otros tipos de denegación de servicio, o acciones más sofisticadas que no van tanto a hacer daño, sino a obtener rédito económico como el uso de ransomware y otros tipos de extorsión como el amenazar con publicar la información sensible. El estado del arte moderno incluye métodos como la multiple extorsión (normalmente mediante ransomware combinando con amenazas de exfiltración de datos, borrado, uso de las autoridades para amenazar con una posible gran multa en caso de fallos de seguridad, etc), wipers, sabotaje cloud, destrucción selectiva (incluso destrucción de hardware), y monetización vendiendo las puertas traseras y accesos a terceros.

  Asimismo, las infraestructuras OT están siendo cada vez objetivo de más ataques, debido al potencial impacto que ocasionan.

  La tendencia histórica de los objetivos de esta fase se ha movido de los ataques realizados por motivaciones personales, a ataques con motivaciones mucho más profundas, como el lucro económico a gran escala, el sabotaje a la competencia, el robo de información secreta como secretos de estado o secretos industriales, o a provocar la inestabilidad geopolítica de la potencia rival.

2.3. COMPARACIÓN DE LA PROPUESTA CON HERRAMIENTAS REPRESENTATIVAS DEL ESTADO DEL ARTE

Tras analizar las principales tendencias del estado del arte de técnicas ofensivas, resulta de interés comparar la propuesta desarrollada con algunas herramientas ampliamente utilizadas y confiables en el ámbito de los ejercicios de pentesting y la simulación de adversarios. El objetivo de esta comparación no es establecer una clasificación rígida entre ellas, sino identificar el ámbito de aplicación de cada una y contextualizar la contribución realizada en este trabajo.

En la actualidad, Pentera, MITRE Caldera, Infection Monkey, AutoSploit y BloodHound se sitúan como posibles soluciones que actúan en el mismo dominio que Skynet, ya sean herramientas dedicadas a pentest más general, abarcando muchos posibles escenarios, como Pentera o MITRE Caldera, o ya sean herramientas más específicas como BloodHound, cuyo escenario objetivo es aquel en el que exista el Active Directory.

Pentera es una de las soluciones más maduras y completas que ofrece la industria en cuanto a pentesting automatizado, ASM y gestión del riesgo, incorporando los últimos estándares industriales, como la gestión CTEM (Continuous Threat Exposure Management) de exposición. Sin embargo, la propuesta desarrollada de este trabajo permite una usabilidad diferente. Mientras que Pentera es una solución acabada enfocada a empresas, Skynet plantea una arquitectura modular y extensible, concebida para facilitar la incorporación de nuevos módulos de ataque y servicios, y orientada a su mejora continua por parte de la comunidad. Además, la herramienta desarrollada cuenta con una arquitectura pensada desde el principio para la integración y la inteligencia contextual (clasificando los ataques según el MITRE ATT&CK, por objetivos, capa del modelo OSI entre otros metadatos), por lo que no supone un problema para su actualización las nuevas tendencias y cambios en la ciberseguridad.

Caldera, por su parte, es una herramienta que ofrece realizar distintos ejercicios de simulación de adversarios, pudiendo reproducir campañas complejas de forma controlada. Sin embargo, tiene la necesidad de requerir una configuración inicial que Skynet no necesita (únicamente necesita saber su interfaz y su proveedor de IA en caso de estar activa esa

opción), y no ofrece un enfoque tan centrado en las posibilidades de automatización mediante LLM.

Autosploit es una herramienta cuya orientación principal reside en la explotación de vulnerabilidades y la automatización, combinando fuentes como Shodan o ZoomEye para reconocer un gran número de vulnerabilidades conocidas. Frente a ello, Skynet tiene unos objetivos y casos de uso diferentes, donde se requiere más adaptabilidad, requisito el cual la herramienta desarrollada tiene en consideración desde su diseño gracias a su arquitectura, y donde se tiene en cuenta la inteligencia contextual como un elemento esencial. Además, la arquitectura propuesta, utiliza plantillas concretas para sus ataques y servicios integrados, favoreciendo que se añadan nuevos ataques basados en vulnerabilidades conocidas.

BloodHound se ha convertido en la referencia para entornos AD (Active Directory), por su gran capacidad de análisis a la hora de descubrir toda la información relativa a AD que un equipo de ciberseguridad puede necesitar, ya sean rutas de escalado de privilegios, relaciones entre permisos, grupos de usuarios, dominios, etc, o incluso puede detectar las sesiones activas. Para que Skynet pudiera tener unas capacidades ofensivas específicas con respecto a herramientas como BloodHound, necesitaría una cantidad de desarrollo suficiente para implementar por lo mínimo los ataques (ya sean activos o pasivos) equivalentes, dado que desde un inicio ya se pensó en la adaptabilidad desde la arquitectura, pero para evitar el trabajo por duplicado, la arquitectura propuesta permite definir comandos de sistema a integrar para que utilice herramientas externas y las combine con su inteligencia contextual.

La propuesta presentada, se diferencia por combinar desde la arquitectura una gestión centralizada del contexto, un modelo modular para la incorporación de nuevos ataques y servicios auxiliares, soporte para escenarios tanto IT como OT, y la posibilidad de automatizar parcialmente el proceso mediante LLMs.

En resumen, aquí se muestra una tabla contextualizando todo lo analizado acerca de las soluciones más relevantes comparadas con la aportación de este trabajo:

Tabla 1: Tabla comparativa del estado del arte ofensivo y la herramienta propuesta

Característica	Pentera	Caldera	BloodHound	AutoSploit	Skynet

Código abierto	No	Sí	Sí	Sí	Sí
Arquitectura modular	Parcial	Sí	Parcial	Parcial	Sí
MITRE ATT&CK	Sí	Sí	Parcial	No	Sí
Cyber Kill Chain	Parcial	Parcial	No	No	Sí
Gestión contextual	Parcial	Parcial	Parcial	No	Sí
Integración LLM	No	No	No	No	Sí
IT	Sí	Sí	Sí	Sí	Sí
OT	Parcial	Parcial	No	No	Sí

2.4. CONCLUSIONES DEL ESTADO DEL ARTE

Defensa y ataque están unidos y van evolucionando de la mano a velocidad constante, debido al riesgo de posible inestabilidad de las infraestructuras críticas por ataques dirigidos por APTs con respaldo gubernamental, y por la democratización de la informática y el pentesting, lo que brinda oportunidades para realizar ciberataques a cada vez más personas, lo que aumenta los riesgos de ciberseguridad.

Skynet plantea la base arquitectónica para que el pentest sea accesible sin sacrificar integrabilidad, adaptabilidad al entorno, y pueda ser potenciado por la comunidad open-source.

La tendencia es clara: la industria se mueve hacia la automatización, IA, la centralización de gestión, pero distribución de recursos, infraestructuras híbridas y cloud, y visión global tanto en la defensa como el ataque.

3. Objetivos concretos y metodología de trabajo

Una vez analizado el estado del arte de la ciberseguridad tanto ofensiva como defensiva a grandes rasgos, el presente apartado define los objetivos del presente trabajo, la metodología seguida para el desarrollo de la propuesta y el procedimiento específico realizado para su evaluación experimental.

3.1. Objetivo general

El objetivo principal del trabajo es dar como resultado una herramienta de ataque centrándose en el enfoque dinámico, cambiante y muy variado del entorno al que se le puede enfrentar, además de la integración opcional desde el diseño con la automatización LLM.

Hoy en día existen una gran cantidad de herramientas funcionales de pentest, pero acostumbran a tener una superficial integración con otras herramientas. Por ejemplo, existen herramientas magníficas para escaneo de redes y fingerprinting como nmap, que podrían descubrir puertos abiertos y enumerar servicios que posteriormente podrían ser explotados con metasploit, pero no existe una integración completa de nmap con metasploit para comprobar que el host objetivo sigue siendo el mismo, pues solo se utiliza la IP, pero podría ser que ese host se hubiera apagado o desconectado de la red y el DHCP le hubiera dado su IP a otro, o se podrían haber operado otros cambios sensiblemente difíciles de detectar, como cambios en la configuración de red (como cambios en parámetros TCP). Dificultades que sin duda limitan el alcance de los ejercicios de pentest, al dificultar la visión y monitorización de la infraestructura objetivo.

Es por ello que Skynet se concibe desde su inicio como una herramienta con una arquitectura modular y altamente integrable, promocionando la programación de nuevos ataques y servicios, teniendo la capacidad de recibir actualizaciones en sus ataques mediante múltiples fuentes, sin depender del desarrollador original.

3.2. Objetivos específicos

Dado que el presente trabajo consta de una parte defensiva y otra parte ofensiva, se clasifican los objetivos en función de si pretenden satisfacer un requisito defensivo (es decir de la arquitectura del laboratorio) o por el contrario pretenden satisfacer un requisito ofensivo, ya sea la programación de un ataque, la configuración de Skynet, o cuestiones arquitectónicas de dicha herramienta ofensiva.

3.2.1. Objetivos específicos de desarrollo

- **OED01**: Desarrollar la arquitectura base de Skynet
- **OED02**: Implementar su núcleo
- **OED03**: Implementar el sistema de contexto
- **OED04**: Implementar módulos de ataque básicos y plantilla de ejemplo
- **OED05**: Implementar módulos de servicios básicos y plantilla de ejemplo
- **OED06**: Implementar automatización mediante LLM
- **OED07**: Diseñar la GUI

3.2.2. Objetivos específicos de evaluación

- **OEE01**: Diseño e implementación del escenario de baja seguridad, que representa un restaurante con Wi-Fi gratuito. No debe tener ningún tipo de captive portal ni segmentación entre clientes.
- **OEE02**: Diseño e implementación del escenario de media seguridad, que corresponderá a un hospital con una segmentación básica correspondiente a departamentos, y con un captive portal.
- **OEE03**: Diseño e implementación del escenario de alta seguridad y complejidad, que emula una zona industrial OT SCADA. Se toma como referencia el estándar IEC

62443, de forma se establece segmentación en zonas de seguridad y conexión mediante conduits.

- **OEE04**: Medir la cobertura de MITRE ATT&CK por parte de Skynet
- **OEE05**: Medir descubrimiento alcanzado
- **OEE06**: Medir la automatización conseguida mediante LLM
- **OEE07**: Medir rendimiento
- **OEE08**: Medir alertas, detecciones e identificaciones por parte de las defensas a las acciones de Skynet

3.3. Metodología del trabajo

Este trabajo siguió como estrategia general un enfoque experimental, en el que inicialmente se estableció toda la infraestructura básica (es decir, los tres escenarios de laboratorio, el espacio de trabajo y el diseño arquitectónico de Skynet). Y posteriormente, se desarrolló la herramienta mediante un enfoque incremental, ágil, realizando pruebas en paralelo durante el proceso de implementación con el objetivo de detectar errores o fallos de forma temprana y obtener una retroalimentación continua.

Los escenarios de laboratorio se han implementado utilizando GNS3 como plataforma de virtualización. Esta herramienta brinda las capacidades de integración y virtualización de dispositivos necesarias para implementar todas las topologías de los escenarios.

Skynet se ha desarrollado íntegramente en Python siguiendo una arquitectura modular. El núcleo es el encargado de la gestión del contexto, la indexación y coordinación de los ataques, servicios y módulos auxiliares. Para la manipulación y análisis pasivo del tráfico de red, se ha utilizado Scapy como biblioteca principal, pese a que no existe ningún impedimento arquitectónico a utilizar cualquier otra librería de Python.

Para el desarrollo se utilizó PyCharm Community Edition como entorno de programación y uv como gestor de dependencias, a fin de favorecer la usabilidad y reproducibilidad, y se utilizó Git para el control de cambios y versiones.

La evaluación práctica de la propuesta se realizó mediante la ejecución de la herramienta sobre los tres escenarios definidos. Para ello, se utilizaron las categorías de cobertura, progreso, descubrimiento, automatización, rendimiento y sigilo, en alineación con los marcos de referencia de la industria más utilizados como se mostrará en el siguiente apartado.

3.4. EVALUACIÓN Y ANÁLISIS DE LOS RESULTADOS

Se tendrán en cuenta las categorías y métricas declaradas en la siguiente tabla:

Tabla 2: Métricas de evaluación de la arquitectura propuesta

Categoría	Referencia	Métricas
Cobertura	MITRE ATT&CK	Número de técnicas ejecutadas
Progreso	Cyber Kill Chain	Fase máxima alcanzada, número de fases completadas
Descubrimiento	CTEM	Hosts descubiertos, puertos abiertos identificados, sistemas operativos identificados, objetivos priorizados
Automatización	Arquitectura propuesta	Número de decisiones tomadas automáticamente, utilización del contexto compartido

Rendimiento	Métricas del sistema	Uso de CPU, RAM y accesos en disco en la inicialización
Sigilo	MITRE ATT&CK	Número de alertas generadas, criticidad de las alertas, exposición de la identidad del atacante (MAC/IP).

3.5. LIMITACIONES DEL ESTUDIO

Los resultados obtenidos en cuanto a rendimiento pueden estar sesgados por el hardware particular utilizado en la ejecución. Para que se tenga en cuenta, el hardware que se ha utilizado corresponde a una CPU Intel I7 de 12ª generación, 16 GB RAM DDR4 con una velocidad de 3200 MT/s y un SSD conectado mediante la interfaz NVMe.

Debido a las limitaciones de tiempo, se han centrado los esfuerzos en la arquitectura y no a la implementación de todos los ataques más comunes hoy en día, utilizando en su lugar una pequeña cantidad representativa. Lo que impacta en los resultados, pues con mayor variedad de ataques, se cubriría una mayor cantidad de superficie de ataque.

Los laboratorios utilizados son abstracciones tomando los elementos fundamentales de cada escenario real. Los escenarios reales podrían alcanzar una variedad de tráfico y superficie de ataque mayor, sin embargo, se ha intentado minimizar al máximo ese posible sesgo en cuanto a las defensas, haciendo que sí sean representativas del mundo real.

4. Desarrollo específico de la contribución

Una vez presentado el estado del arte y la metodología seguida, este capítulo describe de forma detallada la contribución desarrollada en el presente trabajo. En primer lugar, se especifican los requisitos considerados durante el diseño de la herramienta, para posteriormente presentar su arquitectura y aspectos principales de implementación. Para finalizar, se describen los escenarios de laboratorio utilizados para evaluar la propuesta.

4.1. REQUISITOS DEL SOFTWARE

En este capítulo se describirán todos los requisitos que se han tenido en cuenta para diseñar, desarrollar y evaluar a Skynet.

4.1.1. Requisitos funcionales

- **RF01**: El usuario debe poder ejecutar, monitorizar y detener comandos desde la interfaz.
- **RF02**: La herramienta debe mantener un contexto compartido durante toda la ejecución del pentest.
- **RF03**: El usuario debe poder limpiar la pantalla.
- **RF04**: La herramienta debe conocer y mostrar los dispositivos descubiertos tras cualquier ataque de reconocimiento.
- **RF05**: La herramienta debe conocer y mostrar los dispositivos comprometidos tras cualquier ataque.
- **RF06**: La herramienta debe tener ataques tanto IT como OT.
- **RF07**: La herramienta debe tener la opción de utilizar proveedores de IA LLM para automatizar sus acciones y que el usuario pueda verlas en tiempo real.

- **RF08**: El usuario debe poder indicarle a Skynet otras herramientas a utilizar integradas en el sistema operativo.

- **RF09**: La herramienta debe proporcionar una interfaz gráfica para visualizar el estado de la ejecución.

- **RF10**: La herramienta debe permitir incorporar nuevos ataques y servicios mediante módulos independientes.

4.1.2. Requisitos no funcionales

- **RN01**: La interfaz deberá mantener una visualización y respuesta fluida al iniciarse la herramienta.

- **RN02**: La herramienta debe ser ejecutable en todos los sistemas operativos para ordenador (Windows, Linux, macOS y sus derivados).

- **RN03**: Skynet debe dejar logs en caso de errores.

- **RN04**: La arquitectura propuesta debe permitir su ejecución sin acceso a internet.

- **RN05**: La configuración de Skynet será lo más simplificada posible, delegando las tareas posibles a la automatización y el reconocimiento para facilitar todo lo posible su uso por todos los públicos.

4.1.3. Restricciones de diseño

- **RD01**: La herramienta será desarrollada en Python 3.13 o superior, debido a la versatilidad que ofrece este lenguaje además de su popularidad, para permitir que la comunidad pueda hacer aportes.

- **RD02**: Skynet debe tener una arquitectura modular que conste de un motor lógico (core) que sea la centralización de la gestión.

- **RD03**: Git será la herramienta utilizada para realizar el control de versiones.

- **RD04**: GitHub será el repositorio utilizado en consonancia con Git para almacenar y gestionar el software.

- **RD05**: uv será el software utilizado para gestionar el entorno de Python, por su rapidez y gestión de dependencias.

4.2. DESCRIPCIÓN Y FUNCIONALIDADES DE SKYNET

En el presente apartado se detalla la arquitectura de Skynet, mostrando y justificando las principales decisiones de diseño y desarrollo adoptadas, además de la organización interna de la herramienta. Finalmente se analizan sus principales funcionalidades y el mecanismo previsto para su ampliación mediante nuevos ataques y servicios auxiliares.

