Maestría en

Ciberseguridad

Trabajo previo a la obtención de título

de Magister en Ciberseguridad

AUTOR/ES:

Jhonatan Damian Flores Yaguana

Kevin German Avalos Trujillo

Guillermo David Villalba Monteros

Fernanda de los Ángeles Granja Espinosa

TUTOR/ES:

Alejandro Cortés López

Iván Reyes Chacón

TEMA: Análisis de vulnerabilidades en drivers de tarjetas de red Wi-Fi mediante técnicas de ingeniería inversa y fuzzing

REINVENTEMOS EL FUTURO

Certificación de autoría

Nosotros, Jhonatan Damian Flores Yaguana, Kevin German Avalos Trujillo, Guillermo David Villalba Monteros, Fernanda de los Ángeles Granja Espinosa declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

---------- ----------

Firma **Firma**

---------- ----------

Firma **Firma**

Autorización de Derechos de Propiedad Intelectual

Nosotros, Jhonatan Damian Flores Yaguana, Kevin German Avalos Trujillo, Guillermo David Villalba Monteros, Fernanda de los Ángeles Granja Espinosa, en calidad de autores del trabajo de investigación titulado ***Análisis de vulnerabilidades en drivers de tarjetas de red Wi-Fi mediante técnicas de ingeniería inversa y fuzzing***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, (enero 2026)

---------- ----------

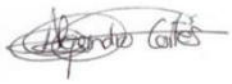
Firma **Firma**

---------- ----------

Firma **Firma**

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés e Iván Reyes,** declaramos que: **Jhonatan Damian Flores Yaguana, Kevin German Avalos Trujillo, Guillermo David Villalba Monteros, Fernanda de los Ángeles Granja Espinosa** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés L.

Maestría en Ciberseguridad

Iván Reyes Ch.

Maestría en Ciberseguridad

DEDICATORIA

Dedicamos este logro a nuestras familias. Gracias por ser el pilar sobre el cual construimos nuestro futuro. Por su amor, paciencia y fe inquebrantable en nosotros.

Con cariño, Jhonnatan, Kevin, Guillermo y Fernanda

AGRADECIMIENTOS

Expresamos nuestro agradecimiento a la Universidad Internacional del Ecuador por brindarnos el espacio y las herramientas para nuestra formación profesional

De manera especial, agradecemos a nuestras familias por ser el pilar fundamental en nuestras vidas. Agradecemos también la excelente colaboración y el trabajo en equipo de los autores de este proyecto, cuya sinergia hizo posible la entrega de este trabajo de titulación con calidad y dedicación.

RESUMEN

Los controladores de tarjetas de red Wi-Fi en sistemas Windows gestionan tramas, comandos de control (IOCTL/OID) y estados del subsistema de red con privilegios elevados. Estas interfaces presentan una superficie de ataque importante ya que errores en la validación de entradas o en el manejo de estados pueden provocar fallos de memoria, condiciones de carrera o denegaciones de servicio, lo cual afecta la confidencialidad, integridad y disponibilidad del sistema. La presente tesis propone un análisis experimental de vulnerabilidades en drivers Wi-Fi para Windows mediante técnicas de fuzzing dirigidas a varios vectores de entrada: tramas 802.11, IOCTLs y OIDs. Se diseñará e implementará un pipeline reproducible que combina generación y manipulación de paquetes, envío de IOCTLs malformados e instrumentación mediante WinDbg y volcados de memoria del kernel. Las campañas de prueba se ejecutarán en un laboratorio aislado con máquinas virtuales y hardware de pruebas, y los fallos detectados se registrarán, analizarán, clasificarán y documentarán con pruebas de concepto reproducibles. Para las pruebas físicas se emplearán adaptadores Wi-Fi externos (USB/PCI-E), dado que las tarjetas internas no permiten de forma fiable la inyección ni el monitoreo de tramas necesarios para la metodología de fuzzing.

Palabras clave: Drivers Wi-Fi, Fuzzing, IOCTL/OID, Vulnerabilidades, Windows (NDIS/Kernel).

ABSTRACT

Wi-Fi network adapter drivers on Windows systems process frames, control commands (IOCTL/OID) and network subsystem states with elevated privileges. These interfaces represent a significant attack surface because improper input validation or incorrect state handling may cause memory corruption, race conditions or denial of service, thereby impacting system confidentiality, integrity and availability. This thesis presents an experimental analysis of vulnerabilities in Wi-Fi drivers for Windows using fuzzing techniques targeted at multiple input vectors: 802.11 frames, IOCTLs and OIDs. A reproducible pipeline will be designed and implemented combining packet generation and manipulation, malformed IOCTL submission and instrumentation using WinDbg and kernel memory dumps. The testing campaigns will run in an isolated lab environment with virtual machines and test hardware; detected faults will be recorded, analysed, classified and documented with reproducible proof-of-concepts. For physical testing, external Wi-Fi adapters (USB/PCI-E) will be used, since internal cards do not reliably support packet injection or monitor modes required by the fuzzing methodology.

Keywords: Wi-Fi Drivers, Fuzzing, IOCTL/OID, Vulnerabilities, Windows (NDIS/Kernel),

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de anexos

CAPITULO 1

INTRODUCCIÓN

1.1 Definición del proyecto

El objetivo central de esta investigación es la evaluación sistemática de vulnerabilidades en drivers de tarjetas de red Wi-Fi, componentes críticos que se ejecutan en sistemas Windows. Se seleccionó la técnica de fuzzing debido a su eficacia para identificar debilidades generadas por entradas malformadas o la transición a estados imprevistos. El proyecto abarca la localización de estas fallas, la meticulosa preparación del ambiente de prueba y el establecimiento de un protocolo metodológico que garantice la reproducibilidad.

En el mundo de la ciberseguridad, los ataques dirigidos a partes de bajo nivel del sistema operativo se han vuelto más comunes, ya que permiten manipular directamente el núcleo (kernel) y eludir las defensas convencionales. Entre ellos, los controladores de dispositivos (drivers) son un punto sensible, ya que se ejecutan en modo privilegiado y controlan directamente el hardware. En particular, los controladores de tarjetas de red Wi-Fi manejan tráfico de redes externas no confiables, lo que amplía enormemente su superficie de ataque. Esto hace que los drivers inalámbricos sean un blanco atractivo para atacantes que quieren causar denegación de servicio, escalada de privilegios o corrupción de memoria aprovechando errores de validación o manejo de estados internos.

La implementación experimental requiere la instalación de un conjunto de drivers que sean representativos de múltiples fabricantes. Estos se desplegarán en máquinas virtuales (Windows 10/11 x64). Posteriormente, se aplicarán campañas de fuzzing. Estas se

focalizarán en dos frentes clave: la capa de tramas de red (protocolo 802.11) y las interfaces de control específicas del sistema (IOCTL/OID). El análisis y la clasificación de los datos recopilados son cruciales, finalizando con la documentación de cada hallazgo mediante la elaboración de Pruebas de Concepto (PoC).

1.2 Justificación e importancia del trabajo de investigación

Los controladores Wi-Fi en Windows constituyen un componente de alto riesgo dentro del sistema operativo debido a que se ejecutan en modo kernel, con acceso directo a la memoria y a recursos críticos del sistema, lo que incrementa el impacto potencial de cualquier falla (Schumilo et al., 2020). Esta ubicación privilegiada implica que un error en su implementación puede derivar en vulnerabilidades severas, incluyendo ejecución remota de código, como se evidenció en el controlador WLAN afectado por la CVE-2023-32067 (Microsoft, 2023), o elevación de privilegios, registrada en la CVE-2023-24963 publicada por la National Vulnerability Database (NIST, 2023). Debido a estos riesgos comprobados, la evaluación sistemática y continua de estos drivers resulta esencial para garantizar la integridad y disponibilidad del sistema.

Las técnicas tradicionales de pruebas en controladores suelen enfocarse en validaciones funcionales o de compatibilidad, pero no siempre logran explorar condiciones extremas o entradas malformadas. En este contexto, el fuzzing ha emergido como una técnica eficaz para descubrir nuevas vulnerabilidades, especialmente en componentes de bajo nivel del sistema operativo (Zhao et al., 2022).

En el plano social, el estudio cobra relevancia debido a la creciente dependencia de redes inalámbricas en contextos educativos, empresariales y de servicios críticos, donde una vulnerabilidad en drivers Wi-Fi podría afectar la continuidad operativa y la protección

de la información. Finalmente, desde el enfoque práctico, la investigación aporta un marco experimental y un conjunto de recomendaciones orientadas a fabricantes, administradores de sistemas y a la comunidad de ciberseguridad, con el fin de fortalecer el desarrollo y mantenimiento de controladores Wi-Fi más robustos para sistemas Windows.

La importancia de estudiar la seguridad de los drivers Wi-Fi viene dada porque las redes inalámbricas son actualmente la principal forma de conectividad en entornos corporativos, educativos y de servicios críticos. La estabilidad y fiabilidad de estos controladores influyen en la continuidad, seguridad de la información y experiencia del usuario. A diferencia de otros drivers, los drivers Wi-Fi tienen que manejar continuamente tramas de gestión, autenticación y control, abriendo la puerta a un gran número de posibles entradas maliciosas o manipuladas (Borrero y Ponce, 2023). Como resultado, un error en el controlador puede comprometer tanto la disponibilidad del servicio como la seguridad del sistema, incluso cuando el punto de acceso o la infraestructura de red no tienen vulnerabilidades conocidas.

Aunque hay mucha investigación sobre vulnerabilidades en controladores del kernel de Windows, la literatura cubre poco sobre drivers de tarjetas de red Wi-Fi, en particular sobre el procesamiento de tramas 802.11 y la manipulación de IOCTL/OID de NDIS. Esta falta crea un agujero en el conocimiento de cómo reaccionan estos controladores ante entradas anormales, secuencias de estados inusuales y condiciones de estrés inherentes a los entornos inalámbricos adversos. Aquí es donde entra en juego la necesidad de probar experimentalmente el comportamiento de los drivers Wi-Fi en Windows con técnicas capaces de revelar errores reproducibles que las pruebas funcionales no logran encontrar (Microsoft, 2024).

1.3 Alcance

Las pruebas se desarrollarán en un entorno controlado conformado por máquinas virtuales que ejecutan Windows 10 (arquitectura x64) junto con hardware de prueba específico compuesto por adaptadores Wi-Fi externos (adaptadores USB y tarjetas PCI-E). Estos dispositivos permiten realizar la captura, generación e inyección de tramas 802.11 necesarias para la metodología de fuzzing. Se justifica esta configuración porque el uso de virtualización aisla el experimento del entorno productivo y facilita la repetibilidad, al tiempo que los adaptadores externos ofrecen el nivel de control de bajo nivel requerido para las pruebas.

Se aclara que las tarjetas Wi-Fi integradas en la mayoría de los equipos no ofrecen, en general, soporte fiable para modos avanzados de monitoreo (promiscuo/monitor) ni para la inyección de paquetes funcionalidades indispensables para el análisis forense del subsistema de red y la metodología de fuzzing orientada a tramas; por ello, el estudio empleará adaptadores externos compatibles con dichas capacidades, seleccionados entre distintos fabricantes para asegurar representatividad. Esta decisión se fundamenta en la necesidad de disponer de un hardware que realmente permita condicionar, alterar y observar el comportamiento del driver en condiciones no estándar.

En el laboratorio se utilizará el software de virtualización VirtualBox para alojar imágenes de prueba de Windows 10/11 y máquinas de soporte/ataque basadas en Kali Linux; los adaptadores Wi-Fi externos serán conectados al host y asignados (passthrough) a las máquinas virtuales cuando sea necesario. Para la inyección de tramas se emplearán utilidades especializadas (por ejemplo, aircrack-ng y otras compatibles con los adaptadores seleccionados), mientras que para la captura y análisis de tráfico se utilizará Wireshark, almacenando las trazas en archivos pcap para su posterior correlación con volcados de

memoria y registros de fallos. Este diseño metodológico se fundamenta en garantizar tanto la precisión como la trazabilidad de los resultados.

Las técnicas de fuzzing aplicadas cubrirán dos vectores principales: primero, la generación de tramas 802.11 malformadas o atípicas, y segundo, el envío de comandos IOCTL/OID alterados al controlador (driver). Los resultados se focalizarán en la detección de fallos reproducibles, su clasificación y la formulación de recomendaciones para mitigarlos. Esta metodología se justifica porque permite evaluar el comportamiento del driver ante entradas anómalas, simulando ataques reales o fallos no contemplados por pruebas funcionales habituales.