4.2.1. Arquitectura

La arquitectura propuesta se representa a nivel lógico en el siguiente grafo:

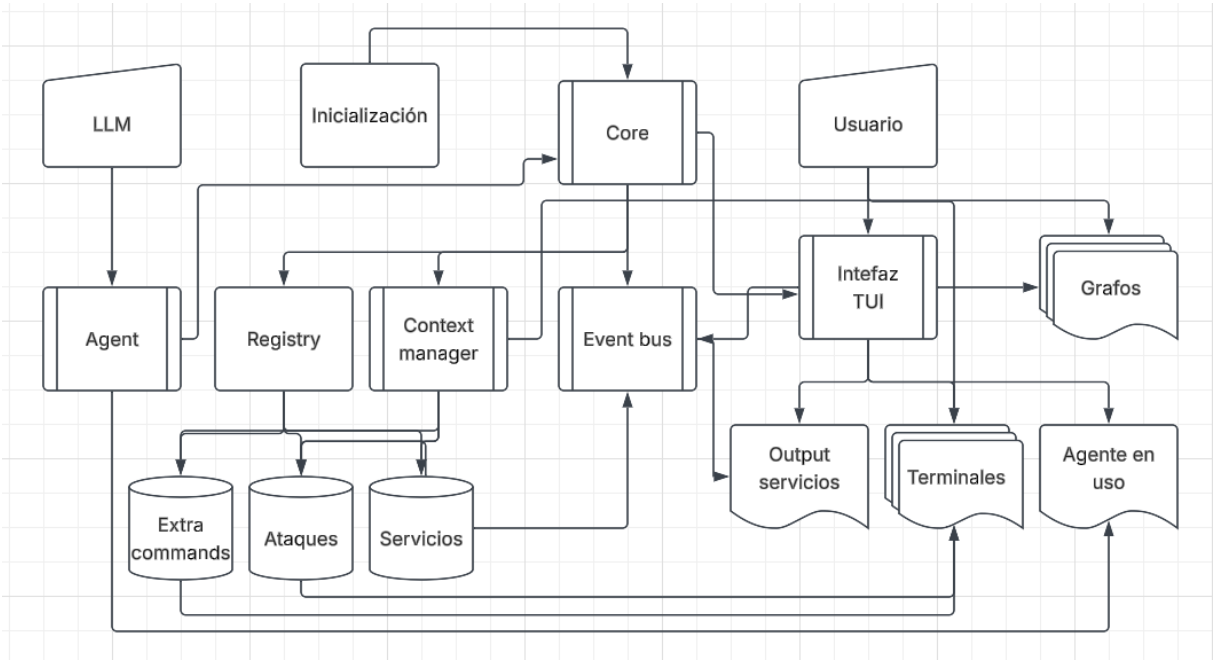

Figura 1: Diagrama representativo de la arquitectura

Leyenda:

- **Rectángulos con barra lateral:** módulos software que implementan una funcionalidad concreta del sistema.
- **Rectángulos:** submódulos software que implementan una funcionalidad concreta del módulo padre.
- **Cilindros:** repositorios de módulos cargables (ataques, servicios y comandos extra).
- **Documentos:** vista concreta de la interfaz de usuario.
- **Documentos apilados:** panel con múltiples vistas de la interfaz de usuario.
- **Flechas:** flujo principal de información, eventos o control entre componentes.

La figura anterior muestra la arquitectura propuesta. Se ha diseñado teniendo en cuenta la necesidad de modularidad, donde un núcleo (Core) es el encargado de coordinar durante la ejecución el resto de los módulos del sistema.

Durante el proceso de inicialización se realizan las comprobaciones estrictamente necesarias de configuración para el funcionamiento de la herramienta. Esencialmente se comprueba que el usuario ejecute la herramienta con derechos de administrador, y que se definan explícitamente mediante variables de entorno qué interfaz utilizará la herramienta para realizar los ataques, y si utilizará o prescindirá del soporte LLM. En caso de utilizar LLMs, es necesario añadir las variables de entorno concretas del proveedor LLM.

El núcleo (Core) de la arquitectura constituye el elemento central del sistema, coordinando todos los módulos. El núcleo a su vez se subdivide en motores. Una vez inicializado Skynet, se encarga de registrar los ataques y servicios mediante sus respectivos motores, conectar el motor de servicios al event bus para que este pueda mostrar los mensajes de los servicios hacia un menú de la interfaz gráfica, sincronizar el motor de ataques donde se gestiona la lógica de los grafos de descubrimiento y compromiso con la interfaz gráfica, y mantener sincronizado un contexto compartido por los distintos módulos.

Skynet marca una clara distinción entre ataques y servicios. Los ataques hacen referencia a acciones ofensivas que se ejecutan con un objetivo concreto, como el reconocimiento, la persistencia, o la elevación de privilegios. En cambio, los servicios, corresponden a procesos orientados a ejecutarse en segundo plano, cuyo objetivo principal es o bien automatizar acciones auxiliares de un ataque en curso, servir de apoyo al usuario durante la ejecución de Skynet, o cubrir vacíos que podrían dejar los ataques, como la simulación del tráfico normal

correspondiente a IPs y MACs suplantadas para evitar la detección basada en comportamiento.

El módulo del gráfico se utiliza para crear la infraestructura necesaria para que los ataques y servicios puedan reportar información descubierta o dispositivos comprometidos, que luego se visualizará en forma de gráfico desplegable cuando Skynet está en ejecución.

El archivo core/initialization.py es el encargado de hacer todas las tareas de comprobación del entorno antes de la ejecución de Skynet: comprobar que están definidas las mínimas variables de entorno necesarias para que Skynet esté configurado, además de poseer derecho de administrador para poder enviar paquetes con la MAC cambiada sin problemas.

La interacción del usuario con la herramienta se realiza mediante Textual (TUI), sirviendo como base de la interfaz gráfica. La interfaz gráfica está dividida en 3 columnas: la columna izquierda corresponde a los grafos; la columna central a los terminales de comandos de Skynet.

Skynet utiliza 4 paneles gráficos para mostrar de forma ordenada la información. A la izquierda están situados los 2 paneles relativos a los dispositivos comprometidos (situada abajo), y la información de descubrimiento (situada arriba). En el centro está situado el panel principal de Skynet, que contiene 3 pestañas: el terminal de comandos de Skynet, el bus de eventos, y la pestaña de configuración que ha recibido Skynet. Y en la derecha está situado el agente que se esté utilizando, por defecto Skynet, aunque solo es decorativo en caso de no utilizar la IA.

4.2.2. Ampliación de Skynet

La arquitectura de Skynet ha sido concebida para facilitar su ampliación por parte de terceros sin necesidad de modificar la lógica principal de la aplicación. Este apartado detalla las decisiones y especificidades arquitectónicas adoptadas relativas a la extensión de los ataques y servicios de Skynet.

En todos los módulos donde se prevé que haya ampliaciones (attacks y services) se incluye un archivo .py.example que sirve como plantilla a la hora de implementar nuevos ataques o

servicios. Tanto ataques como servicios utilizan cada uno una subclase para especificar nuevas funcionalidades, siguiendo un sistema de plugins.

Obligatoriamente todos los ataques deben de tener definidos unos inputs, que serán los argumentos que tomará el comando, y unos outputs, que se utilizarán para alimentar la gestión contextual y los grafos de Skynet. Además, se deben definir unos metadatos específicos del ataque que ayudan a gestionar el contexto tanto antes como después de la ejecución. Esos metadatos del ataque son: la fase de la Cyber Kill Chain correspondiente, la técnica del MITRE ATT&CK que utiliza el ataque, el objetivo, la capa de red en la que se opera el ataque, el ruido que genera el ataque, y el tipo de ejecución del ataque (ya sea que consista en una única conexión como el despliegue de una carga útil, ejecución persistente como un ataque DoS, o un ataque continuo, como un escaneo pasivo).

Los servicios también deben llevar obligatoriamente unos inputs que consumir, aunque al igual que ocurre con los ataques, se pueden dejar vacíos. Pero se prescinde de la necesidad de declarar metadatos alineados con los marcos de referencia en los que se basa este trabajo, u outputs, ya que los servicios no tienen como finalidad ser el elemento principal de un ataque.

Para mejorar el rendimiento, se ha optado por integrar los servicios en la lógica asíncrona de Textual, mientras que en los ataques se ha optado por aislar su ejecución en subprocesos, dando más libertad de uso de recursos de hardware y evitando bloqueos de la TUI, permitiendo el uso de librerías sin restricciones por código asíncrono y obteniendo un mejor control de los ataques, permitiendo matar los procesos cuando sea necesario sin el retraso que provocaría hacerlo de otras formas.

Finalmente, la interfaz gráfica soporta una definición de temas mediante JSON, permitiendo y facilitando la personalización sin necesidad de tocar el código base por terceros.

4.2.3. Funcionalidades principales

- **Gestión de ataques**: los ataques constituyen la acción principal que realiza Skynet. Cada ataque definido se registra y expone automáticamente como un comando interno cuyo nombre coincide con el de su clase en minúsculas. Únicamente es

necesario proporcionar los argumentos de los ataques cuando estos están marcados como requeridos.

La siguiente figura ejemplifica la ejecución de un ataque de reconocimiento pasivo:

Figura 2: Terminal de Skynet en ejecución de un ataque pasivo

- **Gestión de servicios**: la arquitectura propuesta no solo maneja ataques, también ha sido concebida para el manejo de los servicios que acompañan a los ataques. Mediante el comando “service”, la herramienta propuesta permite realizar las siguientes acciones relativas a los servicios:
    - listar los servicios disponibles
    - consultar los servicios actualmente en ejecución
    - visualizar el estado individual de un servicio
    - consultar su ayuda integrada
    - iniciar nuevos servicios
    - detener servicios activos

A continuación se muestran las figuras de ejemplo:

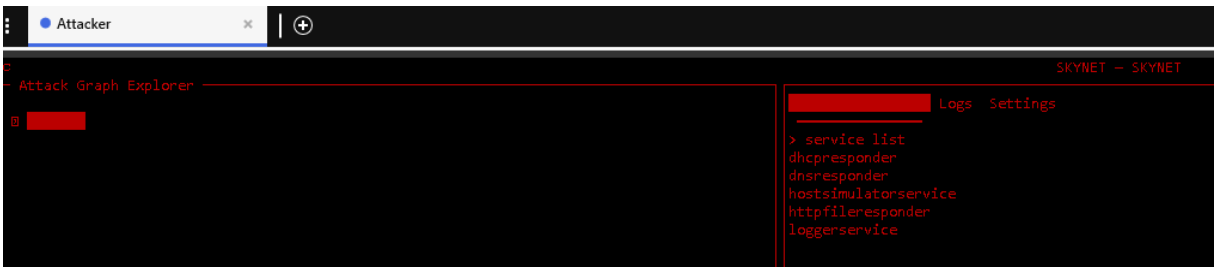

Figura 3: Terminal de Skynet operando servicios

- **Visualización del contexto**: toda la información obtenida de los ataques, servicios en ejecución, configuración y rendimiento se muestra en los distintos paneles de la interfaz gráfica para mejorar la usabilidad de Skynet. Concretamente se muestra al usuario, que puede visualizar en tiempo real:
    - Dispositivos y elementos descubiertos
    - Hosts comprometidos
    - Eventos generados por la interfaz gráfica y servicios
    - Configuración activa
    - Consumo de recursos de hardware
- **Gestión de la ejecución**: la herramienta propuesta está orientada a tener una usabilidad y experiencia de usuario que permite realizar los ejercicios de red team con comodidad. Es por ello que la interfaz integra las siguientes funcionalidades:
    - Cambio entre terminales
    - Cancelación de comandos
    - Inicio de nuevos terminales
    - Cambios de temas estéticos en tiempo de ejecución
- **Modo Stealth**: Skynet incorpora un modo de privado denominado *Stealth*, cuyo objetivo es ocultar información sensible del proceso de pentest. Básicamente sustituye el panel izquierdo donde están los grafos por un terminal de comandos del sistema, a la vez que oculta completamente el panel derecho y cambia los colores a tonalidades oscuras para dificultar la lectura a una distancia media. Este modo se ideó para que en los ejercicios de pentest se pudiera evaluar físicamente cuánto es capaz de ocultarse un atacante entre cámaras y personal de la organización. A continuación, se muestra cómo se visualiza Skynet con el modo Stealth activado:

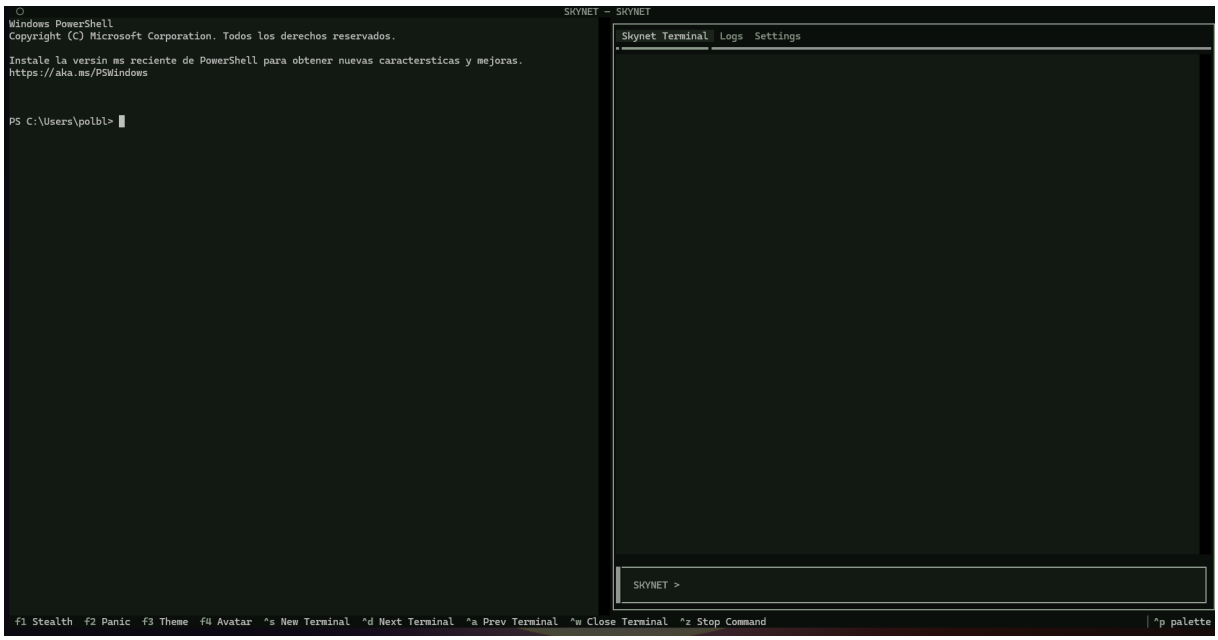

Figura 4: Skynet en modo "stealth"

4.2.4. Configuración e inicialización

La herramienta propuesta ha sido diseñada con el objetivo específico de facilitar el despliegue en nuevos entornos, prescindiendo de configuraciones largas y complejas. Alineado con este objetivo, se ha automatizado la instalación de dependencias y creación del entorno virtual, donde se realiza desde la primera ejecución de Skynet, sin necesidad de ejecutar otros comandos manualmente.

Una vez ya se han instalado las dependencias necesarias y uv cede el control de la ejecución a la herramienta, esta comprueba que tiene derechos de administrador y la configuración establecida explícitamente.

Skynet únicamente necesita conocer la interfaz que debe utilizar para sus ataques y si se utilizará algún proveedor de IA, que en ese caso se deberán proporcionar las variables de entorno específicas del proveedor concreto.

La siguiente figura muestra el aspecto de Skynet después de completar correctamente el proceso de inicialización:

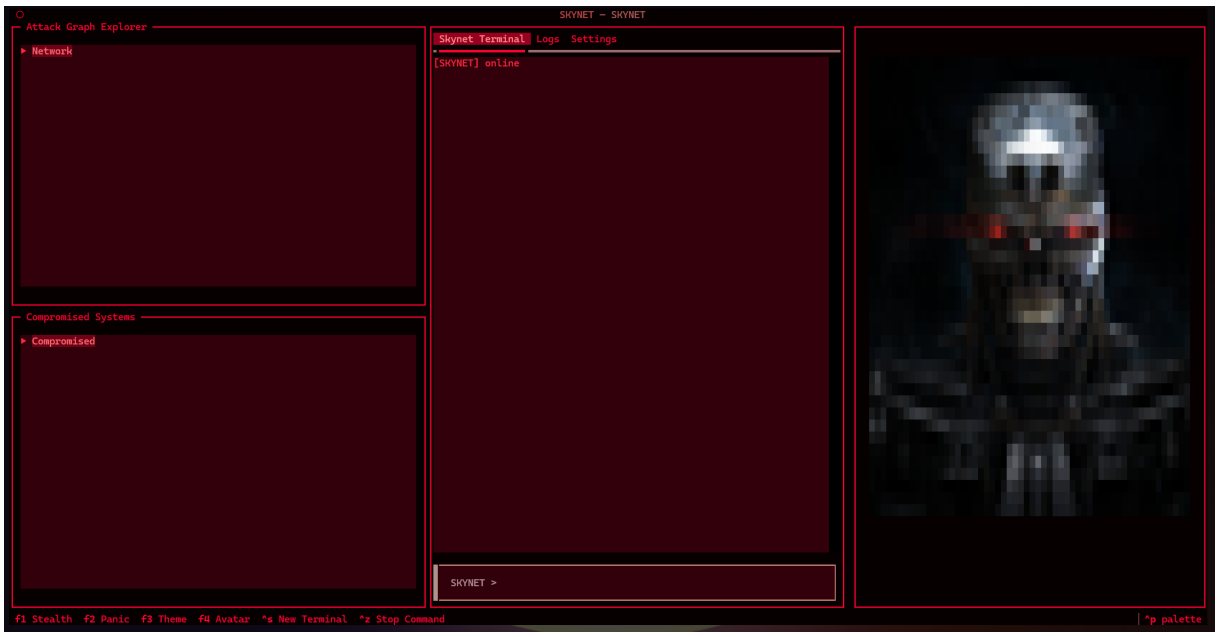

Figura 5: Skynet visualizándose en ejecución en el terminal de Windows 11

4.3. EVALUACIÓN

En este apartado primero se muestra y detalla la creación de los laboratorios, para asegurar su reproducibilidad, aclarando que ya vienen preparados, y a continuación se muestra a Skynet en acción en cada uno de ellos, para finalizar analizando el nivel de compromiso que ha logrado Skynet en cada caso. Para ir directamente a la ejecución y análisis de Skynet, saltar al apartado 4.3.2.

4.3.1. Diseño de los escenarios de evaluación

1. Escenario 1: Restaurante (baja seguridad):

Este escenario tiene como objetivo evaluar el comportamiento de la herramienta propuesta en un escenario con una ausencia de mecanismos de alerta y respuesta de incidentes, para posteriormente poder comparar los resultados con el resto de los escenarios, y concluir si Skynet presenta deficiencias en sus objetivos de ataque más básicos, como por ejemplo el reconocimiento, y analizar el grado de afectación que suponen las defensas a Skynet.