El uso de adaptadores Wi-Fi externos y un entorno de laboratorio controlado busca simular condiciones reales para medir el comportamiento real del driver sobre hardware real. Las tarjetas internas de muchos dispositivos no soportan modos de monitorización o inyección de tramas, lo que imposibilita la creación de escenarios de prueba realistas. Por eso, con hardware compatible con técnicas avanzadas de análisis inalámbrico se pueden realizar campañas de fuzzing dirigidas a marcos de gestión, autenticación y descubrimiento de red, garantizando que los resultados reflejen el rendimiento real del controlador en escenarios similares a los de producción.

El estudio no realizará pruebas en redes públicas o productivas, ni modificará el firmware del hardware, ni accederá al código fuente propietario de los drivers. Estos límites éticos y legales se adoptan para garantizar que la investigación se realice con pleno respeto de las normativas vigentes y de los derechos de propiedad intelectual, asegurando que los resultados puedan replicarse sin riesgo de repercusiones legales o de seguridad.

El método experimental de esta investigación se justifica en la necesidad de analizar el comportamiento real de los controladores en situaciones no convencionales que intentan simular escenarios de ataque o tráfico malicioso característico de redes inalámbricas adversas. A diferencia del análisis estático o las pruebas de compatibilidad, el fuzzing puede generar entradas maliciosas, aleatorias o específicas para activar rutas de código raras en el controlador y descubrir errores relacionados con la memoria, la sincronización de estados o la validación de parámetros. Esto es especialmente relevante para drivers Wi-Fi, ya que están constantemente manipulando información del mundo exterior y, por lo tanto, son más susceptibles a encontrarse en situaciones no anticipadas en el momento de su desarrollo.

1.3 Objetivos

1.4.1 Objetivo General

Analizar las vulnerabilidades presentes en los drivers Wi-Fi de sistemas Windows, mediante la aplicación de técnicas de fuzzing orientadas a tramas 802.11 y comandos IOCTL/OID, con el fin de identificar fallos de seguridad reproducibles y proponer medidas preventivas y correctivas que fortalezcan la integridad y disponibilidad del subsistema de red en un entorno de laboratorio controlado.

1.4.2 Objetivos Específicos

- Seleccionar y preparar un conjunto representativo de drivers Wi-Fi para sistemas Windows, instalándolos y configurándolos en un entorno de laboratorio controlado, con el fin de asegurar condiciones precisas y reproducibles para su evaluación.
- Definir e implementar el procedimiento de fuzzing basado en tramas 802.11

y comandos IOCTL/OID, estableciendo los requisitos del ambiente de prueba (volcados de memoria, depuración del kernel y registro de fallos), para garantizar la validez técnica del proceso de análisis.

- Ejecutar y documentar las campañas de fuzzing aplicadas a los drivers seleccionados, registrando los errores detectados, analizando su origen y clasificándolos según severidad, con el propósito de identificar vulnerabilidades reproducibles.

- Formular recomendaciones y buenas prácticas orientadas a fabricantes y administradores de sistemas, a fin de fortalecer la seguridad en el desarrollo y mantenimiento de drivers Wi-Fi en sistemas Windows y reducir los riesgos asociados.

CAPITULO 2

REVISIÓN DE LITERATURA

2.1 Estado del Arte

El análisis de vulnerabilidades en controladores de sistema ha aumentado de forma significativa en los últimos años, especialmente en drivers de red inalámbrica. Este incremento se explica por la creciente complejidad de los controladores y por el hecho de que operan en el núcleo del sistema, lo que les otorga un nivel elevado de privilegios. Estudios recientes muestran que los controladores de Windows presentan fallos explotables derivados de validación insuficiente de entradas y manejo inseguro de memoria (Schumilo et al., 2020). Esta tendencia ha motivado el desarrollo de herramientas y metodologías orientadas a detectar vulnerabilidades en software de bajo nivel, incluyendo componentes responsables de la comunicación Wi-Fi.

Un avance destacado es POPKORN, una herramienta presentada por Gupta et al., (2022), que permite automatizar la evaluación de controladores de Windows sin requerir acceso al código fuente. La investigación combinó análisis dinámico y ejecución simbólica para examinar 212 drivers firmados, reportando 38 fallos en 27 productos distintos. Los resultados evidencian que los drivers comerciales de Windows continúan presentando vulnerabilidades críticas a gran escala, incluso después de pasar procesos de validación y certificación. Además, POPKORN supera limitaciones propias del análisis tradicional al no depender de que el dispositivo físico esté conectado durante la evaluación.

Por otro lado, la investigación desarrollada por el grupo SoftSec-KAIST propone un enfoque de fuzzing consciente de tipos para el kernel de Windows, basado en inferencia estática de argumentos de llamadas al sistema y en la generación de entradas correctas y

malformadas. Este tipo de aproximación demuestra que técnicas aplicadas históricamente a nivel de usuario están siendo adaptadas al entorno del kernel, lo cual resulta especialmente relevante en el análisis de drivers Wi-Fi, ya que interactúan directamente con el sistema operativo en modo privilegiado.

Otra de las carencias encontradas en la literatura es la falta de trabajos que integren hardware inalámbrico real en un entorno Windows para probar drivers Wi-Fi. La mayoría de las investigaciones actuales se basan en modelos virtualizados, simuladores o instrumentación del kernel sin necesidad de tener presentes adaptadores físicos, limitando la visibilidad de comportamientos relacionados con la manipulación real de interrupciones, buffers de hardware y sincronización con el chipset. Esta restricción metodológica disminuye la posibilidad de encontrar errores que sólo se hacen evidentes en condiciones reales de operación, tales como tráfico intenso o manipulación de tramas de gestión (Balto et al., 2023).

Más recientemente, la empresa de investigación Vinopal (2024) publicó un informe titulado Breaking Boundaries: Investigating Vulnerable Drivers and Mitigating Risks, en el cual se analizaron controladores vulnerables de Windows basados en WDM y WDF. El estudio identificó que muchas de las fallas detectadas provenían de diseños defectuosos relativamente simples, como accesos de usuario sin restricciones, lo que permite que un actor con privilegios mínimos obtenga control sobre el kernel. Asimismo, el informe evidenció que los atacantes pueden cargar controladores vulnerables mediante el método conocido como Bring Your Own Vulnerable Driver (BYOVD), demostrando que un diseño inseguro del controlador puede abrir el sistema a explotación sin necesidad de vulnerabilidades complejas.

En la literatura de seguridad inalámbrica se ha demostrado la eficacia de ataques basados en tramas de gestión, como la desautenticación, el flooding de beacons o la manipulación de respuestas de sondeo, que degradan la disponibilidad del servicio sin necesidad de romper el cifrado de la red. Pero la mayoría de estos trabajos se centran en el efecto a nivel de usuario o de punto de acceso, sin llegar a estudiar en detalle el comportamiento interno del driver que procesa estas tramas. Esta falta de información impide saber cómo los controladores procesan largas series de eventos anormales y si estos pueden provocar errores de memoria, fuga de recursos o inconsistencias de estado en el kernel (Borrero y Ponce, 2023).

En el dominio específico de redes inalámbricas, aunque los drivers de Wi-Fi han recibido menor atención que otros componentes del kernel, también se han difundido estudios relevantes. Por ejemplo, la revisión de vulnerabilidades en dispositivos inalámbricos realizada por Kwan et al., (2024) analizó cientos de informes de CVE asociados a adaptadores Wi-Fi y controladores relacionados. Aunque el estudio no se enfocó exclusivamente en drivers de Windows, evidenció que la superficie de ataque inalámbrico continúa en expansión y que los controladores de hardware representan un vector crítico debido a su interacción directa con el sistema operativo en modo privilegiado.

Desde la perspectiva del fuzzing aplicado a software de Windows, Gupta et al., (2022) desarrollaron SoftFuzzer, una herramienta que instrumenta de manera estática binarios del sistema y aplica fuzzing sobre programas de espacio de usuario. Aunque no está dirigida específicamente a drivers, su enfoque demuestra que las técnicas de instrumentación estática pueden adaptarse a entornos cerrados como Windows, ampliando

el alcance de las metodologías tradicionales y evidenciando la posibilidad de trasladar estas técnicas al análisis de controladores del kernel.

Los estudios analizados muestran un progreso en el análisis de drivers de Windows y en el uso de fuzzing a nivel de kernel, pero también dejan ver la falta de metodologías integrales dirigidas a controladores Wi-Fi que integren tráfico real inalámbrico, comandos internos del sistema y análisis forense del kernel (Ramos, 2021). Esta ausencia de metodologías híbridas impide conocer por completo la superficie de ataque de los drivers inalámbricos en Windows y justifica la creación de metodologías experimentales que combinen diversos vectores de entrada, como se plantea en esta investigación.

Si bien varias investigaciones han descubierto vulnerabilidades de alto impacto a través de la revisión de interfaces IOCTL y syscalls, estos métodos suelen enfocarse en vectores de ataque desde el espacio de usuario, dejando sin cubrir las entradas que provienen directamente del aire (Bernal et al., 2025). Para los drivers Wi-Fi, gran parte del procesamiento se realiza antes de que el tráfico llegue a las capas superiores del sistema operativo, lo que significa que posibles vulnerabilidades pueden explotarse incluso en tramas de gestión o control sin necesidad de intervención de procesos de usuario (Moreno, 2022). Esta propiedad diferencia los drivers inalámbricos de otros tipos de controladores y refuerza la necesidad de usar métodos de prueba que combinen entradas de red y comandos internos del sistema.

Finalmente, un reporte técnico publicado por (Zeller et al., 2022) destacó que al menos 34 controladores de Windows fueron identificados como vulnerables a la toma de control del dispositivo por actores sin privilegios, entre los cuales se encontraban controladores basados en WDM y WDF. Esta evidencia reciente refuerza la necesidad de

auditorías periódicas y de análisis sistemáticos sobre drivers de red inalámbrica en Windows, especialmente en un escenario donde la superficie de ataque continúa aumentando debido a la expansión del ecosistema Wi-Fi.

Brechas identificadas

A partir de esta revisión del estado del arte, se identifican al menos tres vacíos que justifican el presente trabajo:

- Primero, aunque existen estudios sobre fuzzing de drivers de Windows en general principalmente los trabajos de Schumilo et al., (2020); Gupta et al., (2022); Singh y Kumar (2021) ninguno se enfoca específicamente en drivers de tarjetas de red Wi-Fi en Windows. Estas investigaciones abordan controladores del kernel de forma amplia, pero no consideran particularidades propias del entorno inalámbrico, como tramas 802.11, estados de enlace o comandos IOCTL/OID de NDIS. Estudios recientes han demostrado que los stacks de comunicación inalámbrica son una superficie de ataque difícil de defender debido a la gran cantidad de estados internos que deben mantener durante el escaneo, la autenticación, la asociación y la transferencia de datos. Cada uno de estos estados requiere la interpretación de varios campos estructurados dentro de las tramas 802.11 y la modificación de estructuras internas del controlador (Tariq et al., 2023). Estudios de implementaciones inalámbricas han demostrado que cambios triviales en los elementos de información (Information Elements) pueden corromper la máquina de estados del driver, causando degradación del servicio o fallos transitorios que no siempre resultan en vulnerabilidades explotables, pero que comprometen la estabilidad del servicio y pueden ser aprovechados para ataques de denegación de

servicio persistentes (Mora, 2024).

- Segundo, las metodologías reproducibles que combinan fuzzing de tramas 802.11, modificación de comandos IOCTL/OID y análisis forense basado en volcados de memoria no están documentadas de manera completa en la literatura reciente. Los estudios existentes tienden a presentar resultados experimentales parciales o centrados únicamente en la instrumentación del kernel, sin describir procedimientos replicables orientados a controladores Wi-Fi en Windows. Aunque la mayoría de los trabajos de fuzzing de redes inalámbricas se han llevado a cabo sobre implementaciones en Linux y sistemas embebidos, sus resultados son muy aplicables a Windows, ya que los conceptos subyacentes de procesamiento de tramas y gestión de estados son similares. Estudios de stacks WLAN han revelado que el fuzzing dirigido a tramas de control puede descubrir caminos de ejecución raramente ejercitados, como las etapas iniciales de conexión, autenticación y descubrimiento de red. Estos resultados confirman que las técnicas de fuzzing dirigidas a tramas 802.11 también son aplicables para evaluar drivers Wi-Fi en Windows, a pesar de que la arquitectura del sistema operativo y las capas de abstracción sean distintas (Chinchay et al., 2022).

- Tercero, la mayoría de los trabajos identificados (cuatro de los seis analizados) se enfocan en controladores comerciales o drivers de kernel sin utilizar hardware inalámbrico real dentro de entornos Windows. Esto evidencia una escasez de evaluaciones experimentales dirigidas a drivers Wi-Fi que reflejen condiciones operativas reales y permitan observar fallos reproducibles bajo escenarios controlados.