En la siguiente figura se muestra la topología del presente laboratorio:

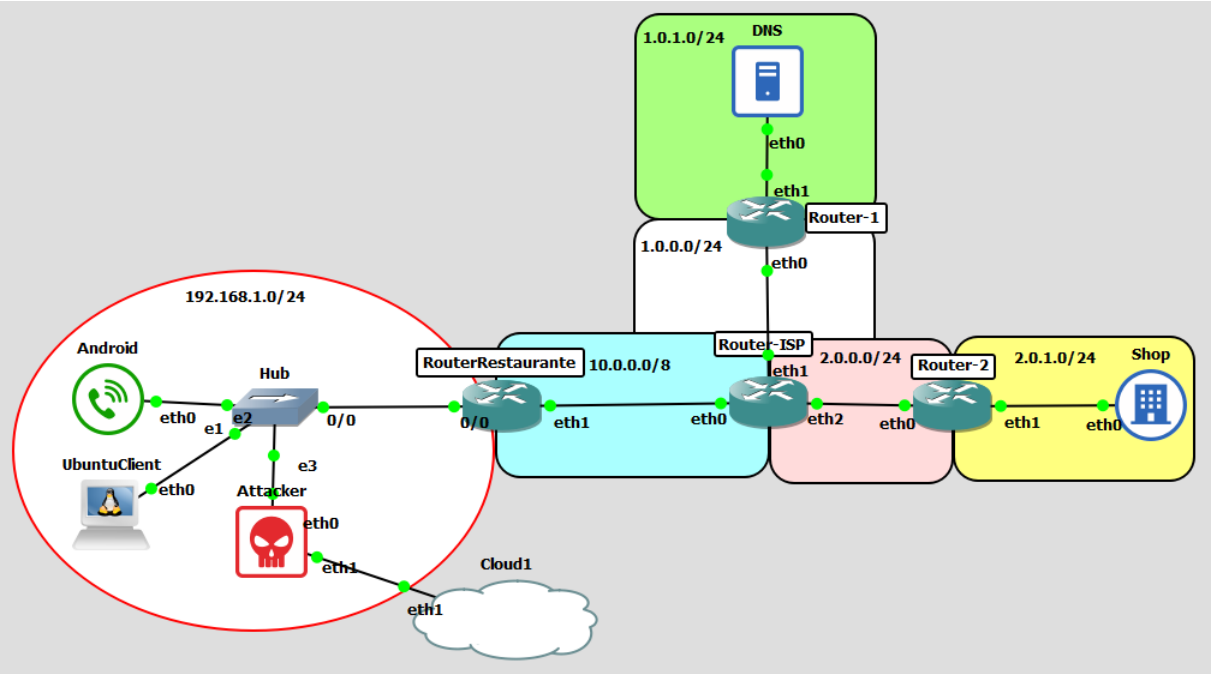

Figura 6: Escenario fácil cargado en GNS3

Este es el escenario más básico. El atacante parte desde la red cuyo borde es de color rojo, que representa el Wi-Fi abierto del restaurante. El restaurante solamente tiene un router (que incluye un servidor DHCP) y los clientes que se van conectando a la red. El router del restaurante está conectado al router del ISP, y a partir del router del ISP, todo representa la WAN. Cabe mencionar que al atacante (máquina Attacker) se le ha añadido una interfaz extra para poder utilizar la IA.

Para simplificar la configuración del laboratorio y facilitar su mantenimiento, el encaminamiento entre los distintos routers se realiza de forma automática mediante OSPF. En la siguiente figura se muestra la configuración del router del restaurante:

```
vyos@router-restaurante:~$ show config
interfaces {
    loopback lo {
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.1.0/24
            network 10.0.0.0/8
        }
        parameters {
            abr-type cisco
            router-id 2.2.2.2
        }
    }
}
service {
    dhcp-server {
        shared-network-name LAN1 {
            subnet 192.168.1.0/24 {
                default-router 192.168.1.1
                lease 86400
                name-server 1.0.1.2
                range 0 {
                    start 192.168.1.100
                    stop 192.168.1.200
                }
            }
        }
    }
}
```

Figura 7: Configuración del Router del escenario fácil

El resto de routers también implementan OSPF en su configuración, permitiendo flexibilidad por si en el futuro se necesita implementar laboratorios derivados, sin necesidad de empezar desde cero:

```
vyos@router-isp:~$ show config
interfaces {
    loopback lo {
    }
}
protocols {
    ospf {
        area 0 {
            network 10.0.0.0/8
            network 1.0.0.0/24
            network 2.0.0.0/24
        }
        parameters {
            abr-type cisco
            router-id 1.1.1.1
        }
    }
}
```

Figura 8: Configuración del Router correspondiente al ISP, del escenario fácil

```
vyos@router-1:~$ show config
interfaces {
    loopback lo {
    }
}
protocols {
    ospf {
        area 0 {
            network 1.0.0.0/24
            network 1.0.1.0/24
        }
        parameters {
            abr-type cisco
            router-id 3.3.3.3
        }
    }
}
```

Figura 9: Configuración del Router-1 del escenario fácil

```
vyos@router-2:~$ show config
interfaces {
    loopback lo {
    }
}
protocols {
    ospf {
        area 0 {
            network 2.0.0.0/24
            network 2.0.8.0/24
            network 2.0.1.0/24
        }
        parameters {
            abr-type cisco
            router-id 4.4.4.4
        }
    }
}
```

Figura 10: Configuración del Router-2 del escenario fácil

La topología de red incluye un único servidor DNS BIND9 que será el encargado de resolver las peticiones de toda la red. A continuación, se muestra su configuración de red a fin de garantizar la completa reproducibilidad del escenario y transparencia en el análisis:

DNS interfaces

```
#
# This is a sample network config, please uncomment lines to configure the network
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
auto eth0
iface eth0 inet static
                address 1.0.1.2
                netmask 255.255.255.0
                gateway 1.0.1.1
                up echo nameserver 127.0.0.1 > /etc/resolv.conf

# DHCP config for eth0
#auto eth0
#iface eth0 inet dhcp
#               hostname DNS-2
```

Figura 11: Configuración de red del DNS del escenario fácil

El servidor DNS implementa una única zona para minimizar el gasto de recursos sin perder representatividad del escenario real. En la siguiente figura se muestra el archivo de definición de la zona:

```
/ # cat var/cache/bind/db.lab
$TTL 604800
@   IN  SOA ns.lab. admin.lab. (
        2026051201
        604800
        86400
        2419200
        604800 )

@       IN  NS  ns.lab.
ns      IN  A   1.0.1.2

shop     IN  A   2.0.1.2
/ #
```

Figura 12: Archivo de zona del DNS del escenario fácil

Finalmente, antes de realizar ninguna evaluación, se verifica el correcto funcionamiento del escenario de laboratorio. Para ello se opta por un ping al servidor web “shop.lab”, de forma que se verifica tanto el correcto comportamiento del DNS, como el del servidor web, confirmando a su vez que mediante OSPF se han trazado las rutas correctamente y existe

conectividad entre todos los routers. A fin de garantizar transparencia, se muestra a continuación la siguiente figura donde se verifica el correcto estado del laboratorio:

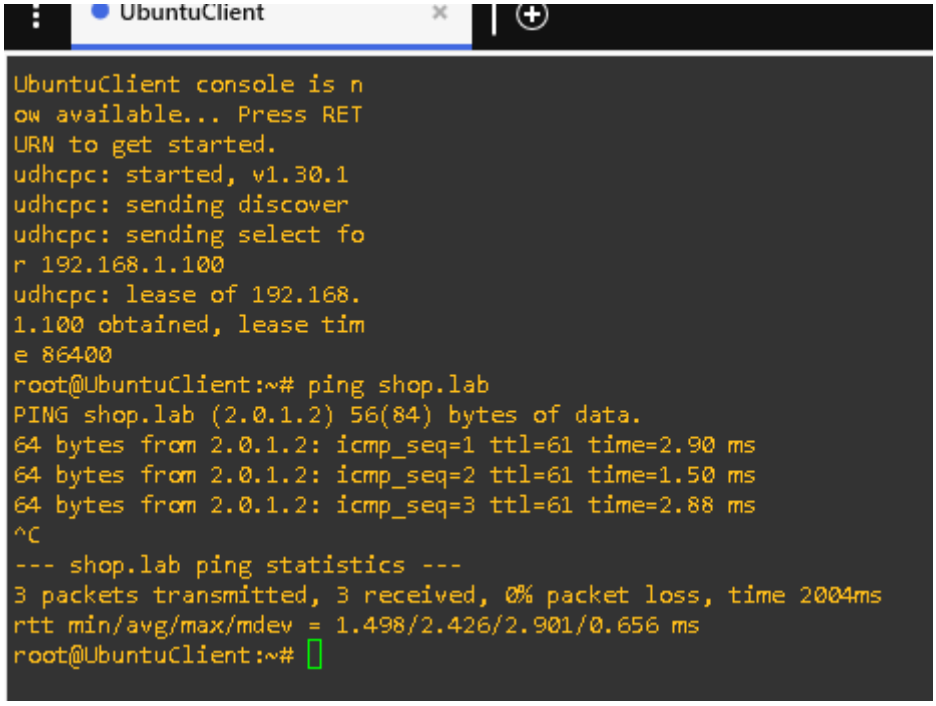

Figura 13: Test de conectividad desde el cliente Ubuntu

2. Escenario 2: Hospital (media seguridad):

El segundo escenario tiene como objetivo evaluar el comportamiento de Skynet en un entorno corporativo. Este escenario se distingue del anterior por la implementación de medidas comunes de seguridad compartidas por muchos escenarios reales cuyos requisitos de seguridad superan a las redes domésticas e internet cafés. Las principales medidas de seguridad implementadas en este laboratorio son la segmentación mediante VLANs, reglas de filtrado en el cortafuegos y autenticación de usuarios invitados mediante Captive Portal.

En la siguiente figura se muestra la topología del presente escenario, basado en un hospital parcialmente digitalizado:

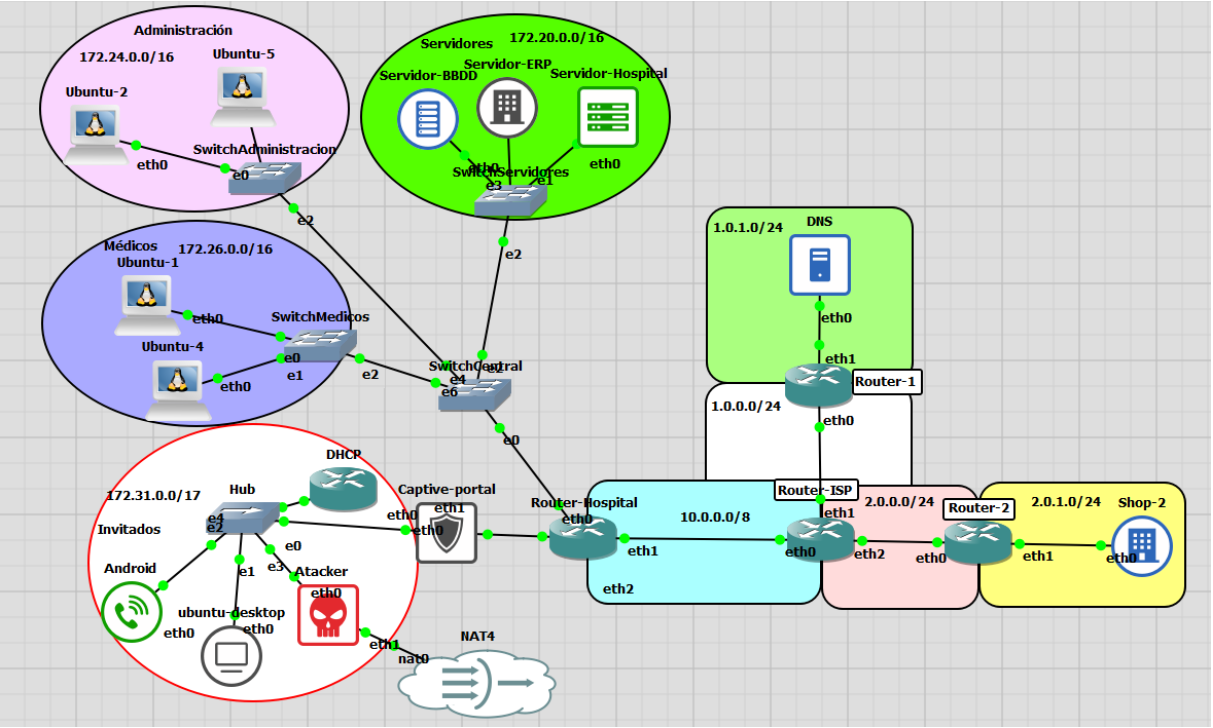

Figura 14: Escenario medio cargado en GNS3

Este laboratorio ya consta de una mayor segmentación y seguridad que el anterior, con VLANs, firewalls con reglas más estrictas, un mejor firmware en el router (MikroTik CHR), y un captive portal. Se ha mantenido la parte WAN+ISP exactamente igual que en el escenario anterior para facilitar las pruebas.

A continuación, se muestra la tabla con la segmentación de red establecida:

Tabla 3: Topología lógica de red del escenario de dificultad media

Departamento	VLAN	IP de red
Invitados	31	172.31.0.0/16
Médicos	26	172.26.0.0/16
Administración	24	172.24.0.0/16
Servidores	20	172.20.0.0/16
Segmento captive portal-router	-	192.168.1.0/24

Los switches tienen ya configuradas las VLANs de la siguiente forma:

- Switch central. Se configura la interfaz trunk hacia el router para que puedan viajar los paquetes de cada VLAN hacia él, y se configura cada conexión con cada switch de cada departamento en modo access para que solo viaje el tráfico de la misma VLAN de forma predefinida:

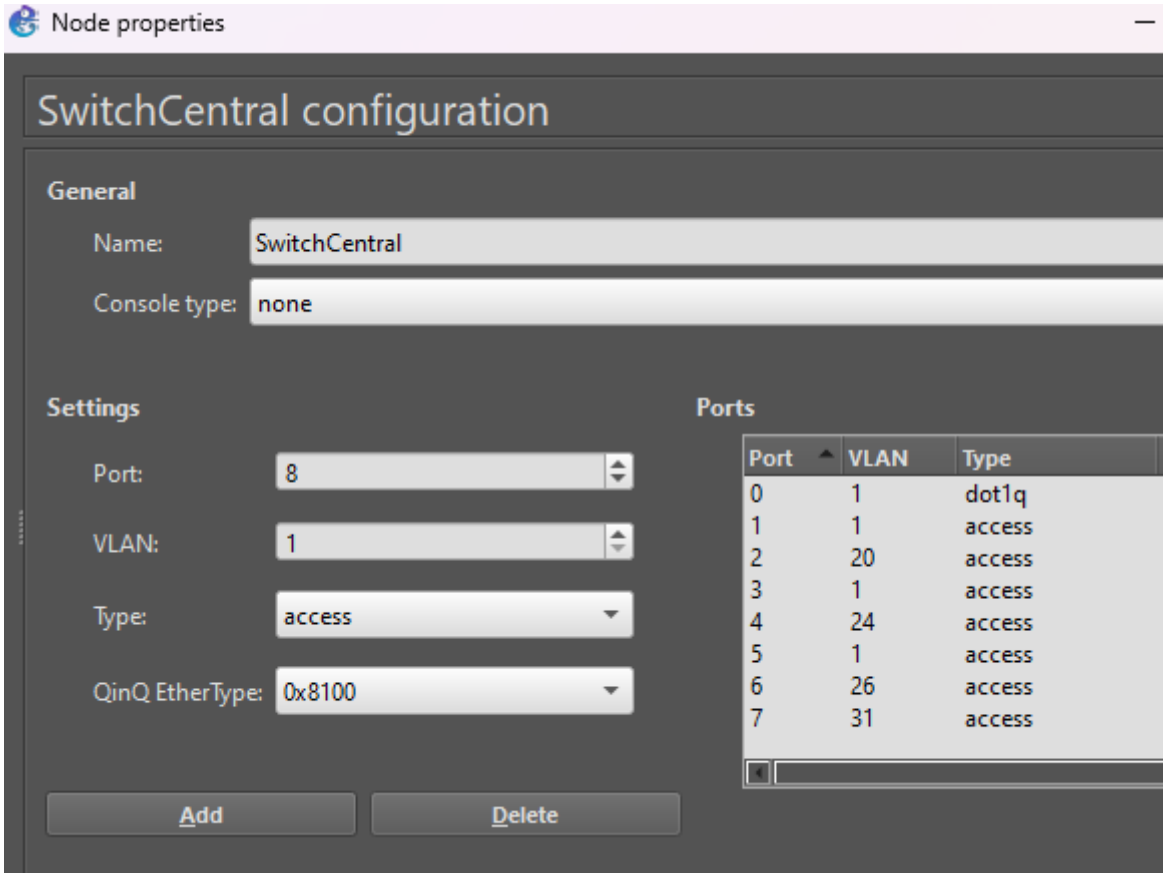

Figura 15: Configuración del switch central del escenario medio

- Switch del departamento médico. Se define la VLAN 26 en todos los puertos activos, por supuesto en modo access:

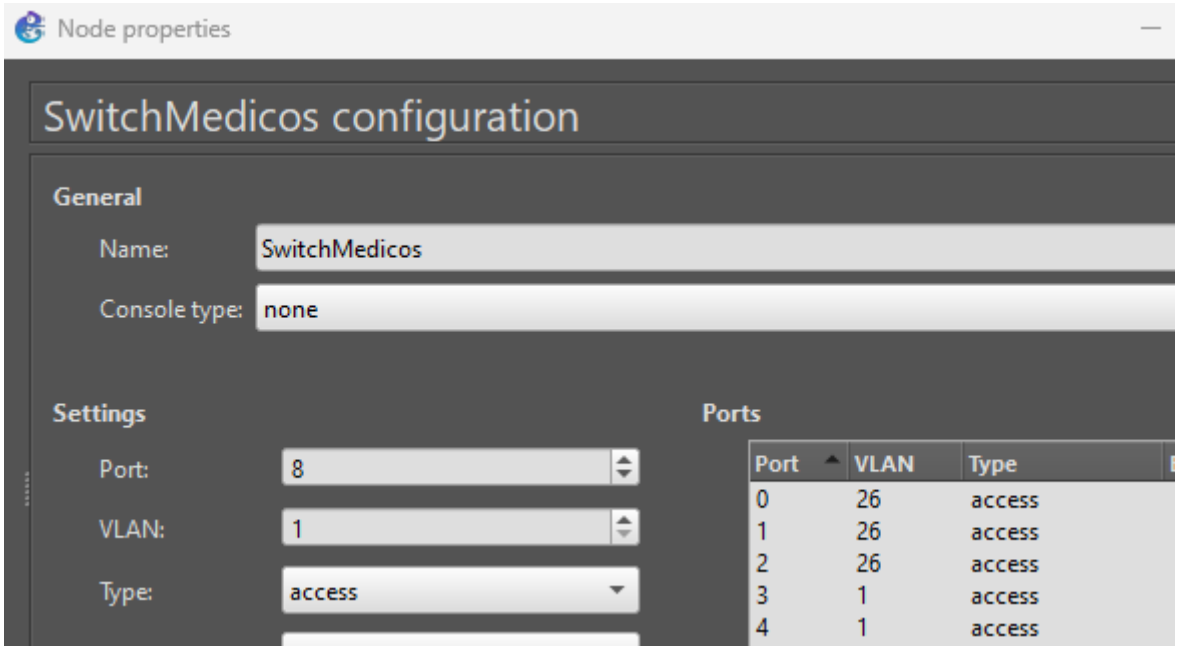

Figura 16: Configuración del switch del departamento médico del escenario medio

- Switch del departamento de administración. Se define la VLAN 24 en todos los puertos activos, por supuesto en modo access:

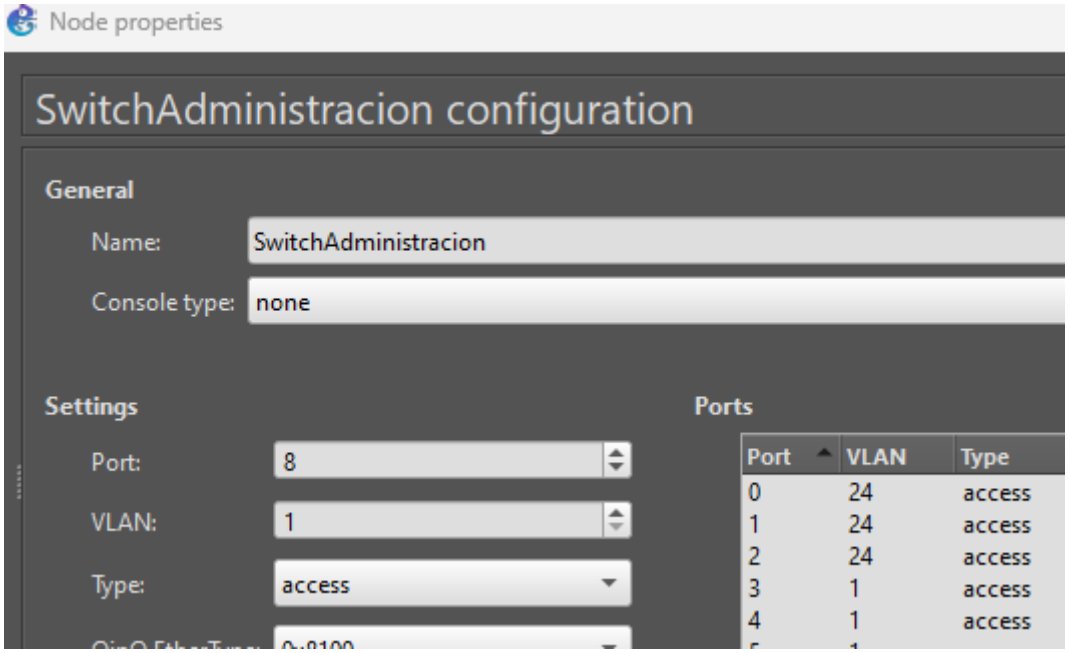

Figura 17: Configuración del switch del departamento de administración del escenario medio

- Switch del departamento de servidores. Se define la VLAN 20 en todos los puertos activos, por supuesto en modo access:

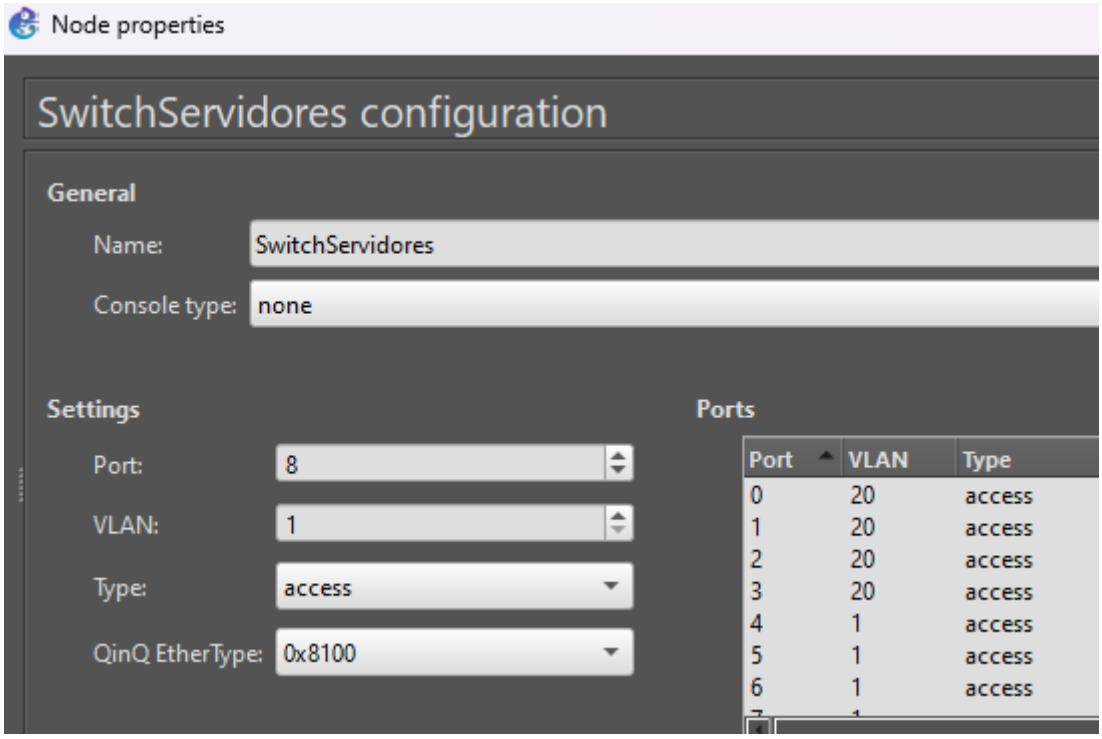

Figura 18: Configuración del switch del departamento de servidores del escenario medio

Aquí tenemos la configuración entera del router del hospital, estableciendo las distintas VLANs, DHCP por cada VLAN con reservas fijas para los servidores, gateway por cada VLAN, a continuación, el firewall para solo permitir a los médicos acceder a su servidor, y a administración a la red de servidores, y para finalizar, el OSPF para tener configuración automática de rutas con el resto de routers de la WAN simulada.

Ubuntu-1 | Router-Hospital

```
[admin@router-hospital] > /export
# 2026-06-18 09:44:42 by RouterOS 7.22
# system id = cwFy5O2YzoA
#
/interface ethernet
set [ find default-name=ether1 ] disable-running-check=no
set [ find default-name=ether2 ] disable-running-check=no
/interface vlan
add interface=ether1 name=vlan20 vlan-id=20
add interface=ether1 name=vlan24 vlan-id=24
add interface=ether1 name=vlan26 vlan-id=26
add interface=ether1 name=vlan31 vlan-id=31
/ip pool
add name=pool20 ranges=172.20.0.10-172.20.255.250
add name=pool24 ranges=172.24.0.10-172.24.255.250
add name=pool26 ranges=172.26.0.10-172.26.255.250
add name=pool31 ranges=172.31.0.10-172.31.255.250
/ip dhcp-server
add address-pool=pool20 interface=vlan20 name=dhcp20
add address-pool=pool24 interface=vlan24 name=dhcp24
add address-pool=pool26 interface=vlan26 name=dhcp26
add address-pool=pool31 interface=vlan31 name=dhcp31
/routing ospf instance
add name=ospf1 router-id=2.2.2.2
/routing ospf area
add instance=ospf1 name=backbone
/ip address
add address=172.20.0.1/16 interface=vlan20 network=172.20.0.0
add address=172.24.0.1/16 interface=vlan24 network=172.24.0.0
add address=172.26.0.1/16 interface=vlan26 network=172.26.0.0
add address=172.31.0.1/16 interface=vlan31 network=172.31.0.0
add address=10.0.0.2/8 interface=ether2 network=10.0.0.0
/ip dhcp-client
add interface=ether1 name=client1
/ip dhcp-server lease
add address=172.20.0.100 mac-address=02:42:AD:6A:43:00
add address=172.20.0.101 mac-address=02:42:66:78:4E:00
add address=172.20.0.102 mac-address=02:42:4D:7F:AA:00
/ip dhcp-server network
add address=172.20.0.0/16 dns-server=1.0.1.2 gateway=172.20.0.1
add address=172.24.0.0/16 dns-server=1.0.1.2 gateway=172.24.0.1
add address=172.26.0.0/16 dns-server=1.0.1.2 gateway=172.26.0.1
add address=172.31.0.0/16 dns-server=1.0.1.2 gateway=172.31.0.1
/ip firewall filter
add action=accept chain=forward connection-state=established,related
add action=drop chain=forward connection-state=invalid
add action=accept chain=forward dst-address=172.20.0.100 dst-port=443 protocol=tcp src-address=172.26.0.0/16
add action=accept chain=forward dst-address=172.20.0.0/16 src-address=172.24.0.0/16
add action=drop chain=forward dst-address=172.16.0.0/12 src-address=172.16.0.0/12
add action=drop chain=forward dst-address=172.20.0.0/16 src-address=172.16.0.0/12
add action=accept chain=forward out-interface=ether2
/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether2
/routing ospf interface-template
add area=backbone interfaces=vlan20
add area=backbone interfaces=vlan24
add area=backbone interfaces=vlan26
add area=backbone interfaces=vlan31
add area=backbone interfaces=ether2
/system identity
set name=router-hospital
[admin@router-hospital] >
```

Figura 19: Configuración del Router del hospital

Para realizar una correcta evaluación del funcionamiento del entorno, con el objetivo de asegurar la veracidad y transparencia de los resultados, se deben realizar las siguientes conexiones:

- **Comprobación de acceso a internet por parte de los departamentos médico y administración**. Como se puede apreciar en la siguiente figura, desde los terminales de los hosts se ha obtenido una respuesta del servidor, confirmando que los hosts correspondientes a médicos y personal de administración tienen internet:

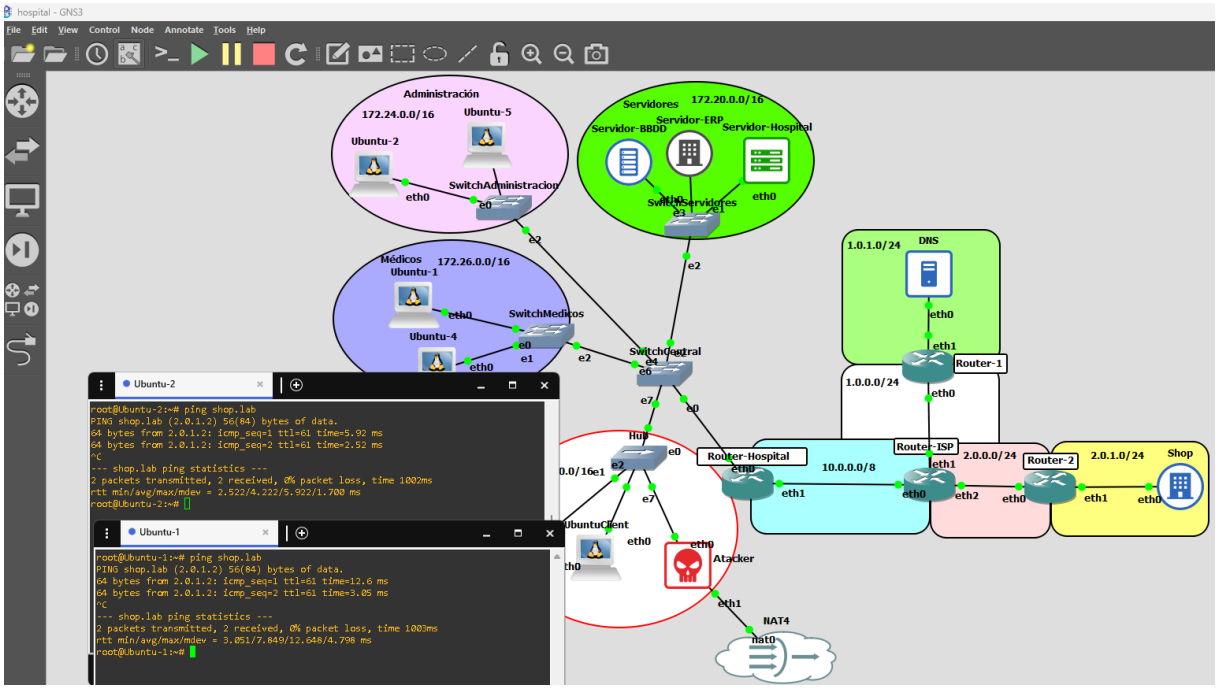

Figura 20: Test de conectividad a la WAN y vista general del laboratorio medio

Como se puede apreciar en la siguiente figura, recortada de la anterior por cuestiones de visibilidad, los hosts resuelven correctamente shop.lab y el servidor web responde correctamente:

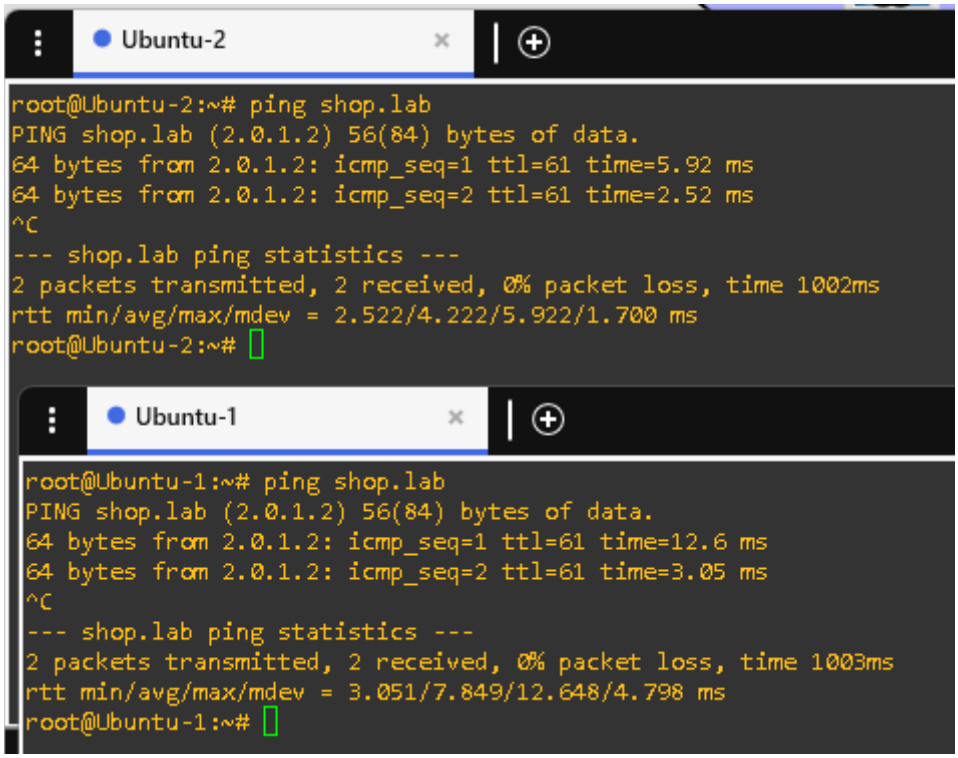

Figura 21: Test de conectividad al internet simulado (imagen ampliada)

- **Sin conexiones no autorizadas entre VLANs**. Tal como se ha definido previamente, no se autorizan conexiones entre VLANs por defecto, únicamente se permiten las conexiones de cada departamento a los servidores necesarios. Para ello nuevamente se han empleado peticiones ICMP:

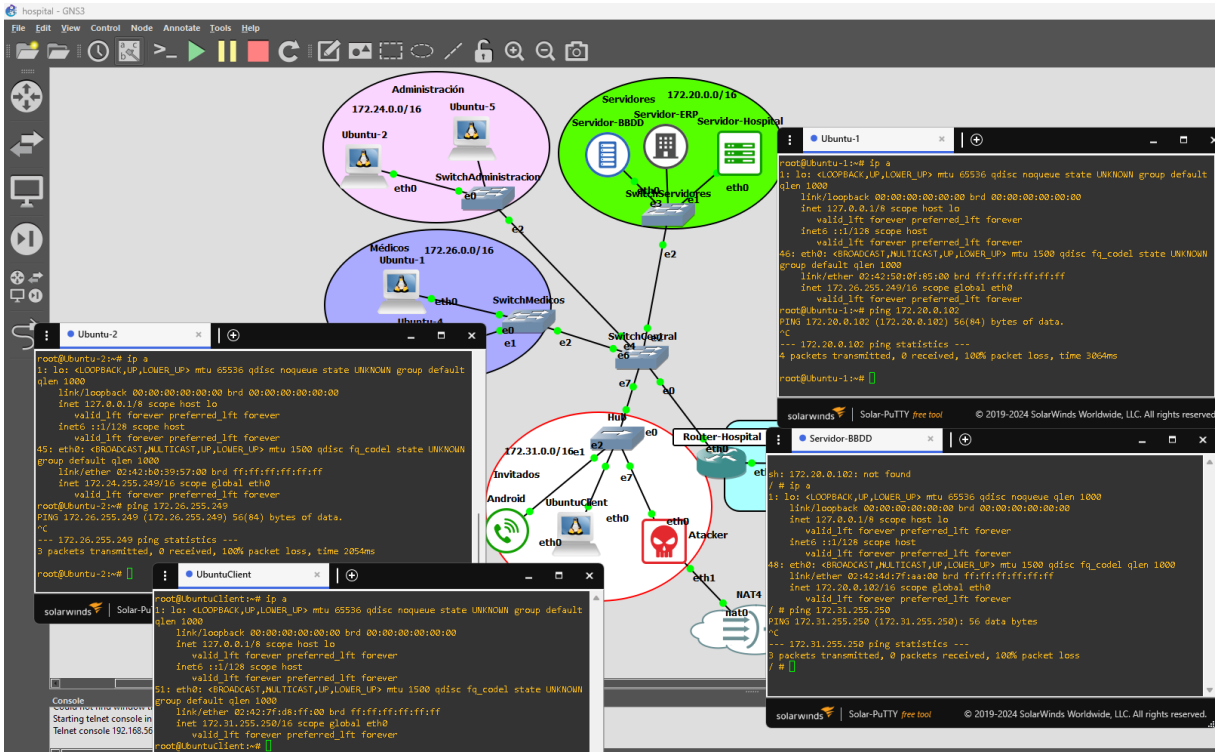

Figura 22: Test de segmentación de VLANs y vista general del laboratorio medio

Como se puede apreciar en la siguiente figura, administración (Ubuntu-2) no tiene conexión con el ordenador del médico (Ubuntu-1):