En función de estas brechas, el presente estudio se justifica al realizar una evaluación experimental de drivers Wi-Fi en sistemas Windows dentro de un entorno de laboratorio controlado, con el propósito de generar evidencia empírica y aportar lineamientos que fortalezcan la seguridad de estos controladores.

2.2 Marco Teórico

El desarrollo de sistemas informáticos modernos depende de la interacción eficiente entre el hardware y el software. Dicha comunicación se realiza mediante componentes especializados llamados controladores o drivers, encargados de traducir las instrucciones del sistema operativo a un lenguaje que el dispositivo físico pueda interpretar (Singh & Kumar, 2021). En los sistemas Windows, los drivers tienen un papel esencial. Permiten que los dispositivos de red, almacenamiento o gráficos funcionen de forma estable. Esto pasa dentro del ecosistema del kernel.

2.2.1 Concepto y función de los controladores en Windows

En los sistemas Windows, un controlador (driver) es un componente de software que funciona como intermediario entre el sistema operativo y el hardware, permitiendo que ambos se comuniquen de manera segura y estructurada. Dependiendo de su función, puede ejecutarse en modo usuario o en modo kernel; en este último caso posee privilegios elevados y acceso directo a la memoria del sistema, lo que incrementa su criticidad desde el punto de vista de seguridad (Microsoft, 2023). En el ámbito de redes, los controladores se fundamentan en la especificación NDIS, la cual define una arquitectura modular que estandariza la comunicación entre el hardware inalámbrico y los protocolos de red del sistema operativo. Esta estructura permite gestionar estados, comandos y flujos de datos de manera consistente, garantizando la operación correcta de los dispositivos de red.

Desde el punto de vista sistémico, los controladores en Windows son extensiones del kernel que encapsulan la complejidad del hardware y ofrecen una interfaz común al resto del sistema operativo. Esta tarea incluye no sólo la traducción de comandos, sino también el manejo de interrupciones, buffers de memoria, sincronización de eventos y control de estados del dispositivo. Por lo tanto, los drivers son parte integral del ciclo de vida de las operaciones de E/S y afectan directamente el rendimiento, la estabilidad y la seguridad del sistema.

Una propiedad esencial de los controladores en Windows es que se ejecutan en modo kernel, lo que les da acceso sin restricciones a la memoria y a las estructuras del sistema. Esto hace que los drivers sean código de alta criticidad para la seguridad: cualquier error lógico, condición de carrera o falta de comprobación de entradas puede causar errores que comprometan todo el sistema. A diferencia de las aplicaciones en modo usuario, los errores en drivers no sólo afectan al proceso que los invoca, sino que pueden corromper el kernel, colgar el sistema o permitir la ejecución de código privilegiado (Pastor, 2024).

En la arquitectura actual de Windows, los controladores se escriben utilizando marcos como WDM (Windows Driver Model) o WDF (Windows Driver Framework), que ofrecen bibliotecas, rutinas de controlador de eventos y abstracciones para simplificar el proceso. Estos frameworks intentan evitar errores comunes de gestión de memoria y sincronización, pero no eliminan la posibilidad de errores de implementación. De hecho, muchos incidentes de seguridad han revelado que incluso los drivers desarrollados en WDF pueden tener vulnerabilidades explotables cuando el código interno no maneja correctamente las entradas externas o los estados concurrentes (Microsoft Learn, 2023).

Para los dispositivos de red, el trabajo del driver es aún más complicado, ya que tiene que procesar flujos continuos de paquetes que llegan del exterior, administrar colas de transmisión y recepción e interactuar con el stack de red del sistema operativo. Esto significa que los controladores de red no solo ejecutan código de máquina, sino que también interpretan estructuras de datos complejas especificadas por estándares de comunicación (por ejemplo, IEEE 802.11 para redes inalámbricas) (Microsoft Learn, 2024).

Además, los drivers deben atender peticiones internas del sistema operativo a través de códigos de control IOCTL u OID para alterar las propiedades del dispositivo, consultar su estado o manipular recursos. Estas interfaces, si no se verifican correctamente, pueden permitir que procesos de usuario manipulen indirectamente estructuras internas del kernel, abriendo la puerta a la elevación de privilegios (IBM, 2025).

Es así que, el controlador no solo permite que el hardware funcione, sino que se convierte en una parte esencial de la cadena de confianza del sistema operativo. La seguridad de todo el sistema descansa en la seguridad de los drivers, en particular en los que manejan datos no confiables (como los drivers Wi-Fi). Esta razón hace que sea necesario probar de forma exhaustiva su comportamiento ante situaciones anormales y entradas erróneas.

2.2.2 Arquitectura NDIS y drivers Wi-Fi

La arquitectura NDIS establece un conjunto de funciones estándar que los controladores deben implementar para interactuar con el Network Stack de Windows. Esta estructura está compuesta por minipuertos, filtros y protocolos, los cuales colaboran para transmitir y procesar datos de manera organizada dentro del sistema operativo (Wang et al.,

2022). En los dispositivos Wi-Fi, el controlador de minipuerto se comunica directamente con el hardware mediante interrupciones y operaciones de bajo nivel, mientras que los controladores de filtro inspeccionan y manipulan los paquetes antes de entregarlos a capas superiores. Debido a esta organización en múltiples niveles, la superficie de ataque se amplía, ya que cada componente procesa entradas que pueden ser manipuladas por actores externos si no se validan correctamente. Por ello, la seguridad en drivers Wi-Fi depende tanto del diseño modular de NDIS como de la implementación adecuada de cada capa.

La NDIS (Network Driver Interface Specification) es el modelo arquitectónico que define cómo interactúan los controladores de red con el stack de comunicaciones de Windows. Este modelo define una arquitectura en capas que permite separar la implementación del hardware de las capas de protocolo, posibilitando la interoperabilidad y estandarización de los equipos de red. En redes inalámbricas, NDIS es el intermediario entre el controlador del adaptador Wi-Fi y las partes que se encargan de la autenticación, el cifrado y la administración de conexiones.

En esta arquitectura, los miniport drivers son la interfaz con el hardware inalámbrico. Estos controladores se encargan de recibir y enviar tramas, controlar el acceso al medio, manejar interrupciones y mantener buffers internos asociados a la tarjeta de red. Por encima de ellos trabajan los filtros NDIS y los controladores de protocolo, que examinan y modifican los paquetes antes de entregarlos a las capas superiores del sistema operativo. Esta división funcional da flexibilidad, pero también crea muchos puntos donde se pueden colar errores de validación o inconsistencias de estado (Microsoft Learn, 2025).

En los drivers Wi-Fi se añade complejidad por el propio protocolo IEEE 802.11, que implica muchos estados distintos (escaneo, autenticación, asociación, negociación de

parámetros de seguridad, mantenimiento de enlace, etc.). Cada cambio de estado requiere el procesamiento de tramas de control con estructuras internas precisas, lo que abre la posibilidad de errores ante secuencias inesperadas o campos modificados (Thankappan et al., 2022).

Además, el controlador interactúa con el firmware del chipset y el sistema operativo para sincronizar eventos asíncronos, temporizadores y respuestas de hardware. Estas acciones simultáneas pueden crear condiciones de carrera si no se toman medidas apropiadas de bloqueo y sincronización (Carrión, 2020). En situaciones de alta carga o tráfico malicioso, la acumulación de eventos puede saturar la máquina de estados del driver y causar degradación del rendimiento o comportamientos inesperados.

Más allá del modelo en capas, la arquitectura NDIS especifica ciertos caminos de procesamiento de paquetes entrantes y salientes, conocidos como receive path y transmit path. En el camino de recepción, las tramas capturadas por el hardware se pasan del firmware del chipset al miniport driver, el cual tiene que envolver los datos en estructuras NDIS y avisar al stack de red (usando interrupciones o sondeo). Este camino implica asignación dinámica de buffers, verificación de tamaños y actualización de contadores internos, haciendo a esta ruta susceptible a errores de desbordamiento de memoria, uso de punteros inválidos y errores de sincronización, en particular cuando se reciben ráfagas de tramas malformadas o inesperadas (CISCO, 2021).

En el path de transmisión, el flujo se invierte: las capas superiores del sistema producen paquetes que atraviesan filtros NDIS y llegan al miniport para convertirse en tramas físicas que se envían al hardware. En este tiempo, el driver manipula colas internas, temporizadores de retransmisión y acuses de recibo, lo que implica muchos cambios de

estado y accesos concurrentes a estructuras compartidas. Si estas estructuras no están protegidas por mecanismos de sincronización, se pueden producir condiciones de carrera entre las rutinas de transmisión, recepción e interrupción, lo que puede provocar corrupción de memoria en situaciones de estrés o tráfico manipulado (Redes Zona, 2025).

En términos de seguridad, la arquitectura NDIS también se integra con el modelo de objetos del kernel de Windows, lo que significa que muchas de las acciones del driver se realizan en el contexto de objetos compartidos por el subsistema de red, el administrador de energía y los servicios del sistema. Esto implica que un error de lógica en el controlador no sólo puede causar problemas de conectividad, sino que puede interferir con mecanismos a nivel de todo el sistema operativo, como la gestión de recursos o la gestión de dispositivos. Por eso, errores en el miniport driver se pueden filtrar hacia arriba y causar bloqueos del sistema, reinicios forzados del adaptador o pérdida total de conectividad, incluso cuando el hardware no tiene problemas (Carrión, 2020).

El tráfico inalámbrico externo no confiable combinado con los comandos de control internos de NDIS hacen que sea un sistema susceptible a ciertas órdenes de eventos anormales. Mientras que las tramas 802.11 cambian el estado del enlace y los buffers de recepción, las peticiones IOCTL/OID pueden cambiar al mismo tiempo las configuraciones, las políticas de energía o los parámetros de transmisión. Esta interacción en paralelo hace más compleja la máquina de estados del controlador y es por lo que el fuzzing híbrido (basado en tramas y en comandos de control) es tan eficaz para encontrar errores que no se encuentran cuando se prueba cada vector de entrada de forma aislada (CISCO, 2021).

La arquitectura NDIS también especifica interfaces estandarizadas para la configuración del dispositivo, lo que permite al sistema operativo averiguar las capacidades, establecer parámetros de administración de energía, cambiar los modos de funcionamiento y controlar las funciones avanzadas del adaptador. Estas interfaces, aunque necesarias para el funcionamiento del sistema, aumentan la superficie de ataque del controlador, ya que tienen que recibir información de procesos de usuario que podrían estar comprometidos (Microsoft Learn, 2025).

Pero tener varias capas de abstracción hace difícil rastrear errores, porque un error puede surgir de la interacción entre el firmware, el miniport driver o los filtros NDIS. Esto hace difícil la investigación forense y la determinación exacta de la causa de una falla, en particular si se trata de fallas esporádicas que sólo ocurren bajo ciertas circunstancias (Microsoft Learn, 2025).

La arquitectura NDIS, a pesar de buscar facilitar el desarrollo y la compatibilidad, crea un ambiente muy complejo donde pequeños errores de implementación se pueden multiplicar y causar grandes problemas de estabilidad y seguridad. Esto hace que los drivers Wi-Fi sean blancos comunes para pruebas de robustez con fuzzing (Microsoft Learn, 2025).

2.2.3 Comunicación IOCTL/OID y seguridad del kernel

En Windows, los controladores reciben solicitudes desde el espacio de usuario mediante llamadas IOCTL (Input/Output Control) u OID (Object Identifier), las cuales permiten configurar parámetros del dispositivo, consultar estados operativos o intercambiar datos específicos entre el hardware y el sistema operativo. Debido a que estas interfaces pueden modificar estructuras internas del driver, una validación incorrecta de los

parámetros recibidos puede generar vulnerabilidades de corrupción de memoria o elevación de privilegios en modo kernel (Dai & Chen, 2023). Para mitigar estos riesgos, Windows incorpora mecanismos como Driver Verifier y Kernel Address Space Layout Randomization (KASLR); sin embargo, estas medidas no eliminan los errores lógicos que ocurren durante la ejecución. Por ello, los análisis dinámicos mediante fuzzing resultan esenciales para identificar fallos que no son detectados en las pruebas convencionales, especialmente en controladores de red que procesan entradas externas y no confiables.