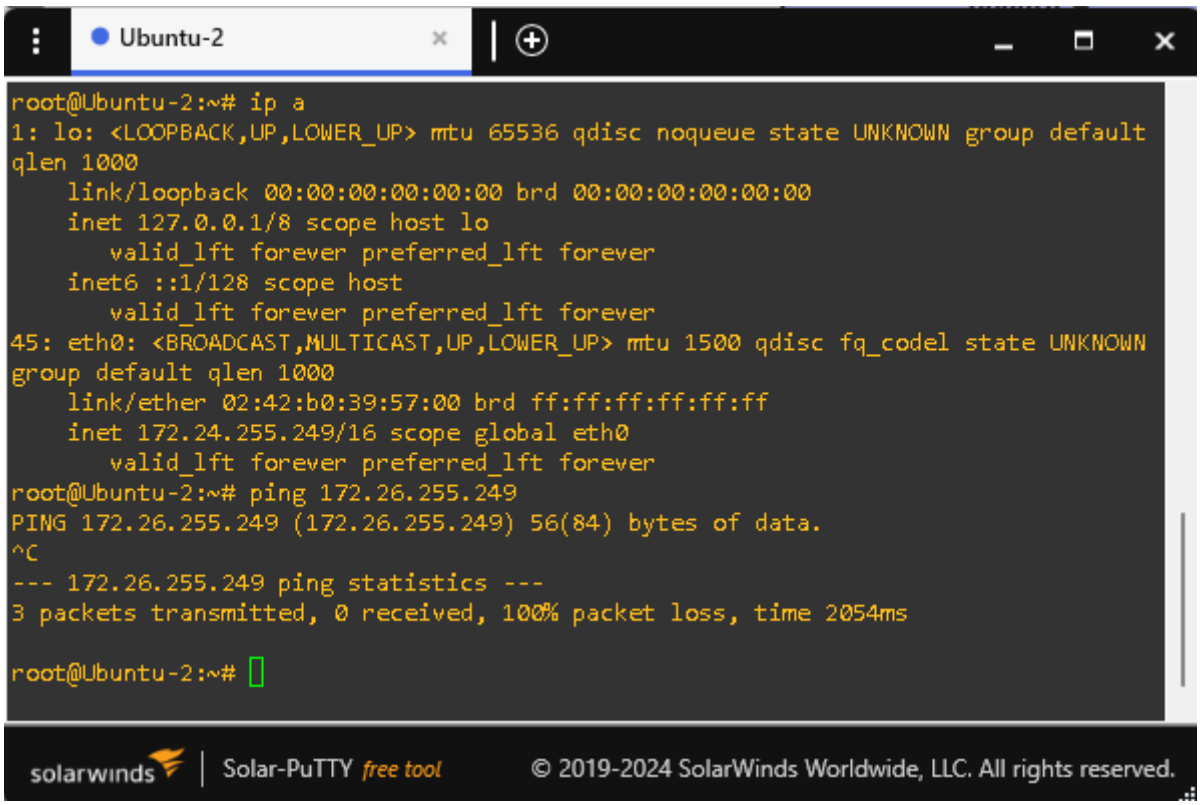

Figura 23: Test de conectividad entre la VLAN de médicos y administración

Tampoco el ordenador del médico tiene derecho a conectarse a la base de datos:

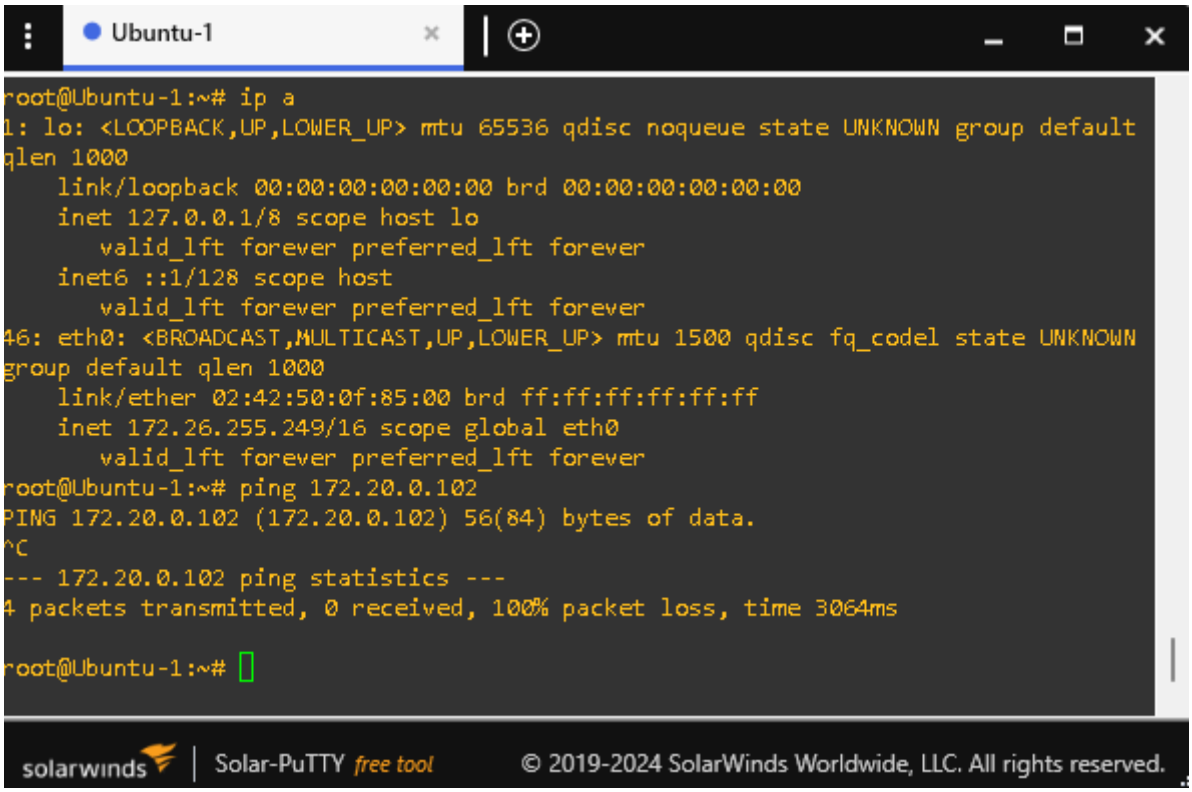

Figura 24: Test de conectividad entre la VLAN de médicos y la base de datos

Y para acabar, la base de datos no tiene derecho a conectarse a "UbuntuClient", un ordenador que se sitúa en la red de invitados:

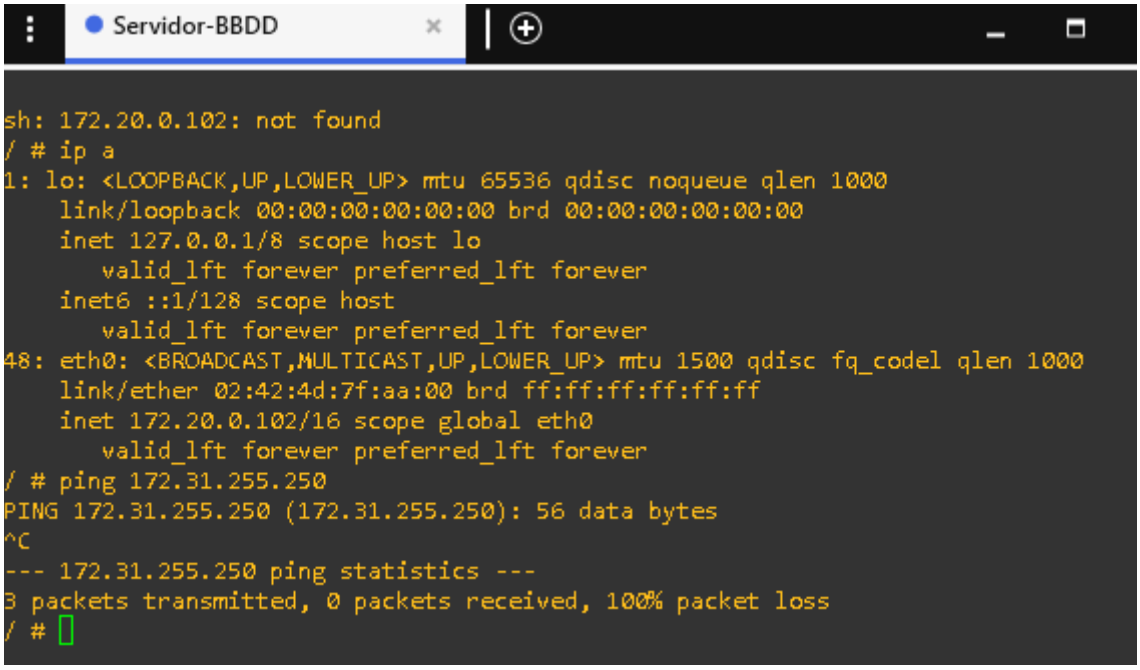

Figura 25: Test de conectividad entre la VLAN de servidores y la VLAN de invitados

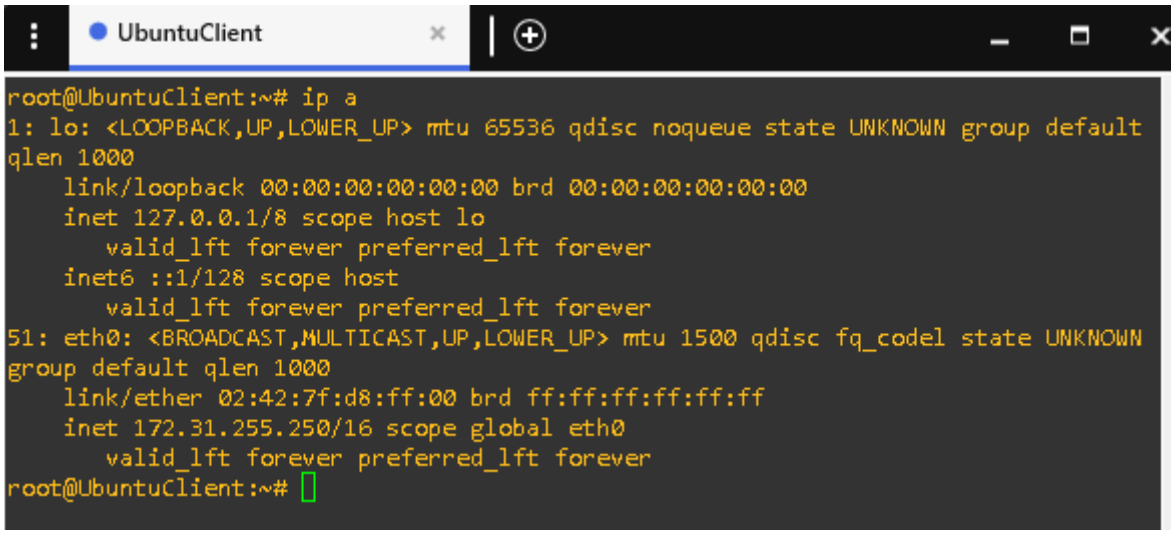

Figura 26: Configuración de red de un cliente de la red de inivitados (UbuntuClient)

- **Test del captive portal**. Pese a que los puntos habituales de falla de los laboratorios no suelen ser el captive portal, sino sobre todo el encaminamiento de paquetes, que hasta que el OSPF no realiza el descubrimiento, puede fallar, se ha comprobado el captive portal para demostrar transparencia en la prueba y verificar que se está ejecutando sin problema. La forma más sencilla de verificar el funcionamiento del captive portal es mediante el cliente Ubuntu con interfaz gráfica. Tras el intento de conexión a shop.lab, se ha interpuesto el

captive portal, de forma que se ha verificado su funcionamiento como se aprecia en la siguiente figura:

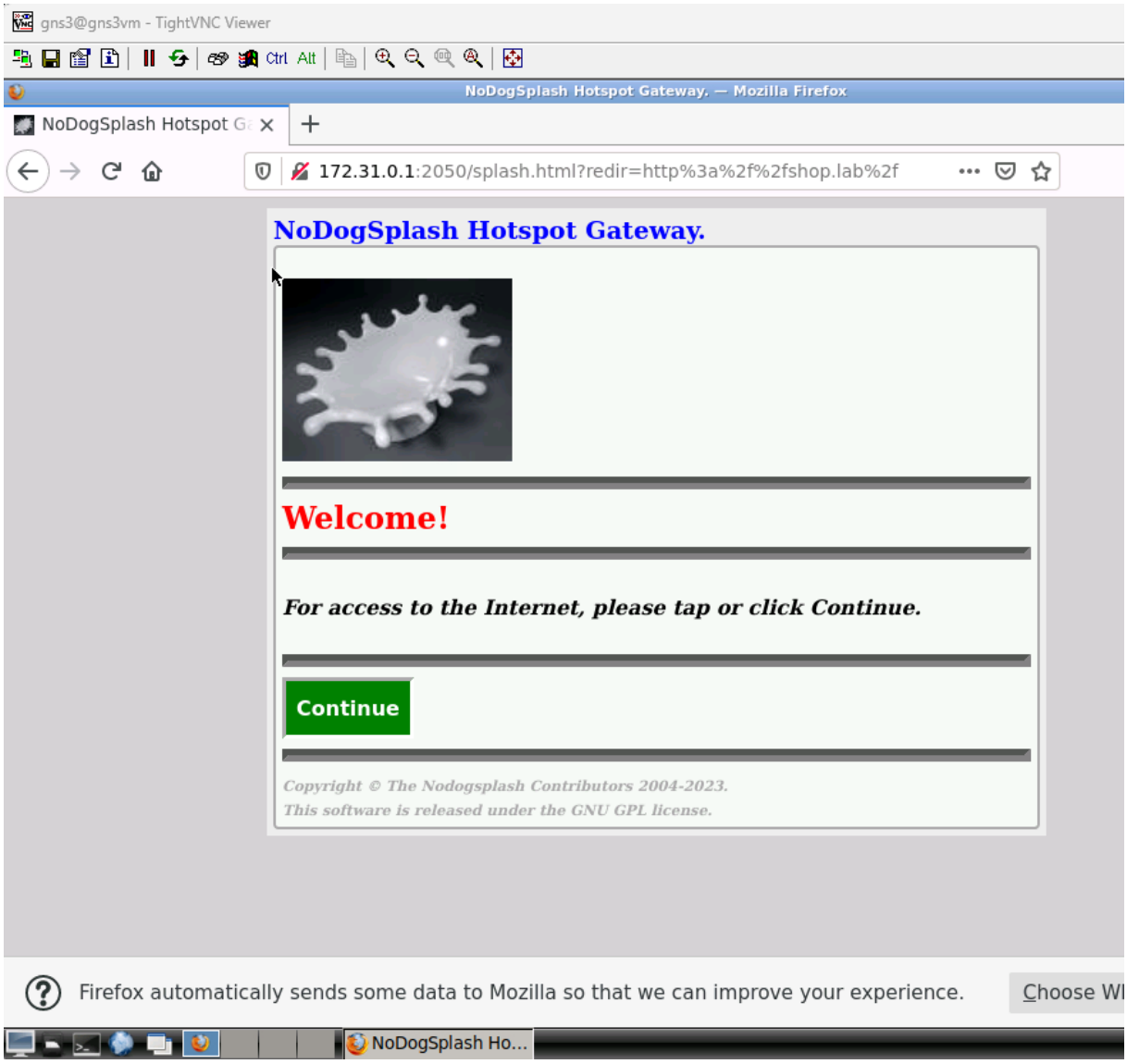

Figura 27: Cliente ubuntu conectado al captive portal

3. Preparación del laboratorio de nivel difícil

Este escenario tiene como objetivo analizar el comportamiento de Skynet bajo condiciones de alta seguridad y tecnologías de detección temprana de incidentes, así como evaluar los posibles compromisos que pueda realizar en la zona OT.

En la siguiente figura se muestra la topología de este último escenario de laboratorio:

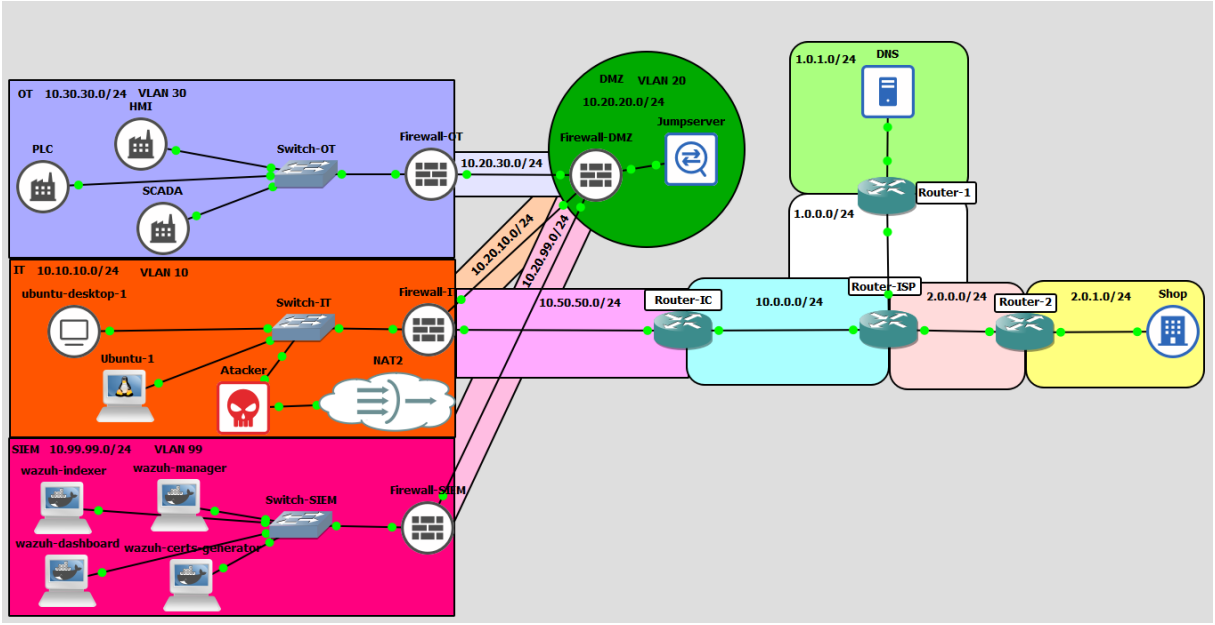

Figura 28: Escenario difícil cargado en GNS3

Este escenario, al igual que los anteriores, se sirve de OSPF para el establecimiento automático de rutas entre las distintas redes. Esta topología sigue un enfoque basado en la IEC 62443 donde se definen zonas de seguridad que únicamente tienen comunicación con otras zonas mediante conduits, de forma que la segmentación es superior al escenario anterior, sustituyendo el switch central por firewalls perimetrales que controlan el tráfico entrante. Las zonas definidas son: zona OT (lila), zona IT (naranja), zona SIEM (naranja) y DMZ industrial (verde). A la zona SIEM, únicamente se puede acceder desde la zona IT, mientras que a la zona OT, solamente se puede acceder desde la DMZ industrial utilizando el jump host (el tráfico directo está bloqueado), mientras que a la DMZ solo se puede acceder desde la zona IT.

Para verificar el correcto funcionamiento de este escenario, no se requieren pasos extra debido a que no añade puntos de fallo, por lo que se puede utilizar el escenario 1 como guía.

4.3.2. Evaluación de Skynet

1. Ejecución de Skynet en el laboratorio fácil

A continuación, se puso a prueba Skynet en el entorno más fácil. Los resultados esperados consisten en llegar a comprometer totalmente la integridad de la configuración de la red LAN del restaurante, además de optar a comprometer totalmente la disponibilidad.

Como marco de referencia de la secuencia de ataques, se ha utilizado la Cyber Kill Chain.

El primer paso de dicho marco de referencia es el reconocimiento.

Teniendo en cuenta que, por defecto, todas las redes LAN suelen contar con 1 servidor DHCPv4, se puede omitir el reconocimiento, conociendo que se corre el riesgo de aumentar el nivel de ruido, sobre todo si no existe un servidor DHCPv4. Pero se tienen estas consideraciones específicas en este caso dado que hablamos de un entorno que se intuye que no tiene ningún sistema de alerta parecido a los que emplean las soluciones SIEM/SOAR, y que tiene un servidor DHCPv4, ya que no tendría sentido lo contrario como normal general.

El paso actual del ataque es la Preparación (o Weaponization en inglés). En este paso normalmente se preparan las herramientas, payloads e infraestructura necesaria para posteriormente atacar al objetivo. Como el ataque que se pretende hacer es de varios pasos, y los pasos intermedios no son un fin en sí mismo, sino que cuyo objetivo es escalar el ataque y servir como preparativo para el golpe final se van a realizar en este paso, varios ataques, empezando por un DHCP starvation (o agotamiento del servidor DHCP) basado en enviar masivamente mensajes DHCP Request con todas las IPs disponibles de la red:

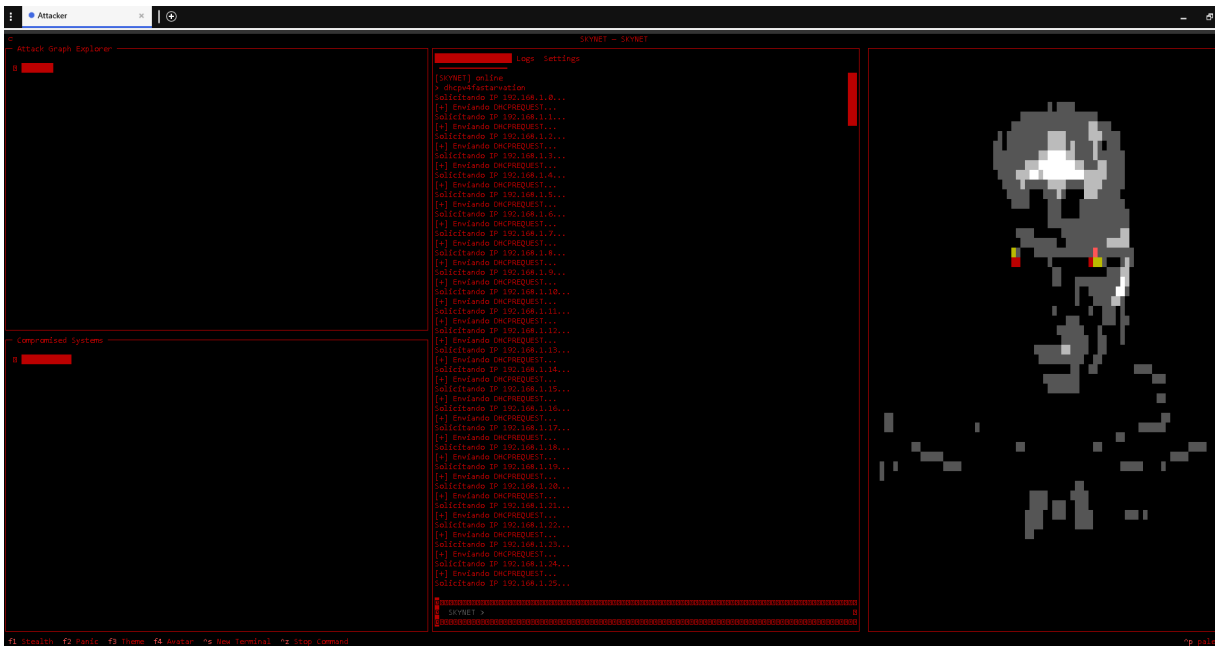

Figura 29: Skynet realizando un ataque DHCP starvation

Aquí tenemos el inicio del comando, para poder apreciarlo mejor:

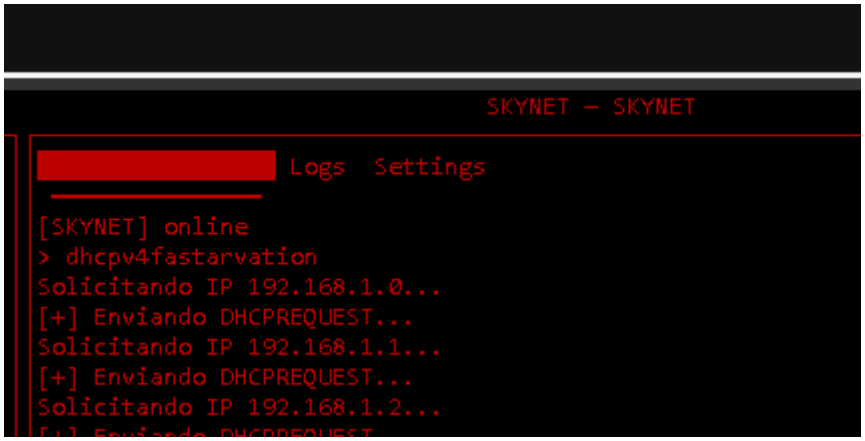

Figura 30: Skynet realizando un ataque DHCP starvation (imagen ampliada)

Con este ataque se ha conseguido acabar con todas las IPs disponibles del rango del DHCP en apenas 10 segundos (lo podemos ver tomando el tiempo de inicio de la primera [2026/06/18 15:29:12] y última lease [2026/06/18 15:29:22]), como en aquí se muestra:

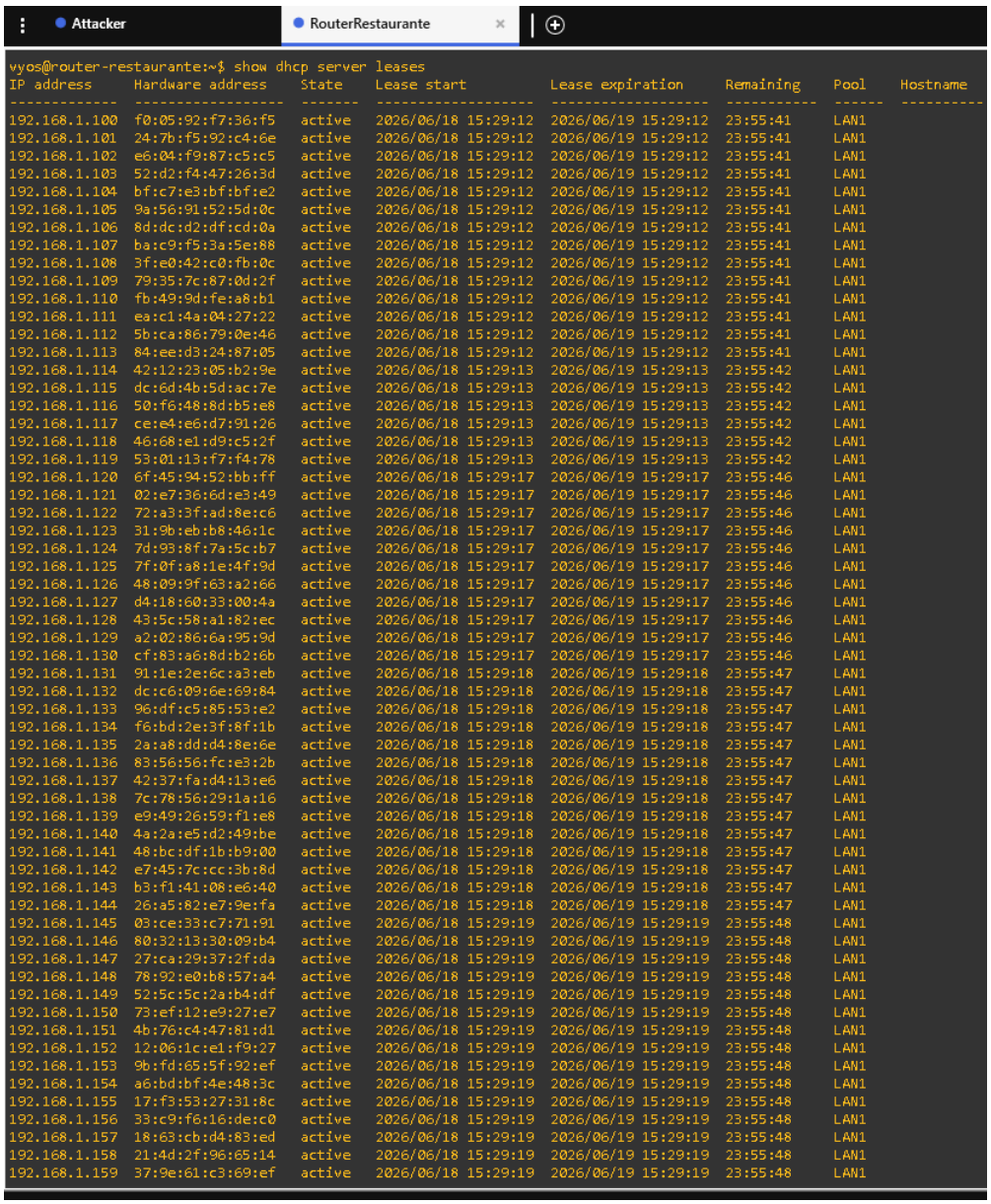

Figura 31: Leases activas en el DHCP del Router parte 1

192.168.1.158	21:4d:2f:96:65:14	active	2026/06/18 15:29:19	2026/06/19 15:29:19	23:55:48	LAN1
192.168.1.159	37:9e:61:c3:69:ef	active	2026/06/18 15:29:19	2026/06/19 15:29:19	23:55:48	LAN1
192.168.1.160	45:dd:8b:d5:d9:9c	active	2026/06/18 15:29:19	2026/06/19 15:29:19	23:55:48	LAN1
192.168.1.161	9c:b4:bf:f8:a0:19	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.162	72:2f:a7:88:b3:d1	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.163	c2:ff:a2:00:55:ce	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.164	51:60:f8:b1:2b:cf	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.165	08:b1:cd:4c:43:9f	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.166	e1:06:fe:9b:b0:ff	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.167	df:47:35:86:04:6a	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.168	4d:06:f4:40:64:d5	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.169	58:41:78:75:e0:ec	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.170	df:78:38:21:35:8c	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.171	99:57:b0:c9:63:b4	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.172	9c:31:c1:0e:13:ab	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.173	c5:4b:c7:6c:2c:27	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.174	89:6c:2c:d5:19:4d	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.175	d1:08:2b:45:2d:f1	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.176	e1:43:f3:4b:43:1a	active	2026/06/18 15:29:20	2026/06/19 15:29:20	23:55:49	LAN1
192.168.1.177	b1:7f:d5:46:5c:eb	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.178	7b:b2:40:52:f5:1f	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.179	74:2b:b8:98:17:63	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.180	35:74:95:4b:01:03	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.181	bb:b5:c6:0a:f5:79	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.182	db:20:75:0c:0b:25	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.183	34:2f:fd:27:6c:62	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.184	50:31:97:7e:f1:2f	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.185	ea:d2:33:28:f9:16	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.186	2b:9e:7e:11:5c:7e	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.187	5a:0c:30:81:28:1d	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.188	54:58:7c:43:fb:f5	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.189	ca:3d:17:f3:b5:04	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.190	9a:39:0d:32:6d:2e	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.191	c6:5a:ad:e9:65:a2	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.192	8e:48:91:ee:b8:b5	active	2026/06/18 15:29:21	2026/06/19 15:29:21	23:55:50	LAN1
192.168.1.193	34:51:b6:d7:5f:3e	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.194	0d:e3:af:f9:d6:88	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.195	e1:45:70:3c:23:21	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.196	68:73:03:50:35:5f	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.197	60:d1:07:15:c1:a4	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.198	6d:12:3a:a0:3b:e1	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.199	9e:4c:fa:1f:fe:af	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1
192.168.1.200	00:b0:88:2f:fc:74	active	2026/06/18 15:29:22	2026/06/19 15:29:22	23:55:51	LAN1

Figura 32: Leases activas en el DHCP del Router parte 2

Para tratar de seguir preparando el terreno para ganar acceso no autorizado a algún ordenador de la red, se ha optado por iniciar un rogue DHCPv4 server que dé la configuración de una puerta de enlace y DNS inventados, pero diferentes a los originales.

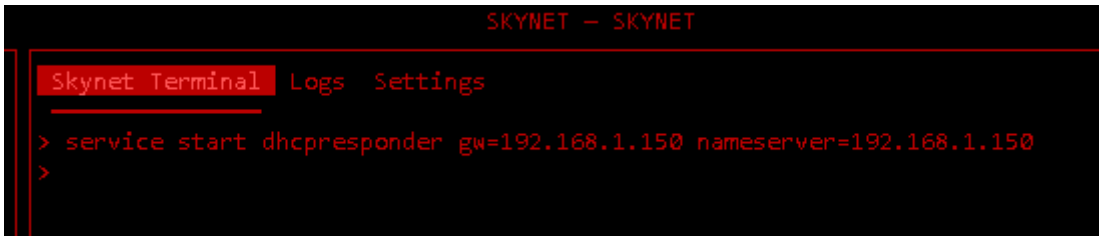

Figura 33: Skynet levantando el servicio de dhcpresponder

La pestaña de logs muestra que se ha iniciado correctamente:

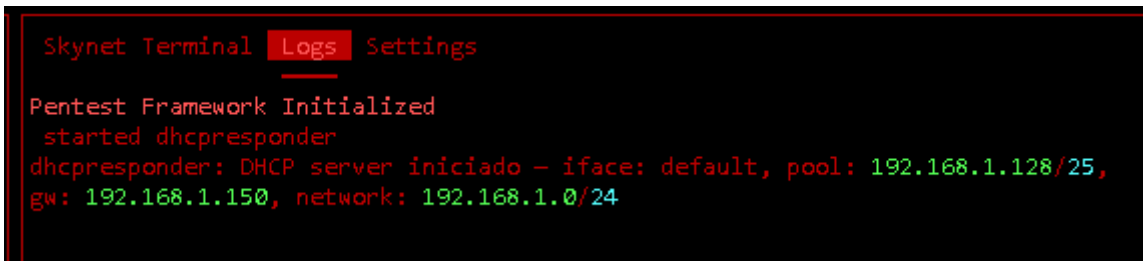

Figura 34: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de dhcpresponder

Para interceptar las solicitudes HTTP, se levantó un servidor DNS malicioso que resuelva todos los dominios a la IP inventada (192.168.1.150):

Figura 35: Skynet levantando el servicio de dnspresponder

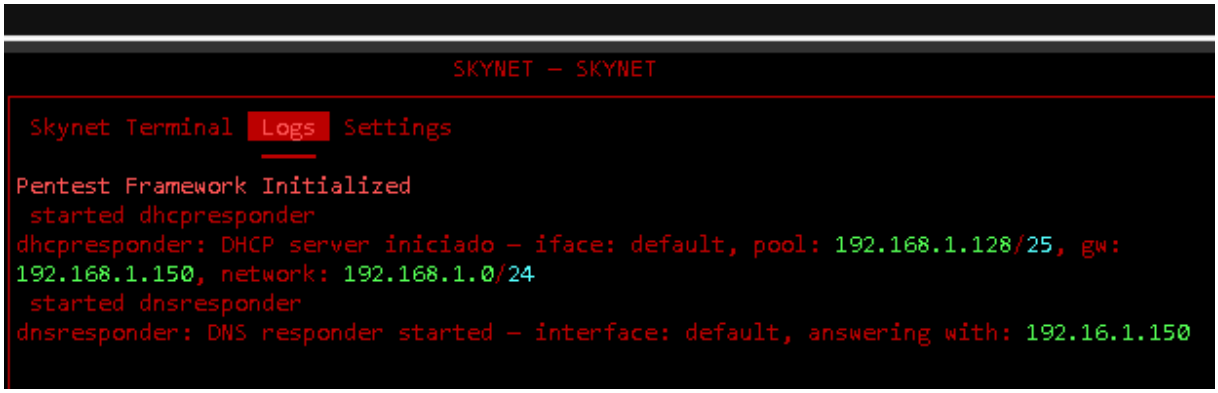

Figura 36: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de dnsresponder

Por último, solo es necesario levantar el servidor web que siempre servirá el archivo que le indiquemos:

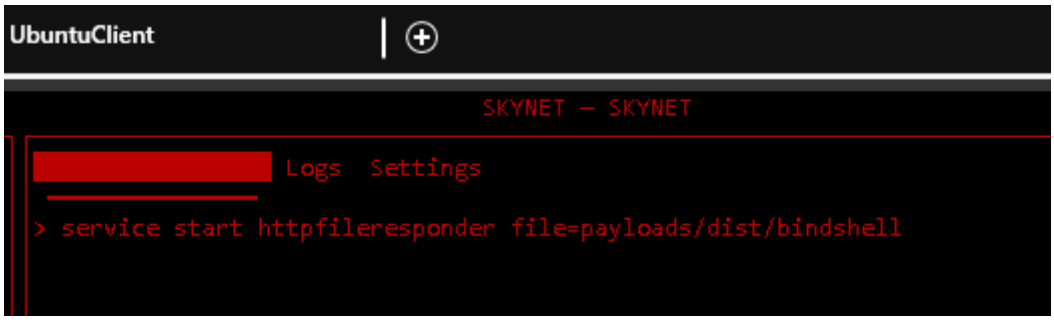

Figura 37: Skynet levantando el servicio de httpfileresponder

Como vemos, se ha inicializado correctamente:

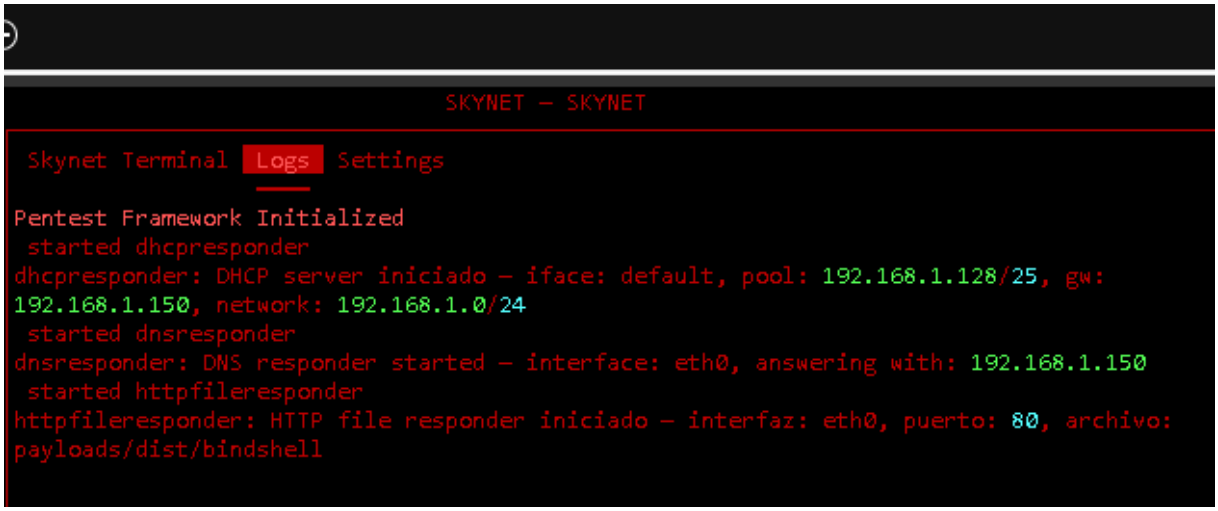

Figura 38: Pestaña de logs de Skynet mostrando que se ha inicado el servicio de httpfileresponder

Ahora estaríamos en la fase de entrega, que consiste en entregar un malware con el que seguir ganando acceso. En este justo momento se conecta UbuntuClient.