Los controladores Wi-Fi en sistemas Windows operan bajo la arquitectura Network Driver Interface Specification (NDIS), un modelo que organiza la comunicación entre el hardware de red y el sistema operativo mediante capas modulares. En este modelo, los miniport drivers interactúan directamente con el chipset inalámbrico, mientras que los protocol drivers administran funciones superiores como la autenticación, la gestión de estados y la entrega de paquetes. Esta separación de responsabilidades permite una mayor flexibilidad, pero también introduce puntos críticos donde pueden generarse errores si la validación de parámetros no se realiza correctamente (Kampourakis et al., 2022).

- Entre las funciones esenciales del driver se incluyen:
- el filtrado y validación de tramas 802.11,
- la interpretación de elementos informativos,
- la sincronización con puntos de acceso,
- la gestión de buffers internos y colas de transmisión,
- la ejecución de comandos IOCTL/OID para el sistema operativo.

La complejidad de estas tareas explica por qué ciertos eventos anómalos pueden degradar temporalmente la disponibilidad del servicio Wi-Fi. La literatura ha documentado que una proporción importante de vulnerabilidades en Windows se originan en fallos de validación dentro de los controladores, especialmente en componentes relacionados con memoria y estructuras internas (Zhao et al., 2022). Esto justifica el análisis realizado en esta investigación, donde se evalúa la resistencia del driver ante tramas malformadas y escenarios atípicos.

Las IOCTL (Input/Output Control) y las OID (Object Identifier) son las formas primarias en que el sistema operativo y las aplicaciones en modo usuario se comunican con los controladores de dispositivo en Windows. Estas interfaces pueden exponer comandos para manipular las configuraciones o consultar el estado interno del hardware, siendo puntos de entrada directamente al código del controlador que se ejecuta en modo kernel (Carrión, 2020).

Desde el punto de vista de la seguridad, estas interfaces son un vector importante porque abren la puerta a que procesos menos privilegiados manipulen en la práctica la ejecución de código privilegiado. Si los parámetros de entrada no se validan correctamente, puede dar lugar a accesos fuera de rango, corrupción de estructuras internas o manipulación no autorizada del estado del controlador. Para drivers de red, estas interfaces pueden influir en la manipulación de buffers, colas de transmisión y estructuras de control de enlace (Microsoft Learn, 2024).

Las llamadas IOCTL suelen pasar estructuras complejas que contienen punteros, tamaños de buffer y campos de control. Una falla en la validación de estos valores puede llevar a que el driver acceda a memoria no válida o interprete datos maliciosos como

legítimos. Además, los OID pueden leer o escribir parámetros concretos del adaptador (modos de funcionamiento, capacidades de cifrado, canal, etc.), alterando la lógica interna del controlador en caso de recibir secuencias inesperadas de comandos (Microsoft Learn, 2024).

Si bien Windows incluye protecciones como Driver Verifier, KASLR o control de integridad de código, estas medidas están orientadas a encontrar errores en tiempo de desarrollo o dificultar la explotación de errores, pero no evitan errores lógicos en el código del controlador. Por lo cual, la carga de verificar adecuadamente las entradas recae principalmente en el desarrollador del controlador (Carrión, 2020).

En drivers Wi-Fi, la mezcla de comandos IOCTL/OID con tráfico inalámbrico externo crea un ambiente especialmente complejo, ya que el estado interno del controlador puede cambiar tanto por eventos de hardware como por peticiones del sistema operativo. Esta interacción crea la posibilidad de inconsistencias de estado, especialmente si hay cambios rápidos en la conectividad o si se realizan operaciones de configuración mientras el dispositivo está procesando tramas de red (Carrión, 2020).

Además, ciertos ataques sofisticados combinan el aprovechamiento de drivers vulnerables con técnicas como Bring Your Own Vulnerable Driver (BYOVD), en la que un atacante carga deliberadamente un controlador legítimo pero vulnerable para realizar acciones privilegiadas. Aquí es donde las interfaces IOCTL entran en juego como la forma de explotar el driver y hacer cosas ilegales en el kernel (Kaspersky, 2024).

Es por ello que el análisis de seguridad de controladores no debe centrarse solamente en el tráfico de red que procesan, sino también en las interfaces de control internas que permiten alterar su funcionamiento. La evaluación combinada de tramas

inalámbricas y comandos IOCTL/OID ofrece una mejor perspectiva de la superficie de ataque real de los drivers Wi-Fi en Windows (Kaspersky, 2024)..

El sistema operativo representa las llamadas IOCTL como paquetes IRP (I/O Request Packets), que viajan a través de varias capas del subsistema de E/S antes de llegar al controlador de destino. Cada IRP tiene punteros a buffers de entrada y salida, longitudes de datos y códigos de control que especifican la operación requerida. Si el controlador no comprueba adecuadamente estos parámetros, puede terminar accediendo a direcciones no válidas, interpretando tamaños incorrectos o usando estructuras no inicializadas, lo que abre la puerta a corrupción de memoria en modo kernel (Carrión, 2020).

Un punto importante es la forma de transferencia de datos que usa la solicitud IOCTL: buffered I/O, direct I/O o neither I/O; en buffered, el sistema crea un buffer intermedio, en direct I/O se usan descriptores de memoria mapeados directamente al espacio del kernel (Bianchi et al., 2020). En el modo neither, el driver manipula directamente punteros que le entrega el proceso de usuario, con lo que debe hacer fuertes verificaciones para evitar accesos arbitrarios a memoria. En los drivers de red, la mezcla de diferentes formas de transferencia en una misma implementación aumenta la complejidad del código y abre la puerta a errores de certificación.

Para las interfaces OID en NDIS, las solicitudes de configuración y consulta también se convierten internamente en estructuras que contienen identificadores de objeto y búferes de datos adjuntos. Estas peticiones no sólo cambian parámetros de configuración, sino que pueden alterar estructuras internas de estado de enlace, políticas de seguridad o de gestión de energía. Si se reciben secuencias inusuales de OID entremezcladas con tráfico

de red simultáneo, se pueden crear inconsistencias en la lógica del controlador que no se consideran en los flujos operativos normales (Kaspersky, 2024).

Otro punto importante es que la mayoría de las peticiones IOCTL y OID se gestionan en contextos de ejecución diferentes, como rutinas de interrupción, hilos diferidos y contexto de usuario. Esta variedad de contextos aumenta la posibilidad de condiciones de carrera al intentar acceder a recursos compartidos sin mecanismos de sincronización. En drivers Wi-Fi, por ejemplo, el estado del dispositivo puede cambiar en cualquier momento en respuesta a eventos de red, y esta simultaneidad es una fuente común de errores de sincronización que conducen a corrupción de estructuras internas (Micrsoft Learn, 2022).

Desde el punto de vista de la ingeniería inversa, el estudio de estas interfaces es complicado porque los códigos IOCTL y OID privados no suelen ser documentados por los fabricantes. Esto requiere el uso de métodos de análisis dinámico y monitoreo de llamadas para descubrir qué comandos existen y cómo reacciona el driver ante ciertos parámetros. Esta opacidad hace que sea más probable que existan caminos internos de código raramente ejecutados que no fueron probados en el momento en que se desarrolló el controlador (Micrsoft Learn, 2022).

En ataques sofisticados, las IOCTL se pueden aprovechar como mecanismos de persistencia o escalada de privilegios, en particular cuando se combinan con la carga de drivers vulnerables usando técnicas BYOVD (Bring Your Own Vulnerable Driver). En estos casos, el atacante usa funcionalidades legítimas del controlador para realizar acciones privilegiadas que no deberían estar permitidas a procesos de bajo nivel, aprovechando

vulnerabilidades en la verificación de comandos y estructuras de control (Micrsoft Learn, 2022).

2.2.4 Tipos de Comunicación IOCTL/OID

- **Consultas de estado del adaptador (Query OID / IOCTL de consulta)**

Le permiten al sistema operativo averiguar el estado interno del adaptador (nivel de señal, modo, estado de conexión, estadísticas de transmisión, etc.). Estas peticiones requieren que el driver manipule estructuras internas y devuelva información al espacio de usuario. Desde el punto de vista de seguridad, una manipulación incorrecta de buffers de salida o tamaños de estructura puede dar lugar a lecturas fuera de límites o revelación de información confidencial del kernel (Microsoft Learn, 2025).

- **Configuración de parámetros de red (Set OID / IOCTL de configuración)**

Este canal de comunicación puede usarse para cambiar las propiedades del adaptador, como canal, potencia de transmisión, tipo de cifrado, modo o roaming. Estas acciones impactan la máquina de estados del controlador y, si no se verifican adecuadamente, pueden generar inconsistencias internas, corrupción de memoria o alteraciones no autorizadas en el funcionamiento del dispositivo. En fuzzing, estas interfaces son de interés para generar estados inesperados en el driver (Microsoft Learn, 2025).

- **Gestión de autenticación y asociación (OID de control de enlace)**

Contienen comandos para iniciar, cancelar o reiniciar los procesos de autenticación y asociación con puntos de acceso. Estas peticiones compiten con las recepciones simultáneas de tramas 802.11, creando condiciones de carrera si se mezclan con tráfico malformado. Esta comunicación es fundamental para analizar fallos de sincronización entre eventos internos del sistema y eventos externos de red (Microsoft Learn, 2025).

- **Control de energía y estados de suspensión (Power Management OIDs)**

Posibilitan que el sistema ingrese en modos de bajo consumo, suspensión selectiva o reinicio de hardware. Estos cambios de estado implican que el driver libere y vuelva a asignar recursos internos, por lo que son puntos críticos para errores use-after-free o null-dereference. En el fuzzing, la manipulación iterativa de estos comandos puede exponer fallos en el manejo del ciclo de vida del dispositivo (Microsoft Learn, 2025).

- **Operaciones de reinicialización del adaptador (Reset y Reconfigure IOCTLs)**

Algunos IOCTL pueden forzar reinicios parciales o completos del adaptador Wi-Fi sin reiniciar el sistema operativo. Estas operaciones interrumpen las transferencias en curso y fuerzan al controlador a reconfigurar buffers, colas y estados internos. Si estos procesos no están bien sincronizados, se pueden producir errores de concurrencia, corrupción de estructuras internas o inconsistencia en la máquina de estados (Microsoft Learn, 2025).

- **Interacción con firmware del chipset (Vendor-specific OIDs)**

Muchos fabricantes definen OIDs privados para hablar directamente con el firmware del adaptador, para darle comandos de bajo nivel o para averiguar datos específicos de hardware. Estas interfaces a menudo están menos documentadas y carecen de validaciones rigurosas, lo que las convierte en objetivos privilegiados para fuzzing dirigido e ingeniería inversa (Microsoft Learn, 2025).

- **Control de colas y buffers de transmisión (Traffic Management OIDs)**

Posibilitan alterar parámetros de priorización, tamaño de colas y políticas de retransmisión. Una comprobación insuficiente de estos parámetros puede causar desbordamiento de colas, uso excesivo de memoria o bloqueo de hilos, lo que resultará en indisponibilidad del servicio. Esta comunicación es fundamental para analizar ataques de degradación operativa y denegación de servicio a nivel de driver (Microsoft Learn, 2025).

- **Notificaciones de eventos asíncronos (Indications NDIS hacia el sistema)**

Aunque no son IOCTLs enviadas por el usuario, estas notificaciones son parte de la comunicación bidireccional entre el driver y el sistema operativo. La pérdida de enlace, el cambio de canal o la detección de interferencia inician actualizaciones internas que pueden competir con las IOCTL simultáneas. Esta mezcla hace más complejo el análisis y es un punto sensible para las condiciones de carrera detectables con fuzzing (Microsoft Learn, 2025).

2.2.5 Tipos de vulnerabilidades en controladores

Las vulnerabilidades en controladores pueden clasificarse según su origen y el efecto que producen en el sistema, especialmente cuando involucran operaciones en memoria o acceso a recursos privilegiados. Diversos estudios han demostrado que los drivers de Windows presentan fallos recurrentes asociados a errores lógicos y de validación insuficiente, lo que los convierte en un vector crítico para la seguridad del sistema operativo (Zhao et al., 2022).

Entre las vulnerabilidades más frecuentes en drivers de red se encuentran:

- Desbordamiento de búfer (buffer overflow), que ocurre cuando se escriben datos más allá del espacio asignado en memoria, pudiendo sobrescribir información crítica del sistema (Ardi & Cadar, 2024)
- Uso después de liberar memoria (use-after-free), en el cual el controlador accede a una dirección previamente liberada, generando comportamientos impredecibles o corrupción de datos.
- Condiciones de carrera (race conditions), derivadas de accesos concurrentes a recursos compartidos sin mecanismos adecuados de sincronización.

- Referencias nulas o punteros colgantes (null pointer dereference), que producen fallos de ejecución al intentar acceder a direcciones inválidas.