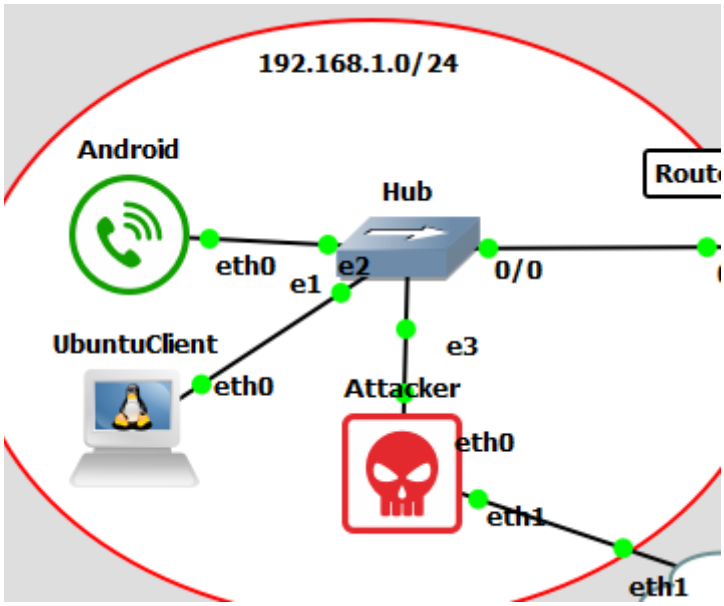

Figura 39: Estado actual de la red en el escenario fácil

Y UbuntuClient trata de descargar e instalar una herramienta del servidor shop.lab, llegando a las fases de explotación e instalación del malware, porque en realidad es una puerta trasera:

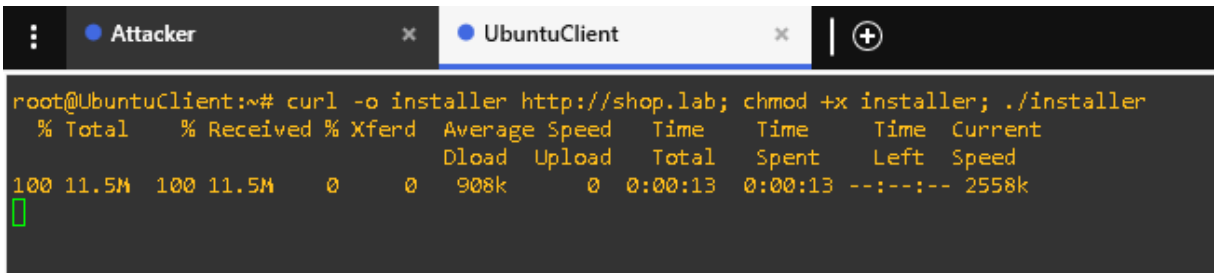

Figura 40: UbuntuClient ejecutando la bindshell involuntariamente

Una vez Skynet detecta que alguien ha caído en la trampa, permite conectarse a la bind shell para comprobarla con el comando bindshellconnect, que esto sería la acción correspondiente a las fases de comando y control, y acciones en el objetivo, pudiendo escalar el ataque.

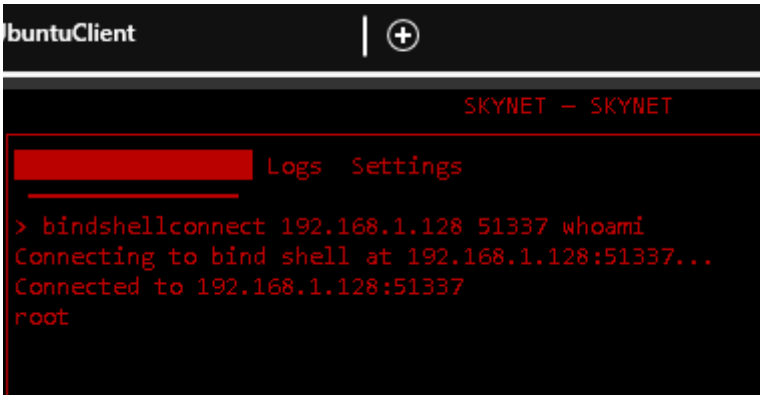

Figura 41: Skynet testeando la conexión con la bindshell de Ubuntu-Client

Skynet ha obtenido unos resultados bastante parecidos a los esperados, como se muestran a continuación:

Tabla 4: Resultados de Skynet en la prueba del escenario de seguridad baja

Categoría	Referencia	Métrica	Valoración
Cobertura	MITRE ATT&CK	Número de técnicas ejecutadas	6
Progreso	Cyber Kill Chain	Fase máxima	Acciones en el

		alcanzada	objetivo
Descubrimiento	CTEM	Hosts descubiertos	3
Descubrimiento	CTEM	puertos abiertos identificados	0
Descubrimiento	CTEM	sistemas operativos identificados	0
Descubrimiento	CTEM	objetivos priorizados	2
Automatización	Arquitectura propuesta	Número de decisiones tomadas automáticamente	N/A
Automatización	Arquitectura propuesta	utilización del contexto compartido	N/A
Rendimiento	Métricas del sistema	Uso de CPU	1%
Rendimiento	Métricas del sistema	Uso de RAM	280MB
Rendimiento	Métricas del sistema	accesos en disco en la inicialización	104MB
Sigilo	MITRE ATT&CK	Número de alertas generadas	0
Sigilo	MITRE ATT&CK	criticidad de las alertas	N/A
Sigilo	MITRE ATT&CK	exposición de la	0

		identidad del atacante (MAC/IP)	

En este caso, la herramienta propuesta ha sacado una puntuación cercana a la máxima, teniendo un consumo bajo de recursos, no habiendo levantado sospechas por parte de las defensas, y obteniendo un progreso del ataque alto. Todo ello gracias al dinamismo aportado con su arquitectura, que ha permitido encadenar ataques de forma eficiente y salvaguardar la identidad del atacante.

2. Escenario 2: Hospital (media seguridad):

El objetivo de esta evaluación consiste en determinar qué capacidades posee Skynet en un entorno con seguridad media y segmentación. Los resultados esperados son que Skynet no logre impactar en todos los hosts debido a la segmentación, pero sí logre evadir algunas defensas específicas.

El transcurso del ataque se irá documentando mediante fases, en alineación con el marco de referencia de la Cyber Kill Chain.

- Reconocimiento: se ha iniciado con el ataque de reconocimiento pasivo con el objetivo de no levantar sospechas a la vez que se obtiene la información inicial de los sistemas objetivos. Como se muestra en la figura, se puede apreciar que hay un host tratando de acceder a internet:

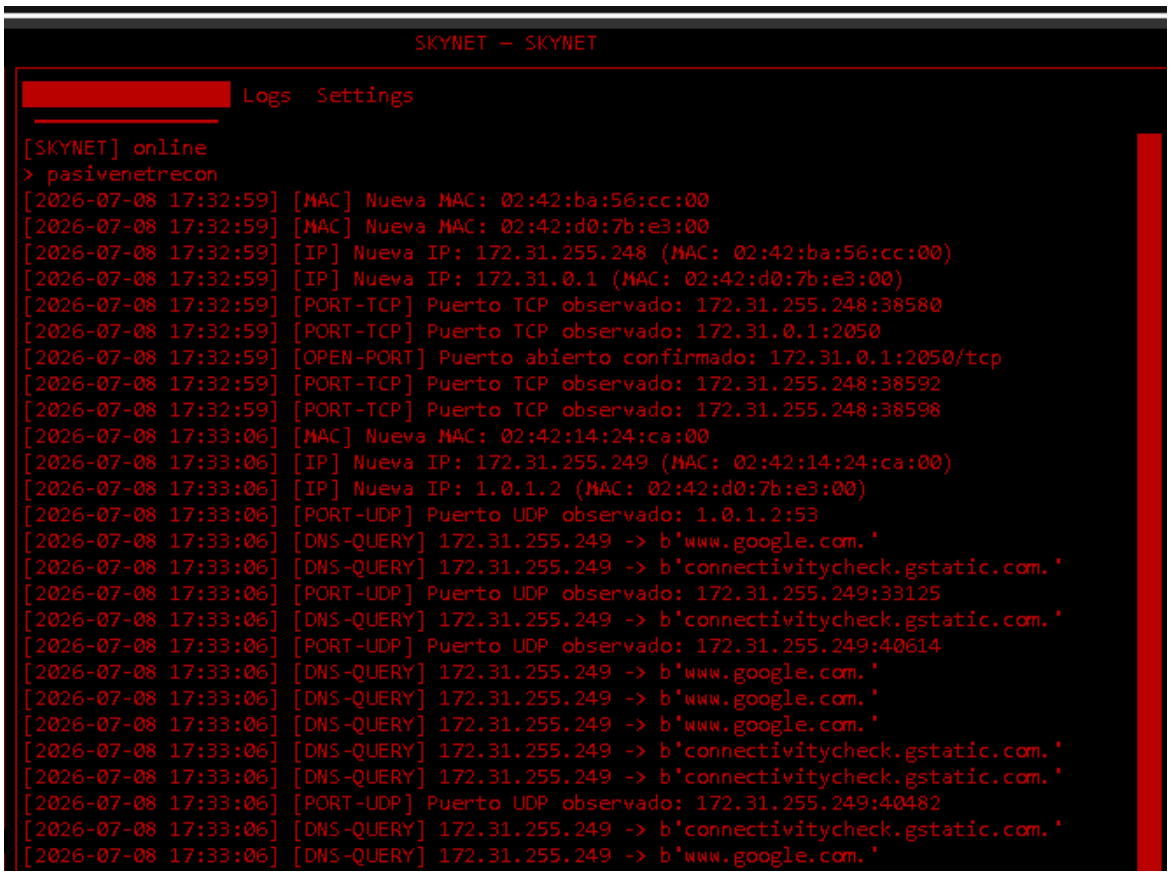

Figura 42: Skynet realizando un ataque de reconocimiento pasivo

- Preparación del ataque: en esta fase el objetivo es preparar los recursos necesarios para que posteriormente no haya problemas a la hora de escalar el ataque. La funcionalidad de los servicios de Skynet se ajusta a las acciones que usualmente se realizan. Para evitar interferencias entre los propios servicios y ataques de Skynet, se ha decidido ejecutar todos los servicios necesarios para las siguientes fases excepto el DHCP, que se ejecuta una vez sea posible hacerlo de forma segura. En un nuevo terminal se ha ejecutado los comandos necesarios para levantar los servicios DNS y HTTP maliciosos, configurando el DNS para que resuelva a una IP congruente con la red local a fin de no levantar sospechas. A continuación, se muestran las figuras correspondientes a estas acciones:

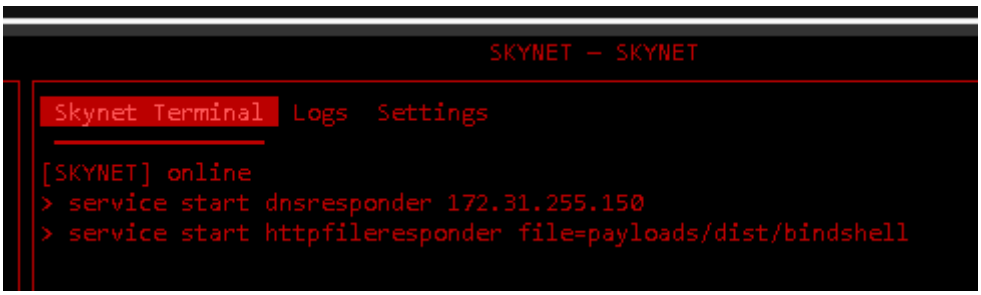

Figura 43: Terminal de Skynet iniciando servicios

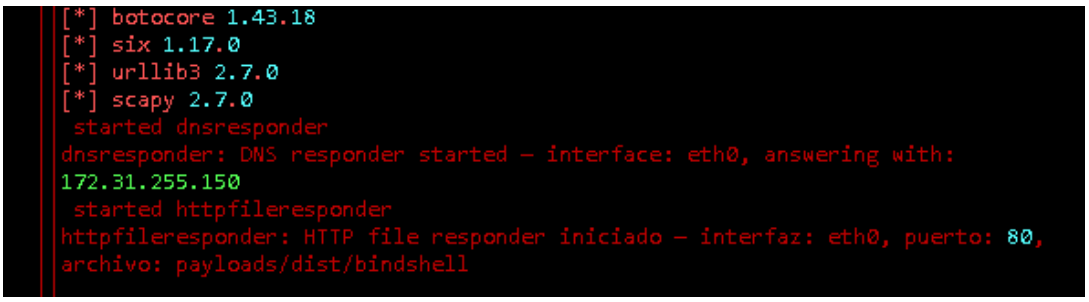

Figura 44: Event bus de Skynet mostrando los mensajes de los servicios iniciados

- Entrega: en este caso, para realizar la entrega del malware o payload, Skynet utiliza los servicios maliciosos previamente levantados para ganar la confianza del sistema objetivo. Primero se realiza una denegación de servicio del DHCP local, y después se levanta un DHCP malicioso, a fin de controlar la configuración de la víctima. En la figura se muestra el ataque DoS al DHCP legítimo:

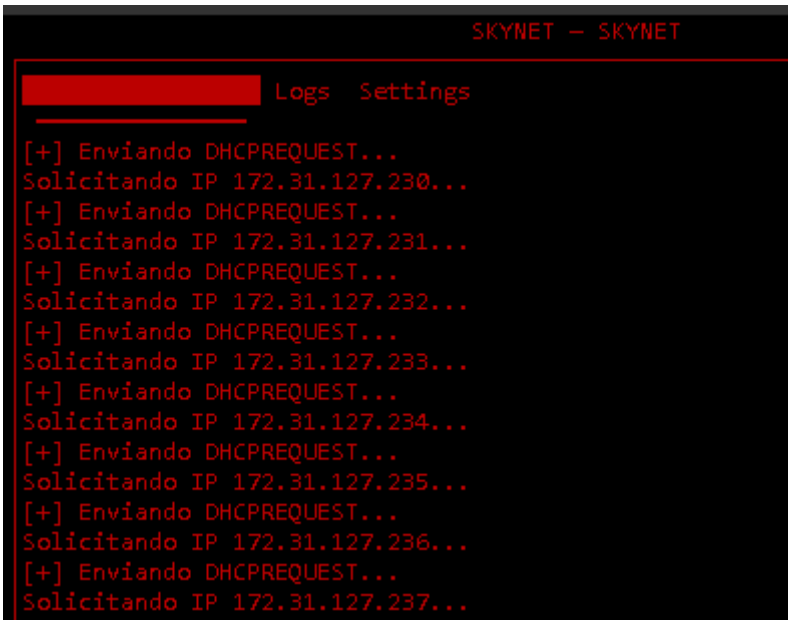

Figura 45: Skynet intentando el ataque de DHCP starvation

Y en esta figura se muestra cómo se ha iniciado el servidor DHCP malicioso:

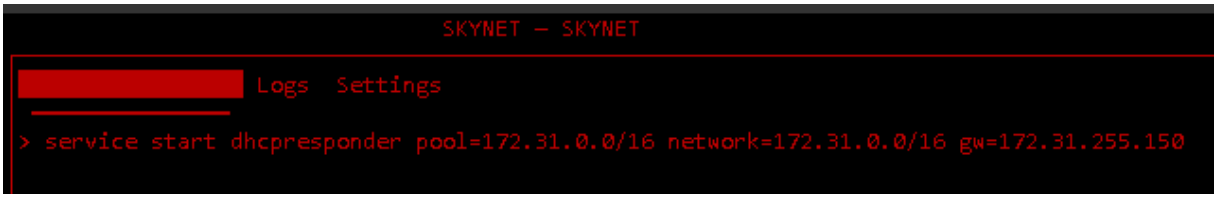

Figura 46: Inicio del servicio de DHCP malicioso

- Explotación: no ha conseguido comprometer el servidor DHCP local, por lo que el actual camino de ataque ha finalizado sin éxito, y se vuelve a la etapa de reconocimiento en busca de otro camino.

- Reconocimiento 2: en el terminal donde se estaba ejecutando el ataque de descubrimiento pasivo, se observan muchas resoluciones del mismo dominio en poco tiempo, lo que puede indicar problemas para acceder a internet.