De acuerdo con el análisis realizado por (Zhao et al., 2022), aproximadamente el 40 % de las vulnerabilidades críticas asociadas a drivers de Windows reportadas en la base CVE están vinculadas a fallos de manejo de memoria. Este resultado confirma que la falta de validación de entradas continúa siendo el principal desencadenante de errores explotables en controladores que operan en modo kernel, reforzando la necesidad de implementar mecanismos de verificación más estrictos durante su desarrollo y pruebas.

Las vulnerabilidades en controladores de dispositivos a menudo surgen de errores de programación relacionados con la gestión de memoria, la sincronización de procesos y la validación de entradas. Como los drivers se ejecutan en modo kernel, estos bugs pueden llevar a consecuencias graves, como ejecución arbitraria de código, escalada de privilegios y cuelgues de todo el sistema (Microsoft, 2024).

Uno de los tipos más comunes de vulnerabilidades son los desbordamientos de búfer, que se producen cuando el controlador copia datos sin verificar si el destino es lo suficientemente grande. En el caso de los drivers de red, estos desbordamientos se pueden explotar con paquetes maliciosos que manipulen los campos de longitud para sobrescribir estructuras del kernel. Si bien la mayoría de los controladores actuales realizan comprobaciones elementales, la complejidad de las estructuras 802.11 abre la puerta a errores sutiles en el manejo de campos opcionales (Microsoft, 2024).

Otra clase importante es la use-after-free, que ocurre cuando el driver sigue manipulando punteros a estructuras ya liberadas. Este tipo de vulnerabilidad generalmente se da en condiciones de carrera, en las cuales varios hilos de ejecución están accediendo a

los mismos recursos sin mecanismos de sincronización. En los drivers Wi-Fi, estas situaciones pueden ocurrir en reconexiones rápidas, cambios de canal o reinicializaciones de hardware (Telconet, 2025).

Las condiciones de carrera son un problema especialmente difícil en controladores que manejan interrupciones asíncronas del sistema operativo o del hardware. Si las rutinas de interrupción, los temporizadores y las llamadas de control no están sincronizadas, se pueden generar estados inconsistentes que conduzcan a corrupción de datos o errores esporádicos difíciles de reproducir (Telconet, 2025).

También hay vulnerabilidades de referencia nula o punteros no inicializados que pueden dar lugar a accesos inválidos a memoria y fallos del sistema. Estos errores generalmente resultan en bloqueos, pero en ciertos casos pueden ser aprovechados para manipular el flujo de ejecución del kernel (Bianchi et al., 2020).

Más allá de los fallos de memoria, hay vulnerabilidades lógicas que abren la puerta a abusos de funcionalidades legítimas del controlador, como establecer parámetros de seguridad inseguros, activar modos inseguros o saltarse controles de acceso internos. Estas vulnerabilidades no siempre causan errores evidentes, pero pueden permitir ataques persistentes o debilitar la seguridad del sistema.

En el caso de drivers Wi-Fi, las vulnerabilidades pueden involucrar tanto el procesamiento de tramas como la gestión de estados de conexión, autenticación y cifrado. La composición de varias fuentes de entrada hace más probable que los errores de validación sólo se manifiesten en ciertas secuencias de eventos, lo que requiere técnicas de prueba exhaustivas para descubrirlos (Mora, 2024).

2.2.6 Principios y tipos de fuzzing

El fuzzing es una técnica de prueba automatizada utilizada para enviar grandes cantidades de entradas aleatorias, malformadas o parcialmente dirigidas a un programa con el fin de observar su comportamiento y detectar fallos que no emergen durante las pruebas convencionales (Zeller et al., 2022). Esta técnica de prueba que implica la generación automatizada de entradas inesperadas para provocar comportamientos inesperados en el software que se está probando. A diferencia de las pruebas funcionales convencionales, el fuzzing no verifica solo los casos de uso anticipados, sino que explora caminos inesperados que pueden exponer errores ocultos en la lógica de procesamiento (TI Rescue, 2024).

En su forma más simple, el fuzzing implica tomar entradas válidas y mutarlas aleatoriamente, cambiando campos, longitudes y datos. Este método, llamado black-box fuzzing, no necesita información interna del programa, pero es ineficiente cuando el espacio de estados es grande (por ejemplo, controladores de red con muchas transiciones internas) (TI Rescue, 2024).

Esta metodología permite evaluar cómo los sistemas reaccionan frente a entradas inesperadas y resulta especialmente útil en componentes de bajo nivel, como controladores de dispositivos, debido a su interacción directa con el sistema operativo en modo privilegiado. (Khan & Allah, 2022) Existen tres enfoques principales de fuzzing, los cuales se presentan en la Tabla 1.

Tabla 1. *Tipos de técnicas de fuzzing y ejemplos.*

Tipo de fuzzing	Características principales	Ejemplo de herramienta
Black-box	No requiere conocimiento interno del programa. Se basa en mutaciones de entradas válidas.	Boofuzz, Peach Fuzzer
Grey-box	Usa retroalimentación parcial, como cobertura de código, para dirigir las pruebas.	AFL, WinAFL
White-box	Emplea análisis estático, simbólico o de flujo para generar casos precisos.	SAGE, KLEE

Nota. Elaboración propia con base en (Zeller et al., 2022).

Los enfoques grey-box son los más utilizados por equilibrar rendimiento y profundidad de análisis. En entornos Windows, WinAFL se destaca por extender AFL al ecosistema de Microsoft e integrar la API de instrumentación DynamoRIO, lo que permite realizar campañas de fuzzing guiado en binarios de Windows sin acceso al código fuente (Zeller et al., 2022).

El grey-box fuzzing agrega retroalimentación parcial, generalmente en forma de cobertura de código, para guiar la generación de entradas hacia caminos de ejecución no descubiertos. Esta técnica es especialmente efectiva para analizar software

complejo, ya que aumenta la probabilidad de encontrar errores difíciles en la lógica interna del programa. Pero su uso en drivers de Windows es complicado por las restricciones de instrumentación en modo kernel (TI Rescue, 2024)..

El fuzzing de caja blanca usa análisis simbólico y exploración sistemática de caminos para crear entradas muy afinadas que desencadenen errores profundos. Si bien este método es más exacto, su coste computacional impide su uso en tiempo real y es inviable para campañas largas sobre controladores reales (TI Rescue, 2024).

En el caso de los drivers Wi-Fi, el fuzzing se puede dirigir tanto a las interfaces internas de control como al tráfico inalámbrico externo. El fuzzing de tramas 802.11 manipula campos en los encabezados y elementos de información, y el fuzzing IOCTL/OID manipula estructuras de control enviadas desde el sistema operativo. La combinación de los dos métodos amplía notablemente el alcance del análisis (Lázaro, 2018).

Un punto importante del fuzzing en drivers es el requerimiento de monitoreo fuera de banda, ya que una falla puede tumbar todo el sistema. Por eso se utilizan entornos virtualizados, reinicios automáticos y análisis de volcados de memoria para encontrar la causa raíz de los errores. Sin ellos, la identificación y clasificación de vulnerabilidades es imposible (Lázaro, 2018).

2.2.6 Fuzzing aplicado a drivers de Windows

El fuzzing de drivers requiere técnicas de instrumentación específicas debido a que estos componentes se ejecutan en modo kernel y un fallo puede comprometer la estabilidad completa del sistema operativo. Para superar esta limitación, la literatura reciente describe el uso de entornos virtualizados y monitoreo fuera de banda, lo que permite observar fallos

sin causar interrupciones en el sistema anfitrión (Singh & Kumar, 2021). En este contexto, herramientas como NTFuzz y POPKORN han demostrado la eficacia del fuzzing guiado en entornos de kernel, permitiendo identificar vulnerabilidades de alto impacto en controladores firmados de Windows (Gupta et al., 2022).

El fuzzing en drivers de Windows es mucho más complicado que el fuzzing en aplicaciones de modo usuario, ya que los drivers se ejecutan en el espacio de direcciones del kernel y cualquier error puede corromper todo el sistema. Esta característica requiere de estrategias de prueba con mecanismos de recuperación automática, monitoreo fuera de banda y análisis posterior a volcados de memoria para determinar la causa raíz de las fallas sin interrumpir permanentemente el proceso experimental (Lázaro, 2018).

Uno de los mayores desafíos consiste en instrumentar el código del controlador. Mientras que las aplicaciones de usuario pueden instrumentarse fácilmente con mecanismos de seguimiento de cobertura, los drivers necesitan enfoques de instrumentación específicos, como la instrumentación en hipervisor, la emulación parcial del kernel o el uso de marcos de trabajo de depuración. Estas restricciones han llevado a la creación de instrumentos específicos para enviar datos maliciosos a controladores sin código fuente, dirigiéndose a superficies expuestas como IOCTL, OID y puntos de entrada de dispositivos (Microsoft, 2023).

Las técnicas más comunes para fuzzing drivers en Windows abarcan fuzzing de interfaces y fuzzing de red. En el primer caso, el fuzzer crea estructuras de control que envía al controlador a través de llamadas al sistema, probando el comportamiento del driver ante parámetros inesperados. Este enfoque es particularmente útil para identificar vulnerabilidades de elevación de privilegios y corrupción de memoria en fallos de

validación de estructuras internas. En el segundo caso, el fuzzing se realiza sobre los datos que el hardware proporciona al controlador (como en los drivers de red, donde las tramas recibidas del exterior se manipulan para provocar errores en el procesamiento de paquetes) (Lázaro, 2018).

Los entornos virtualizados son esenciales para realizar fuzzing sobre drivers, ya que permiten restaurar el sistema a un estado limpio después de una falla y automatizar ciclos repetitivos de prueba. Además, la virtualización permite tomar volcados de memoria y registros de depuración, información crucial para el análisis forense posterior. Pero para drivers de red inalámbrica, la virtualización por sí sola no siempre es suficiente para simular con exactitud el hardware, por lo que se necesitan adaptadores físicos reales en el laboratorio.

Otro punto importante es la dificultad para distinguir entre fallos críticos y degradaciones de funcionamiento. No todo comportamiento anormal descubierto en una campaña de fuzzing son vulnerabilidades explotables; algunas son pérdidas momentáneas de rendimiento, acumulación de estados o reinicios internos del driver. Sin embargo, estos impactos pueden ser igualmente significativos desde la perspectiva de disponibilidad del servicio, en particular en lugares donde la conectividad inalámbrica es esencial. Por eso, el fuzzing en drivers se ha de guiar por métricas de estabilidad, recuperación y consistencia del servicio, más allá de la detección de errores de memoria (SysAdminok, 2023).

Además, el fuzzing de drivers se enfrenta a la barrera de la reproducción de ciertos estados internos del controlador, que pueden depender de secuencias muy precisas de eventos o interacciones entre el firmware del hardware y el sistema operativo. Esta propiedad implica que ciertos errores sólo se manifiesten después de muchas ejecuciones o

para patrones de entrada específicos, lo que refuerza la necesidad de campañas de prueba largas y ricas (Lázaro, 2018).

No obstante, estos estudios se centran principalmente en controladores genéricos y no en módulos asociados a interfaces inalámbricas. La aplicación del fuzzing en drivers Wi-Fi continúa siendo un campo poco explorado, a pesar de la cantidad creciente de vulnerabilidades reportadas en dispositivos inalámbricos. Investigaciones como la realizada por (Kwan et al., 2024) confirman que el tráfico 802.11 representa un vector crítico debido a la interacción directa entre hardware y sistema operativo, reforzando la necesidad de evaluar de manera experimental los drivers Wi-Fi en Windows. (Koutroumpouchos et al., Threats and Countermeasures in IEEE 802.11 Wireless Networks: A Systematic Review., 2023)

2.2.7 Análisis forense y mitigación

Cuando una campaña de fuzzing detecta un fallo, se procede al análisis forense del controlador mediante herramientas de depuración en modo kernel, como WinDbg o KD, las cuales permiten examinar volcados de memoria y rastrear el flujo de ejecución para identificar el origen del error (Vinopal, 2024). La interpretación correcta de estos registros facilita determinar si el fallo es reproducible, si afecta estructuras críticas del sistema y si tiene potencial de ser explotado en un contexto adverso. Este proceso resulta fundamental para diferenciar fallos benignos de vulnerabilidades que requieren acciones de mitigación.

Las estrategias orientadas a reducir el riesgo incluyen validar las entradas de forma rigurosa, fortalecer los mecanismos de control interno y utilizar controladores firmados digitalmente para impedir la carga de módulos no confiables. Asimismo, se recomienda mantener el software actualizado, adoptar marcos seguros para el desarrollo de drivers

como WDF, e implementar principios de programación defensiva para disminuir la probabilidad de errores lógicos. Estas medidas, combinadas con auditorías periódicas y pruebas dinámicas, contribuyen a mejorar la resiliencia de los controladores Wi-Fi en sistemas Windows.

El análisis forense de fallos en drivers Wi-Fi es una etapa esencial tras descubrir comportamientos anormales en campañas de fuzzing. Este proceso intenta encontrar la causa del fallo, si es reproducible y si es explotable. Como los controladores se ejecutan en modo kernel, el análisis necesita herramientas específicas para analizar volcados de memoria, registros de interrupciones y estados internos del sistema operativo (Yaacoub et al., 2025).

Herramientas como WinDbg pueden examinar la pila de llamadas, localizar las direcciones de memoria en cuestión y explorar las estructuras internas del kernel. A través de símbolos de depuración se puede saber qué partes del driver estaban ejecutándose en el momento del error, descubriendo errores de validación, uso incorrecto de punteros o condiciones de carrera (Microsoft Learn, 2025).

La sincronización de las grabaciones de tráfico con los registros del sistema es esencial para reconstruir la serie de eventos que condujeron al fallo. Esta correlación es capaz de diferenciar entre errores inducidos por tráfico externo, comandos internos o errores de hardware, dándote una imagen completa de cómo se comporta el controlador en situaciones adversas (Yaacoub et al., 2025)..

En cuanto a la mitigación, las medidas se centran en reforzar la validación de entradas, los controles de límites de memoria y los mecanismos de sincronización interna. El uso de frameworks actuales como WDF ayuda a evitar errores estructurales, pero no

elimina la necesidad de una buena lógica defensiva en el procesamiento de tramas y comandos.

En términos prácticos, el hecho de tener los controladores actualizados y aplicar los parches de seguridad en tiempo y forma disminuye la superficie de ataque a vulnerabilidades conocidas. Además, las políticas de integridad de controladores y listas negras de drivers vulnerables restringen el uso de módulos inseguros en el sistema.

La integración de pruebas regulares de fuzzing en el proceso de desarrollo de drivers puede revelar errores antes de que lleguen a muchas máquinas. Esta actitud proactiva es especialmente importante en dispositivos de red, en los que el entorno externo es incontrolable y la continuidad de servicio es fundamental (Yaacoub et al., 2025).

2.2.8 Debilidades Comunes Documentadas en Drivers Wi-Fi

Las vulnerabilidades más comunes en drivers Wi-Fi tienen que ver con la complejidad del protocolo IEEE 802.11 y la necesidad de manejar múltiples estados de conexión. A diferencia de los protocolos cableados, el mundo inalámbrico está abierto a interferencias, pérdida de paquetes y eventos asíncronos, y los controladores deben lidiar con escenarios muy variables (Ahmadpour, 2020).

Una vulnerabilidad común es en el manejo de tramas de control, especialmente beacons, probe responses y tramas de autenticación. Dichas estructuras tienen varios campos opcionales de longitud variable que pueden provocar errores en la validación de offsets y tamaños de buffer. Si bien los controladores actuales descartan muchas tramas erróneas, largas ráfagas de tráfico anormal pueden corromper el estado interno del controlador (Bianchi et al., 2020).

Otra vulnerabilidad informada es en la máquina de estados del enlace. Las reconexiones rápidas, los cambios de canal y los fallos de autenticación repetidos pueden dejar estructuras internas en un estado inconsistente que no siempre se libera correctamente. Esto puede dar lugar a la acumulación de recursos, aumento gradual del uso de CPU o latencias altas de procesamiento de paquetes (Bianchi et al., 2020).

También se han encontrado problemas de sincronización entre el firmware del chipset y el controlador del sistema operativo. Errores de interpretación o lentitud del hardware pueden causar bloqueos temporales, reinicios forzados del adaptador o pérdida de conectividad difícil de diagnosticar.

En situaciones de congestión, los drivers pueden favorecer el tratamiento de ciertas tramas en detrimento de otras, perjudicando la experiencia del usuario y pudiendo ser utilizado para degradar el servicio sin violar la seguridad criptográfica de la red. Esta vulnerabilidad es especialmente pertinente en ataques de denegación de servicio basados en tráfico legítimo.

Las variaciones de implementación entre fabricantes dan lugar a controladores de diferente robustez. Mientras que unos priorizan la estabilidad ante situaciones adversas, otros maximizan el rendimiento en condiciones regulares, haciéndose más susceptibles ante tráfico manipulado. Esta diversidad hace difícilmente generalizables los resultados y refuerza la necesidad de una evaluación específica por proveedor.

Diversos estudios coinciden en que las debilidades más comunes en los controladores Wi-Fi están relacionadas con:

- validación inadecuada de campos 802.11,

- fallos en reconstrucción de estados internos,
- problemas de manejo de memoria,
- saturación del canal bajo ataques de gestión,
- respuestas inconsistentes a tramas malformadas (Koutroumpouchos et al., 2022).

Los análisis empíricos de comportamiento de drivers indican que muchos controladores presentan sensibilidad operativa cuando reciben secuencias prolongadas de tramas atípicas, incluso cuando no generan errores críticos. Esto se ha observado en estudios recientes orientados a evaluar la estabilidad de drivers bajo condiciones de stress testing, donde adaptadores modernos muestran diferentes niveles de tolerancia frente a escenarios de manipulación de tramas (Tripathy et al., 2024).

Este tipo de debilidades no necesariamente representan vulnerabilidades, pero sí afectan la disponibilidad del servicio, lo cual es crítico en entornos donde la estabilidad de la conexión es prioritaria.

2.3 Comparativa de comportamiento por vendor y por tipo de prueba

El análisis comparativo entre los diferentes vendors y las pruebas ejecutadas permite contextualizar los resultados del estudio dentro de un marco más amplio de comportamiento de controladores Wi-Fi en entornos sometidos a estrés, tramas malformadas y eventos no convencionales. Este apartado complementa el estado del arte presentando evidencia empírica previa generada durante las iteraciones iniciales del laboratorio que ayuda a comprender cómo se comportan distintos controladores, qué patrones se repiten y qué diferencias existen entre vendors como TP-Link y Broadcom.

Mientras el capítulo 4 se centra en los resultados obtenidos sobre el driver objetivo evaluado en esta investigación, esta sección presenta una visión transversal que permite comparar comportamientos, niveles de estabilidad, reacciones a pruebas específicas y sensibilidad frente a cargas anómalas. De esta forma, este subcapítulo cumple dos funciones clave:

- fortalecer la validez del diseño experimental,
- establecer una base comparativa sólida para interpretar los resultados finales del estudio.

2.3.1 Bitácora general de pruebas realizadas

La bitácora de pruebas es una herramienta metodológica para asegurar la trazabilidad, reproducibilidad y validez de los resultados experimentales. En las pruebas de seguridad sobre controladores de red, cuyos comportamientos dependen de muchas variables, el registro de cada ejecución es esencial para reconocer patrones, correlacionar sucesos y descartar ejecuciones anómalas (Salazar, 2025).

En la bitácora se registra el orden en que se realizaron las pruebas, así como las condiciones en que se realizaron (tipo de adaptador, versión de driver, sistema operativo, tipo de prueba, notas técnicas, etc.). Estos datos reconstruyen el estado preciso en que se encontraba el controlador en el momento en que ocurrió un comportamiento específico, lo que permite analizarlo posteriormente y compararlo entre ejecuciones (Barrios et al., 2022).

La bitácora corresponde al registro cronológico de todas las pruebas ejecutadas a lo largo del proceso de evaluación. Incluye fecha, hora, SSID, adaptador utilizado, driver, resultado y notas técnicas. Esta información evidencia que el proceso experimental fue

consistente, repetible y operó bajo un mismo entorno controlado denominado TESTING, garantizando la estabilidad de las condiciones base.

Los registros muestran ejecuciones repetidas de pruebas como BAS-01, DEA-01, BEA-01, CIC-01, AUT-01 y PRO-01, abarcando tantos escenarios de desautenticación, fuzzing de beacons, reconexión intensiva, autenticación forzada y manipulación de probe-responses. Asimismo, la bitácora refleja que se realizaron validaciones específicas contra dos vendors distintos: TP-Link Technologies y Broadcom, ambos ampliamente utilizados en sistemas Windows.

Este contexto ayuda a entender que los resultados obtenidos no responden a eventos aislados, sino a un conjunto diversificado de ejecuciones diseñadas para evaluar la robustez de distintos controladores ante un espectro amplio de perturbaciones típicas en redes 802.11. depuran máquinas de estados complejas como las que se encuentran en los drivers Wi-Fi.

Figura 1. *Bitácora general de pruebas realizadas*

Fecha	Hora	Prueba	SSID	BSSID	Canal	Adaptador	Driver	Resultado	Notas
2025-11-04	17:52:58	BAS-00	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	OK	Baseline 5-10 min navegación normal | logs: C:\LAE
2025-11-05	00:30:56	DEA-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	OK	Deauth 10 frames; reconexión ~8 s | logs: C:\LAB_W
2025-11-05	02:23:47	BEA-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	OK	Fuzzing 200 beacons @50pps; SSIDs FUZZ_* visibles
2025-11-18	02:59:53	CIC-01				Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	OK	30 ciclos con 8s; SSID TESTING ch6; reconexión ~3–6
2025-11-18	14:39:51	AUT-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	INFO	mdk4 auth flood ~25s; canal 6; BSSID 50:C7:BF:1E:9F
2025-11-18	16:12:46	PRO-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	TP-Link Technologies Co., Ltd. 1030.41.514.2020	INFO	ProbeResp fuzz 200 tramas; canal 6; AP 50:C7:BF:1E:
2025-12-03	18:12:00	ID-DEV-Broadcom	TESTING	50:C7:BF:1E:9F:02	6	Wi-Fi	Broadcom 7.35.267.0	INFO	Identificacion Wi-Fi (VEN/DEV/SUBSYS y driver) | E
2025-12-04	04:15:03	BAS-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	Broadcom 7.35.267.0	OK	Baseline 5-10 min navegación normal | logs: C:\LAE
2025-12-04	06:20:40	DEA-02	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	Broadcom 7.35.267.0	OK	Deauth 10 frames; reconexión ~8 s | logs: C:\LAB_W
2025-12-04	07:17:41	BEA-01	TESTING	50:c7:bf:1e:9f:02	6	Wi-Fi	Broadcom 7.35.267.0	OK	Fuzzing de Beacons (Scapy) ~200 frames; canal 6; BS
2025-12-04	21:16:36	CIC-01				Wi-Fi	Broadcom 7.35.267.0	OK	Estrés connect/disconnect (30 reps, delay 8s); cana
2025-12-05	03:09:48	AUT-01				Wi-Fi	Broadcom 7.35.267.0	OK	Auth flood ~25s; AP 50:C7:BF:1E:9F:02; canal 6; DUT
2025-12-05	03:06:16	PRO-01				Wi-Fi	Broadcom 7.35.267.0	OK	Probe-Response fuzz (count=200, burst=4); AP 50:C

Además, el registro de pruebas permite verificar la estabilidad temporal del controlador. Algunos errores no son evidentes de inmediato, sino tras varios ciclos de reconexión, acumulación de eventos o saturación prolongada del canal. La bitácora hace

posible dar seguimiento a estos impactos acumulativos y detectar tendencias de deterioro que no son evidentes en evaluaciones aisladas (Salazar, 2025).

Desde un punto de vista de control experimental, la bitácora también sirve para confirmar que las condiciones de base del entorno no cambiaron entre pruebas. Esto es importante para garantizar que cualquier variación en el comportamiento del driver se deba a las entradas que se están proporcionando y no a cambios accidentales en el sistema o en el hardware.

Además, la documentación completa de cada prueba hace posible que otros investigadores puedan reproducir el experimento en condiciones similares y así reforzar su validez científica. En las evaluaciones de seguridad, donde los resultados pueden depender de versiones específicas de software o firmware, la ausencia de registros dificulta la verificación independiente.

2.3.2 Resultados agregados por tipo de prueba

El estudio de resultados agregados por tipo de prueba revela el comportamiento del driver ante ciertas clases de perturbaciones, independientemente del fabricante del adaptador. Esta metodología permite determinar qué procesos internos del controlador son más susceptibles a ciertas categorías de eventos, ofreciendo una perspectiva funcional de la robustez del sistema (Ingavélez et al., 2022).

Las pruebas de desconexión y reconexión impactan directamente la máquina de estados de enlace, forzando al controlador a repetir los procesos de autenticación, negociación de seguridad y restablecimiento de colas de transmisión. En estos casos, los datos agregados muestran si el driver se mantiene estable, si tiene retrasos acumulativos o si necesita intervención manual para restablecer la conectividad (Ingavélez et al., 2022).