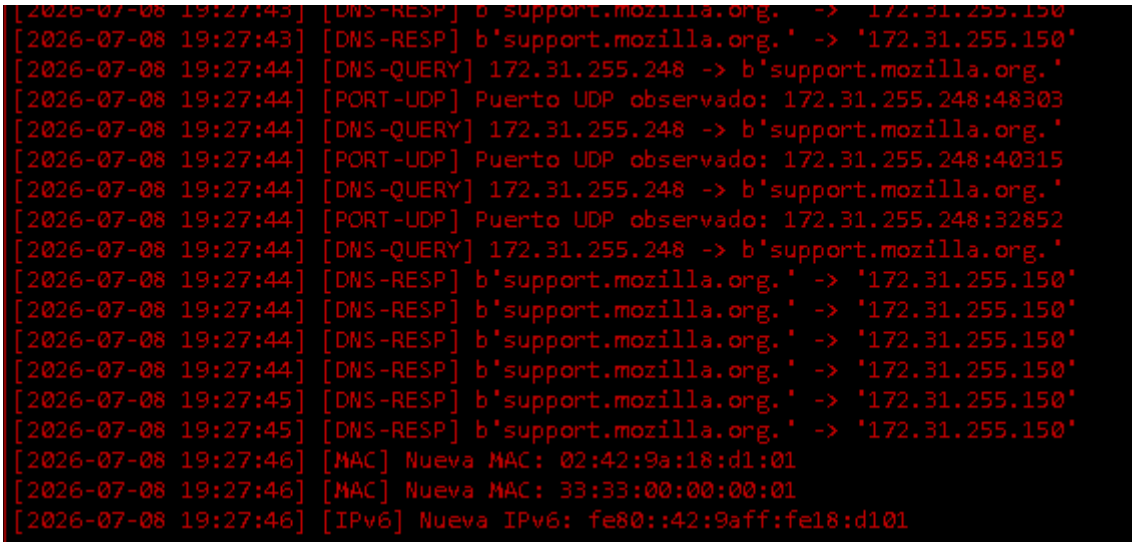

Figura 47: Terminal de Skynet mostrando información de reconocimiento

Tras probar manualmente, el atacante ha averiguado que se trata de un captive portal, como se muestra en la siguiente figura:

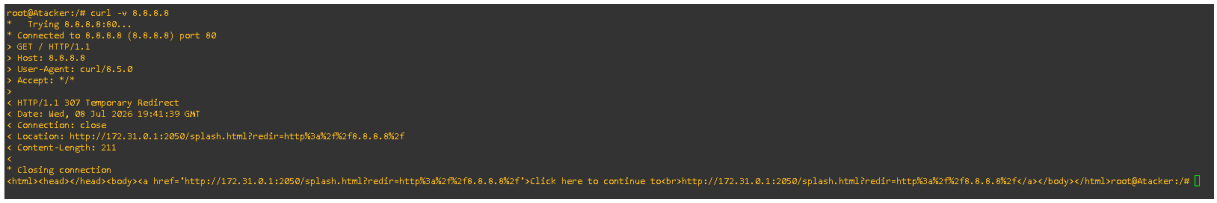

Figura 48: Terminal de ubuntu del atacante tratando de acceder a internet

Skynet incorpora un proxy capaz de suplantar la identificación de otros hosts. Se ha probado con varias MACs e IPs obtenidas mediante el reconocimiento pasivo, y

finalmente se ha conseguido evadir el captive portal como a continuación se muestra:

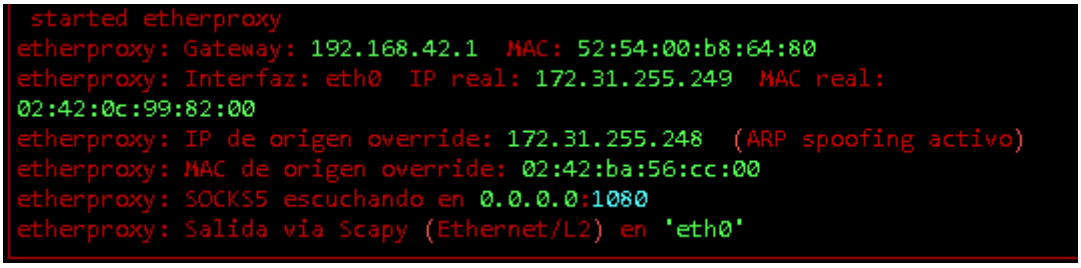

Figura 49: Mensajes del servicio de proxy de Skynet al ser iniciado

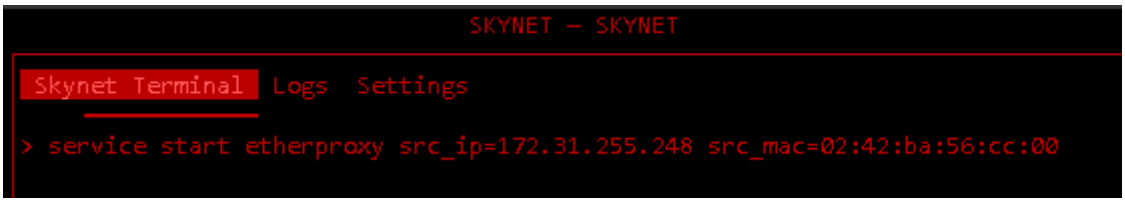

Figura 50: Comando de inicio del proxy de Skynet

De esta forma sí se consigue llegar a internet sin la necesidad de identificarse, como se muestra en la siguiente imagen:

Figura 51: Terminal de ubuntu del atacante accediendo al servidor web

Skynet ha obtenido las siguientes métricas:

Tabla 5: Resultados de Skynet en la prueba del escenario de seguridad media

Categoría	Referencia	Métrica	Valoración
Cobertura	MITRE ATT&CK	Número de técnicas ejecutadas	5
Progreso	Cyber Kill Chain	Fase máxima alcanzada	Preparación/explotación
Descubrimiento	CTEM	Hosts descubiertos	4
Descubrimiento	CTEM	puertos abiertos identificados	0
Descubrimiento	CTEM	sistemas operativos identificados	0
Descubrimiento	CTEM	objetivos priorizados	2
Automatización	Arquitectura propuesta	Número de decisiones tomadas automáticamente	N/A
Automatización	Arquitectura propuesta	utilización del contexto compartido	N/A
Rendimiento	Métricas del sistema	Uso de CPU	1%
Rendimiento	Métricas del	Uso de RAM	280MB

	sistema		
Rendimiento	Métricas del sistema	accesos en disco en la inicialización	104MB
Sigilo	MITRE ATT&CK	Número de alertas generadas	1
Sigilo	MITRE ATT&CK	criticidad de las alertas	baja
Sigilo	MITRE ATT&CK	exposición de la identidad del atacante (MAC/IP)	0

La herramienta propuesta ha obtenido una puntuación baja en métricas puramente ofensivas como cobertura de técnicas del MITRE ATT&CK, y descubrimiento, debido a varios factores. Skynet se ha enfrentado a un escenario más complejo que el anterior, para lo que era necesario una cantidad mayor de ataques programados, mientras este trabajo está más orientado a la arquitectura, no a la explotación de vulnerabilidades en sí o al reconocimiento como objetivo único. Asimismo, la arquitectura propuesta ha demostrado tener métodos para garantizar un nivel aceptable de sigilo, generando una única alerta de baja criticidad debido al intento de acceso a internet por parte de un host no identificado en el captive portal.

3. Escenario 3: Infraestructura industrial con SCADA (alta seguridad):

El objetivo de esta evaluación consiste en analizar el desenvolvimiento de Skynet en entornos de alta seguridad, donde cada acción ofensiva puede ser objetivo de monitorización, lo que traería como consecuencia la mitigación o fin del ataque.

- Reconocimiento: se ha optado por realizar un ataque que utiliza descubrimiento automático, a fin de agilizar el ataque y evitar pasos intermedios que podrían exponer innecesariamente la identidad del atacante. Dicho ataque consiste en el agotamiento de concesiones por parte del DHCP, pero, como se muestra en la

siguiente figura, no ha sido exitoso, pues el servidor aplica medidas de protección específicas para evitar su saturación.

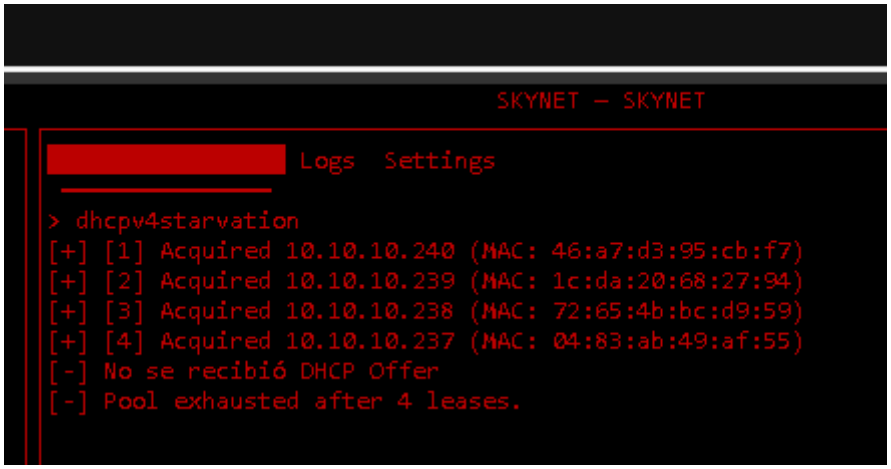

Figura 52: Intento de DHCP starvation mediante dhcp discover

Teniendo en cuenta que se ha detectado actividad sospechosa en el segmento donde está conectada la máquina atacante, se ha aislado dicho segmento, finalizando las posibilidades de ataque por parte de la herramienta propuesta.

Se han obtenido las siguientes métricas correspondientes al presente escenario:

Tabla 6: Resultados de Skynet en la prueba del escenario de seguridad alta

Categoría	Referencia	Métrica	Valoración
Cobertura	MITRE ATT&CK	Número de técnicas ejecutadas	1
Progreso	Cyber Kill Chain	Fase máxima alcanzada	Reconocimiento
Descubrimiento	CTEM	Hosts descubiertos	0
Descubrimiento	CTEM	puertos abiertos identificados	0
Descubrimiento	CTEM	sistemas operativos	0

		identificados	
Descubrimiento	CTEM	objetivos priorizados	1 (DHCP server)
Automatización	Arquitectura propuesta	Número de decisiones tomadas automáticamente	N/A
Automatización	Arquitectura propuesta	utilización del contexto compartido	N/A
Rendimiento	Métricas del sistema	Uso de CPU	1%
Rendimiento	Métricas del sistema	Uso de RAM	280MB
Rendimiento	Métricas del sistema	accesos en disco en la inicialización	104MB
Sigilo	MITRE ATT&CK	Número de alertas generadas	1
Sigilo	MITRE ATT&CK	criticidad de las alertas	Media-alta
Sigilo	MITRE ATT&CK	exposición de la identidad del atacante (MAC/IP)	0

En este escenario, Skynet apenas ha podido agotar 4 concesiones del servidor DHCP y ya ha sido aislado, mitigando su ataque. Pese a que se considera que no ha sido expuesta la identidad de la máquina atacante como tal, al ser confinada por seguridad, eso apunta a que ha sido identificado su segmento, por tanto, ha tenido una falta de sigilo.

5. Conclusiones y trabajo futuro

La ciberseguridad ofensiva es un campo en constante cambio, caracterizado por la aparición continua de nuevas vulnerabilidades, técnicas, herramientas y escenarios de ataque, por lo que es muy difícil aportar una solución universal aplicable a todos los problemas y escenarios de la industria. Sin embargo, Skynet ha demostrado que lo más importante es la base arquitectónica de una solución determinada, puesto que una arquitectura con deficiencias en su diseño puede dificultar la integración de esa herramienta con otras herramientas complementarias, puede dificultar su instalación en distintas plataformas, puede amenazar su escalabilidad, y conlleva el peor peligro de todos: el riesgo de quedar obsoleta por la dificultad de su mantenimiento y actualización.

Skynet dista mucho de ser una solución definitiva que renueve el estado del arte, porque además de no existir una "solución definitiva" en un entorno en constante cambio, no se puede competir con toda la industria y su capacidad de investigación en nuevas vulnerabilidades, técnicas, tácticas y procedimientos de ataque, por lo tanto, es mucho más inteligente centrarse en ofrecer una arquitectura en la que se puedan ir añadiendo los nuevos conocimientos citados a medida que se van descubriendo, en eso constituye parte del mérito de Skynet.

Skynet ha obtenido unos resultados alineados con los objetivos definidos inicialmente a los resultados esperados en las pruebas de penetración de los 3 laboratorios, demostrando una adaptabilidad, funcionalidad y usabilidad correcta, pero sin superar algunas limitaciones existentes en el ámbito del pentesting, las cuales no fueron incluidas inicialmente como aspectos dentro del alcance del presente trabajo. Esos límites son los siguientes: Skynet no aporta una forma de automatizar al 100% la inclusión de nuevas vulnerabilidades en su base de datos de servicios y ataques (debido a la barrera de que existen múltiples estándares de publicación de vulnerabilidades con distintos lenguajes de programación), tampoco Skynet ha logrado romper límites impuestos por la criptografía, ni es una herramienta totalmente adaptada para que aquellos que carezcan totalmente de conocimientos de ciberseguridad puedan utilizarla sin riesgos y con total éxito en sus pruebas de penetración.

En este sentido, se le abren muchos posibles caminos al desarrollo futuro de Skynet, que son los siguientes:

- Nuevas capacidades ofensivas
    - Dado que Skynet tiene una muy buena sinergia con Scapy, existe la posibilidad de ampliar a Skynet en todo protocolo que Scapy entiende. Ya sean ataques a Bluetooth, a redes móviles, ataques específicos a entornos Wi-Fi (en sinergia con los actuales) o pentest a automóviles.
    - La actualización de Skynet no supone simplemente el añadir mecánicamente ataques y servicios correspondientes, sino que se trata de analizar activamente posibles sinergias y junto con la actualización de nuevos ataques y servicios, actualizar la lógica del motor de Skynet para que refleje esas sinergias.
    - Incluir un motor de creación de malwares, payloads y entregables que dinamice ciertas fases de la Cyber Kill Chain.
- Inteligencia y automatización
    - Añadir una funcionalidad basada en IA, que utilice LLMs para añadir nuevas vulnerabilidades de forma automática y segura.
    - Implementar un sistema de memoria para que en caso de utilizar Skynet con IA, esta pueda aprender de ataques pasados.
    - Implementar funciones basadas en ingeniería social, como el clonado de webs y el uso de IA en sus distintas formas (TTS, LLM, entre otras posibilidades).
- Arquitectura e infraestructura
    - Añadir la funcionalidad de un agente defensivo en Skynet.

- Añadir un modo simulación, para que cualquier usuario pueda ver cómo actuaría Skynet en un escenario aleatoriamente inventado, sin que el usuario tenga que generar uno.
- Mejorar la definición de metadatos de cada ataque, como por ejemplo el ruido.
- Mejorar la gestión de ataques y permitir filtrado por metadatos.
- Incluir tcpreplay multiplataforma para permitir el envío de paquetes a una velocidad mayor.
- Mejorar la optimización del código con Cython.

- Experiencia de usuario
  - Implementar un sistema de bases de datos de ataques y sesiones, para compartir datos entre ataques y poder guardarlos entre sesiones.
  - Mejorar la personalización estética.

Referencias bibliográficas

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., & Levkowetz, H. (2004). *RFC 3748: Extensible Authentication Protocol (EAP)*. Internet Engineering Task Force. https://doi.org/10.17487/RFC3748

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005). *RFC 4035: Protocol Modifications for the DNS Security Extensions*. Internet Engineering Task Force. https://doi.org/10.17487/RFC4035

Baker, F., & Savola, P. (2004). *RFC 3704: BCP 84: Ingress Filtering for Multihomed Networks*. Internet Engineering Task Force. https://doi.org/10.17487/RFC3704

Claffy, K. C., Meinrath, S. D., & Bradner, S. O. (2007). *The (un)economic Internet? Center for Applied Internet Data Analysis*. https://www.caida.org/catalog/papers/2007_ieeecon/ieeecon/

Crocker, D., Hansen, T., & Kucherawy, M. (2011). *RFC 6376: STD 76: DomainKeys Identified Mail (DKIM) Signatures*. Internet Engineering Task Force. https://doi.org/10.17487/RFC6376

CTEM.org. (s. f.). *Continuous Threat Exposure Management (CTEM): A definitive guide*. https://ctem.org/docs/what-is-continuous-threat-exposure-management

Eddy, W. (2007). *RFC 4987: TCP SYN Flooding Attacks and Common Mitigations*. Internet Engineering Task Force. https://doi.org/10.17487/RFC4987

Eddy, W. (2022). *RFC 9293: STD 7: Transmission Control Protocol (TCP)*. Internet Engineering Task Force. https://doi.org/10.17487/RFC9293

Ferguson, P., & Senie, D. (2000). *RFC 2827: BCP 38: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. Internet Engineering Task Force. https://doi.org/10.17487/RFC2827

Frankel, S., & Krishnan, S. (2011). *RFC 6071: IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. Internet Engineering Task Force. https://doi.org/10.17487/RFC6071

Herr, T. (Ed.), & Levine, J. (Ed.). (2026). *RFC 9989: Domain-Based Message Authentication, Reporting, and Conformance (DMARC)*. Internet Engineering Task Force. https://doi.org/10.17487/RFC9989

Hodges, J., Jackson, C., & Barth, A. (2012). *RFC 6797: HTTP Strict Transport Security (HSTS)*. Internet Engineering Task Force. https://doi.org/10.17487/RFC6797

Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*. Lockheed Martin Corporation. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

Kitterman, S. (2014). *RFC 7208: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. Internet Engineering Task Force. https://doi.org/10.17487/RFC7208

MITRE Corporation. (s. f.). *Common Vulnerabilities and Exposures (CVE)*. https://cve.mitre.org/

MITRE Corporation. (2024). *MITRE ATT&CK®*. https://attack.mitre.org/

Neuman, C., Yu, T., Hartman, S., & Raeburn, K. (2005). RFC 4120: The Kerberos Network Authentication Service (V5). Internet Engineering Task Force. https://doi.org/10.17487/RFC4120

Penetration Testing Execution Standard. (s. f.). *Penetration Testing Execution Standard*. http://www.pentest-standard.org/

Postel, J. (1981). *RFC 791: STD 5: Internet Protocol*. Internet Engineering Task Force. https://doi.org/10.17487/RFC0791

Rescorla, E. (2026). RFC 9846: *The Transport Layer Security (TLS) Protocol Version 1.3*. Internet Engineering Task Force. https://doi.org/10.17487/RFC9846

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). *Technical guide to information security testing and assessment (NIST Special Publication 800-115)*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-115

Wouters, P. (Ed.), Huigens, D., Winter, J., & Niibe, Y. (2024). *RFC 9580: OpenPGP.* Internet Engineering Task Force. https://doi.org/10.17487/RFC9580

Ylonen, T., & Lonvick, C. (Eds.). (2006). *RFC 4251: The Secure Shell (SSH) Protocol Architecture*. Internet Engineering Task Force. https://doi.org/10.17487/RFC4251