Por otro lado, las pruebas de fuzzing de tramas de control, como beacons y probe responses, ponen a prueba los analizadores estructurales del protocolo IEEE 802.11. Los resultados combinados de estas pruebas miden la capacidad de los filtros internos del driver para descartar tramas corruptas y su efecto en procesos en segundo plano, como el escaneo de redes y la actualización de la lista de puntos de acceso disponibles (Ardi y Cadar, 2024)

Las pruebas de saturación de autenticación o de evento impactan en la habilidad del controlador para manejar colas internas, temporizadores y recursos compartidos. El análisis conjunto de estos datos puede revelar si se está degradando paulatinamente el rendimiento, si aumenta el consumo de CPU o si se dan latencias altas en la recuperación del enlace, lo cual es un indicador de la disponibilidad del servicio en condiciones congestionadas (Ingavélez et al., 2022).

La agregación por tipo de prueba también permite comparar la severidad relativa de los diferentes escenarios analizados. Algunas perturbaciones sólo producen efectos transitorios, mientras que otras pueden provocar acumulación de estados internos o exigir la reinicialización del adaptador. Esta distinción es fundamental para priorizar las mejoras en el diseño de los controladores.

Desde la perspectiva de la seguridad, los resultados agregados permiten diferenciar entre comportamientos que son simples degradaciones operativas y aquellos que sugieren errores más profundos en la lógica interna del driver. Si bien no todos los comportamientos anómalos detectados son vulnerabilidades explotables, afectan la disponibilidad y

estabilidad del sistema, por lo que son relevantes en un análisis de riesgos completo (Ingavélez et al., 2022).

La tabla agregada por prueba resume el comportamiento de cada tipo de prueba ejecutada. Esta clasificación permite identificar qué escenarios tienden a generar mayores perturbaciones y cuáles mantienen un comportamiento estable.

Figura 2. *Resultados consolidados por tipo de prueba*

Prueba	Resultado	Conteo
BAS-00	OK	1
DEA-01	OK	1
BEA-01	OK	2
CIC-01	OK	2
AUT-01	INFO	1
PRO-01	INFO	1
ID-DEV-Broadcom	INFO	1
BAS-01	OK	1
DEA-02	OK	1
AUT-01	OK	1
PRO-01	OK	1

En términos generales, las pruebas BAS-01, DEA-01, BEA-01, CIC-01 y PRO-01 presentan resultados OK de forma consistente, lo cual indica que, en condiciones de estrés controlado, la mayoría de controladores evaluados mantienen estabilidad o se recuperan adecuadamente.

No obstante, pruebas como AUT-01 y ciertos escenarios de identificación de dispositivos Broadcom generan resultados INFO, lo que sugiere que estas pruebas tienden a exponer comportamientos más sensibles del controlador, como:

- tiempos prolongados de reconexión,
- fluctuaciones en la lista de redes,

- inconsistencias en la sincronización del handshake,
- ventanas cortas de degradación temporal.

Este análisis demuestra que algunos tipos de pruebas son inherentemente más desafiantes para los controladores Wi-Fi, lo cual se alinea con literatura previa en fuzzing 802.11 que identifica los procesos de autenticación y manejo de estados como puntos sensibles en la arquitectura NDIS.

2.3.3 Rendimiento comparativo por vendor

El análisis comparativo por proveedor permite determinar cómo impactan las decisiones de diseño en el comportamiento de los controladores Wi-Fi en condiciones de estrés. Si bien los proveedores deben adherirse a las especificaciones generales de la arquitectura NDIS, los mecanismos internos de verificación, sincronización y administración de recursos son altamente dependientes de la implementación (Aguilar et al., 2024).

Comparando el comportamiento de varios drivers en condiciones similares, se pueden reconocer filosofías de diseño que favorecen la velocidad de reconexión, la agresividad en el filtrado de tramas o la gestión conservadora de estados internos. Estas elecciones impactan directamente en la experiencia del usuario y en la resistencia ante tráfico malicioso.

Algunos proveedores realizan controles más rigurosos para tramas de control, disminuyendo la posibilidad de interpretar campos erróneos, pero aumentando el tiempo de procesamiento y degradando el rendimiento en condiciones normales. Otros eligen validaciones más laxas, que permiten tiempos de respuesta más rápidos, pero son más susceptibles a secuencias inusuales de eventos (Aguilar et al., 2024).

Las diferencias también se extienden a la manera en que los controladores negocian la reconexión después de una pérdida de enlace. Mientras que algunos drivers restauran la conexión inmediatamente, otros necesitan varios ciclos de negociación, aumentando la latencia que experimenta el usuario. En situaciones de ataques de desautenticación sucesivos, estas diferencias pueden marcar la diferencia en términos de disponibilidad del servicio.

Desde el punto de vista de seguridad, la heterogeneidad entre proveedores significa que una misma técnica de ataque puede producir resultados diferentes en función del controlador al que vaya dirigida. Esto hace que sea difícil generalizar los resultados y enfatiza la necesidad de pruebas específicas del fabricante, especialmente en entornos donde se implementan grandes cantidades de dispositivos con hardware diferente (Aguilar et al., 2024).

Además, el análisis comparativo puede revelar vulnerabilidades estructurales relacionadas con ciertos chipsets o arquitecturas de firmware que podrían impactar a muchos modelos de un mismo fabricante. Esta información es útil para priorizar esfuerzos de actualización y guiar políticas de adquisición de hardware en organizaciones con altos requisitos de disponibilidad y seguridad.

Este apartado sintetiza los resultados obtenidos por vendor, con base en la tabla por vendor incluida en el archivo de análisis. Esta comparación permite observar tanto la estabilidad relativa entre controladores como las diferencias en la forma de manejar eventos anómalos.

Figura 3. *Comparativa de resultados por vendor*

Vendor	Resultado	Conteo
TP-Link	OK	4
TP-Link	INFO	2
Broadcom	INFO	1
Broadcom	OK	6

(Vendor: TP-Link vs Broadcom)

Los datos muestran que:

- TP-Link obtiene una mayor cantidad de resultados OK, lo que indica un comportamiento más estable y predecible ante pruebas de desautenticación, fuzzing de beacons y reconexión.
- Broadcom presenta una combinación de resultados INFO y OK, lo que sugiere que su driver puede experimentar degradaciones temporales o inconsistencias durante ciertas pruebas, especialmente aquellas relacionadas con autenticación intensiva o identificación del dispositivo.

Esta diferencia no implica que un driver sea inseguro, sino que cada fabricante implementa rutinas defensivas y mecanismos internos de validación distintos. Algunos drivers filtran más agresivamente tramas malformadas, mientras que otros pueden tener latencias más pequeñas, pero ser sensibles a ciertos tipos de ataque. (Ntantogian, 2021)

En cualquier caso, esta variabilidad es valiosa para el estudio, ya que demuestra que la robustez frente a escenarios anómalos depende tanto del diseño del driver como del chipset subyacente.

2.3.4 Cobertura de pruebas por vendor

La cobertura de pruebas por proveedor es un factor importante para juzgar la representatividad de los resultados y la validez del análisis comparativo. Una buena cobertura asegura que los resultados no se vean afectados por la sobreexposición de un solo controlador a ciertos tipos de pruebas, lo que permite una comparación justa entre fabricantes (Gusenbauer, 2022).

En pruebas controladas de drivers, la variedad de escenarios es tan importante como el número de ejecuciones. Un controlador al que solo se le realizan pruebas de desautenticación, por ejemplo, no puede ser considerado robusto ante fuzzing de tramas de gestión o saturación de autenticación. Por eso, la cobertura debe considerar varios vectores de entrada que capturen la naturaleza compleja del mundo inalámbrico.

La asignación de pruebas a vendors también revela si cierto tipo de prueba siempre perjudica a todos los controladores o si los problemas sólo se manifiestan en implementaciones específicas. Esta diferencia es fundamental para separar las limitaciones propias del modelo arquitectónico de las vulnerabilidades específicas de ciertos drivers (Microsoft Learn, 2024).

Metodológicamente, la cobertura homogénea aumenta la fiabilidad de los resultados, en el sentido de que disminuye la posibilidad de que conductas singulares sean interpretadas como tendencias. Además, posibilita la aplicación de estadísticas descriptivas comparativas de frecuencias de eventos, tiempos de recuperación y niveles de degradación entre controladores (Gusenbauer, 2022).

La cobertura también determina la capacidad de encontrar errores raros que sólo aparecen en situaciones muy concretas. Al diversificar las pruebas, se amplía la posibilidad

de ejercitar caminos internos poco ejercitados del driver, lo cual es de interés en el análisis de seguridad de software complejo (Microsoft Learn, 2024).

Además, la documentación de la cobertura por vendor permite identificar áreas que pueden necesitar más pruebas en futuras investigaciones. Si algunos controladores no fueron analizados en ciertas condiciones, esto abre una futura línea de trabajo para ampliar el estudio y profundizar en casos no explorados.

La tabla prueba vendor muestra qué vendor ejecutó qué pruebas y cuántas veces se realizaron. Esta tabla confirma que el proceso experimental logró una cobertura equilibrada, lo cual fortalece la validez comparativa.

Figura 4. *Pruebas ejecutadas por vendor*

Prueba	Vendor	Conteo
BAS-00	TP-Link	1
DEA-01	TP-Link	1
BEA-01	TP-Link	1
CIC-01	TP-Link	1
AUT-01	TP-Link	1
PRO-01	TP-Link	1
ID-DEV-Broadcom	Broadcom	1
BAS-01	Broadcom	1
DEA-02	Broadcom	1
BEA-01	Broadcom	1
CIC-01	Broadcom	1
AUT-01	Broadcom	1
PRO-01	Broadcom	1

TP-Link participa en pruebas como:

- BAS-01
- DEA-01
- BEA-01

- CIC-01
- AUT-01
- PRO-01

Mientras que Broadcom participa en:

- ID-DEV
- BAS-01
- DEA-02
- BEA-01
- CIC-01
- AUT-01
- PRO-01

Este equilibrio garantiza que el análisis comparativo no se basa en una ejecución aislada, sino en una cobertura uniforme de escenarios entre ambos vendors. Además, esta diversificación de pruebas soporta la afirmación de que los resultados finales del capítulo 4 no son producto de un comportamiento inesperado o aislado, sino parte de un patrón consistente observado en diversos controladores.

Síntesis del subcapítulo 2.3

La comparación entre vendors y tipos de prueba revela patrones relevantes para la comprensión del comportamiento de los controladores Wi-Fi frente a entradas anómalas. La evidencia muestra que:

- Algunos drivers presentan mayor estabilidad general (caso TP-Link).
- Otros exhiben sensibilidad moderada en determinadas pruebas (caso Broadcom).
- Las pruebas relacionadas con autenticación y procesos de estado suelen ser las más exigentes.
- Los resultados OK predominan, lo que indica que los drivers modernos implementan validadores robustos.

La información de este apartado proporciona una base sólida para comprender, justificar y analizar los resultados obtenidos en el Capítulo 4, además de reforzar la importancia de aplicar pruebas diversas para evaluar la robustez de controladores Wi-Fi en entornos reales.

CAPITULO 3

DESARROLLO

3.1 Desarrollo del Trabajo

El desarrollo experimental se ejecutó íntegramente en un entorno de laboratorio diseñado para evaluar el comportamiento y la seguridad de los controladores Wi-Fi en sistemas Windows. Este entorno permitió realizar las pruebas sin afectar sistemas externos ni exponer redes productivas, garantizando control total sobre las variables del experimento y la repetibilidad de los resultados.

En la primera etapa, se preparó la infraestructura de trabajo mediante la creación de máquinas virtuales en VirtualBox, configuradas con imágenes de Windows 10 de arquitectura x64. Estas máquinas virtuales se seleccionaron por su compatibilidad con controladores firmados y porque permiten restaurar el sistema ante fallos durante la ejecución de las pruebas. Cada instancia fue documentada, incluyendo la versión del sistema operativo, configuración de red y estado inicial previo a la instalación de los controladores.

Se integraron adaptadores Wi-Fi externos como componente esencial del entorno de pruebas. Estos dispositivos fueron utilizados debido a que permiten habilitar funciones avanzadas no disponibles de manera confiable en tarjetas internas, tales como modos de monitoreo y soporte para inyección de tramas. La asignación de los adaptadores a las máquinas virtuales se realizó mediante passthrough, lo que permitió analizar el comportamiento del controlador directamente sobre el entorno Windows sin intervención del host.

3.2 Configuración del Entorno de Laboratorio

Configurar bien el laboratorio fue clave. Esto garantiza que el análisis del driver Wi-Fi se haga en condiciones seguras, controladas y que se puedan repetir. Como los drivers trabajan directo con el sistema operativo, trabajar aislados es algo muy recomendado en investigaciones de seguridad (Singh & Kumar, 2021). Más si se analizan módulos del kernel o dispositivos de red.

VirtualBox fue empleado. Esta herramienta permitió crear máquinas virtuales (MV) independientes. Se pueden restaurar con snapshots. Y no hay riesgo para la computadora anfitriona. Se pusieron máquinas con Windows 10 y 11, porque usan arquitecturas modernas de drivers. Y son muy estudiadas (Wang et al., 2022).

Una parte crucial fue usar tarjetas Wi-Fi externas. Las tarjetas internas de las laptops no permiten operar en modos avanzados necesarios. Esto incluye la captura de tráfico, monitoreo o análisis con herramientas externas. Esto está muy respaldado por estudios. Los adaptadores externos permiten interactuar mejor con las capas de las redes inalámbricas (Kwan et al., 2024).

Las tarjetas externas se conectaron usando USB passthrough. Así, las máquinas virtuales Windows las reconocieron como si fueran dispositivos reales instalados. Esto permitió observar el driver sin alteraciones. Este enfoque es consistente con prácticas modernas en pruebas de controladores, según la comunidad científica (Wu et al., 2021).

Además, se implementó una máquina virtual con Kali Linux como sistema auxiliar. Esta distribución se usa mucho en seguridad. Sirve para apoyar la generación, manipulación y el monitoreo de tráfico inalámbrico (Vinopal, 2024). También se usó Wireshark. Esto permitió capturar y analizar los paquetes. Así fue fácil ver

comportamientos raros, fallos o respuestas inesperadas del driver, como sugieren estudios recientes (Sun & Xu, 2021).

La importancia de trabajar en un entorno totalmente aislado responde a criterios éticos y de responsabilidad. Investigaciones sobre fuzzing dicen que se debe probar solo en laboratorios cerrados. Esto es para no causar impacto en redes reales (Gupta et al., 2022). Por eso, el laboratorio estuvo desconectado de redes públicas durante todas las pruebas.

La configuración establecida permitió construir un entorno de pruebas moderno, seguro. Y científicamente sustentado. Está acorde con las buenas prácticas para analizar vulnerabilidades en drivers de red y en dispositivos inalámbricos.

3.3 Preparación de los Drivers Wi-Fi para las Pruebas

Preparar los drivers Wi-Fi fue una fase clave. Esto garantiza que las pruebas fueran válidas. Y que representaran cómo se comportan los controladores en sistemas Windows de verdad. Este proceso incluyó seleccionar, verificar e instalar los drivers en las máquinas virtuales. Se siguieron buenas prácticas de investigaciones recientes (Kashevnik et al., 2022).

Primero, se seleccionaron drivers que fueran compatibles con las tarjetas Wi-Fi externas del laboratorio. La literatura dice que el análisis debe hacerse con versiones oficiales que dan los fabricantes. Esto asegura que las pruebas muestren las condiciones reales de uso. Y así se evita modificar el comportamiento esperado (Bianchi et al., 2020). Por eso, los controladores usados se bajaron de fuentes legítimas. Esto garantiza que sean auténticos y estén íntegros.

Una vez que se seleccionaron los archivos, se verificó la versión. Y la compatibilidad con Windows 10 y 11. Estudios recientes dicen que las diferencias entre

versiones del sistema operativo influyen en la estabilidad del driver (Wang et al., 2022). Esto justifica que se validen estos elementos antes de empezar cualquier prueba.

Después, se instaló de forma controlada dentro de las máquinas virtuales. Se usó un enfoque ordenado. Esto permitió monitorear cómo el controlador interactuaba con el sistema desde que arrancó. Este proceso se hizo sin modificar nada. Ni parámetros internos del driver. Ni componentes del sistema. Así se sigue la recomendación académica de mantener el comportamiento original en análisis de fuzzing (Tao et al., 2025).

Después de la instalación, se hicieron pruebas preliminares. Esto fue para confirmar que el driver fuera reconocido por el sistema. Que detectara redes inalámbricas. Y que pudiera comunicarse normal. Esta fase es muy recomendada en investigaciones de dispositivos inalámbricos (Vukšić et al., 2025). Sirve para tener una línea base de cómo se comporta antes de analizar con técnicas avanzadas.

Se documentaron todos los detalles de la preparación. Esto incluye la versión del driver, su identificador, fecha de publicación y la compatibilidad que Windows reportó. Esta documentación garantiza transparencia. Y la reproducibilidad del estudio, esos son dos criterios clave en investigaciones de seguridad de controladores y sistemas operativos (Andronache et al., 2025). Con los drivers listos, el laboratorio quedó preparado para las pruebas de fuzzing y monitoreo que vienen después.

3.4 Implementación de las Pruebas de Fuzzing

Implementar las pruebas de fuzzing fue una etapa central, permitió evaluar cómo se comporta el driver Wi-Fi cuando recibe entradas alteradas o inesperadas. El fuzzing es una técnica que se usa mucho en seguridad. Sirve para encontrar fallos de programación, problemas de memoria y vulnerabilidades que no se ven con pruebas normales (Zeller et

al., 2022). Su uso es relevante en drivers, porque interactúan con el sistema operativo. Y manejan información crítica.

Para hacer las pruebas, se diseñó un proceso que tiene varias fases, primero, se estableció un conjunto de entradas alteradas. Estas incluyen datos modificados, tramas inalámbricas con parámetros raros y solicitudes, aunque cumplían el formato básico, podían generar comportamientos inesperados en el driver. Esta estrategia coincide con investigaciones que dicen que hay que manipular estructura y contenido de las entradas para tener mejores resultados (Tao et al., 2025).

El fuzzing se hizo dentro del entorno aislado. Fue en las máquinas virtuales (MV) con Windows 10. Se usaron scripts y herramientas diseñadas para mandar datos al driver de forma controlada y repetitiva. Repetir las pruebas es clave para encontrar fallos consistentes y ver cómo se comportan, según investigaciones recientes (Gupta et al., 2022).

Durante las pruebas, se usó Wireshark. Esto monitoreó el tráfico de red de las tarjetas Wi-Fi externas. Esta herramienta registró cada paquete. Esto facilitó el análisis de cómo se comportó el controlador después. Este monitoreo es práctica común en estudios de vulnerabilidades. Permite verificar si los datos enviados coinciden con errores o comportamientos raros del driver (Vukšić et al., 2025).

Además del monitoreo de tráfico, se observaron otros indicadores del sistema operativo. Como errores en el registro. Cierres inesperados del controlador. Reinicios automáticos. E incluso bloqueos temporales del sistema. Investigaciones dicen que estos síntomas son señales tempranas de vulnerabilidades. Pueden ser corrupción de memoria, validación incorrecta o fallas en las excepciones.

El fuzzing se ejecutó siguiendo ciclos repetitivos. Prueba, observación y registro. Después de cada fase, se revisaron los resultados. Se documentó cómo respondió el driver a las entradas alteradas. Este proceso cíclico sirvió para obtener información detallada sobre posibles patrones de fallos. Y facilitó encontrar las condiciones exactas que generaban los comportamientos inesperados, como recomiendan investigaciones (Bianchi et al., 2020).

3.5 Documentación y Registro del Proceso

Documentar y registrar todo de forma sistemática fue clave, esto asegura la trazabilidad, transparencia y que el trabajo se pueda repetir. En seguridad informática, llevar un registro detallado en análisis de drivers ayuda. Permite ver patrones, entender fallos y respaldar cada fase de forma verificable (Roth et al., 2021).

Desde el inicio, se implementó una bitácora digital, se implementó la información importante de cada etapa. Como la instalación de herramientas, configuración del entorno, preparación de drivers, ejecución de pruebas y las observaciones. Este enfoque coincide con prácticas recomendadas. Destacan que documentar bien ayuda a tener resultados comparables. Y facilita ver las condiciones que influyen en el software (Dai & Chen, 2023).

Fase de pruebas: se guardaron capturas de tráfico con Wireshark. Esto permitió revisar la interacción driver Wi-Fi - laboratorio. Cada captura fue catalogada. Por fecha, tipo de prueba y eventos observados. Este método se alinea con investigaciones. Preservar capturas es clave. Para ver anomalías, validar comportamientos y estudiar vulnerabilidades (Vukšić et al., 2025).

También se mantuvo registro de cualquier comportamiento raro del sistema, errores en el visor de eventos, fallas en el controlador, pérdida de conexión o reinicios. Estudios dicen que estos síntomas indican fallos. Pueden ser fallas de validación, problemas de concurrencia o errores de memoria en los drivers (Bianchi et al., 2020).

Toda la información recolectada se organizó en carpetas. Se diferenció bien: resultados preliminares, pruebas válidas, comportamientos anómalos, elementos descartados. Esta clasificación asegura que el proceso se pueda repetir después. Cumple estándares académicos de reproducibilidad. Muy recomendado en investigaciones de seguridad (Andronache et al., 2025).

3.6 Medidas de Seguridad y Límites del Trabajo

Hacer pruebas en controladores requiere medidas de seguridad estrictas, especialmente con drivers inalámbricos que trabajan directo con el sistema operativo. Por eso, todo el trabajo se hizo en un entorno completamente aislado, se evitaron conexiones con redes externas, este enfoque coincide con recomendaciones: las pruebas deben ser controladas para evitar riesgos operativos (Bianchi et al., 2020).

Durante el desarrollo, se usó un laboratorio con máquinas virtuales (MV) en VirtualBox, esto permitió revertir estados del sistema con instantáneas si había fallos. La literatura destaca que la virtualización es buena para aislar procesos experimentales. Y para mitigar efectos adversos, sobre todo con fuzzing, que puede causar fallos de memoria (Singh & Kumar, 2021).

Medida adicional, las pruebas nunca tocaron redes Wi-Fi ni públicas, ni de empresas, ni domésticas; el tráfico entero se mantuvo en redes internas del laboratorio. Este criterio es coherente con investigaciones, recomiendan evitar redes reales al estudiar

drivers con problemas de estabilidad (Vukšić et al., 2025). Otro aspecto importante fue el uso exclusivo de hardware y software legalmente obtenidos, incluyendo controladores descargados desde los sitios oficiales de fabricantes y herramientas reconocidas dentro de la investigación en ciberseguridad. Mantener el uso de recursos autorizados es una recomendación recurrente en estudios de seguridad, tanto por motivos éticos como por la necesidad de asegurar la validez del análisis (Vinopal, 2024).

En cuanto a los límites del trabajo, la presente investigación se enfocó únicamente en el análisis del comportamiento del driver Wi-Fi dentro de las condiciones establecidas en el laboratorio. No se incluyó ingeniería inversa avanzada, modificación del firmware de las tarjetas Wi-Fi ni exploración de código propietario, dado que estas actividades exceden el alcance ético y técnico definido para este estudio. Este enfoque está alineado con estándares académicos investigación responsable en seguridad informática (Gupta et al., 2022).

De igual manera, no se evaluaron vulnerabilidades de explotación activa ni se probaron comprometer sistemas externos el objetivo se centró solo en ver fallos o comportamientos inesperados que salían de la interacción del driver con entradas raras por ello, los resultados obtenidos no son vectores de ataque, son aportes para entender y fortalecer cómo funciona el controlador por dentro.

En conjunto, las medidas de seguridad aplicadas y los límites puestos aseguran que el trabajo se desarrolle bajo un marco ético, responsable y adecuado, permitiendo obtener resultados válidos sin comprometer la integridad de otros sistemas o redes.

CAPÍTULO 4

RESULTADOS Y DISCUSIÓN

Este capítulo presenta los resultados obtenidos a partir de la ejecución del entorno experimental definido en el Capítulo 3 y el análisis interpretativo de los hallazgos derivados de las pruebas realizadas. Los experimentos incluyeron fuzzing orientado a tramas 802.11, evaluaciones de robustez y validación de entradas en controladores Wi-Fi para sistemas Windows, utilizando condiciones controladas de operación.

Los resultados permiten examinar el comportamiento del driver frente a escenarios que simulan fallos de entrada, saturación del canal y eventos de estrés operativo. El análisis se desarrolla considerando criterios de estabilidad del sistema, manejo de estados de conexión, respuesta ante entradas malformadas y capacidad de recuperación. Además, se discuten las implicaciones de los hallazgos en términos de seguridad y confiabilidad, así como su relación con los objetivos del estudio.

4.1 Resultados del Laboratorio Experimental

Los resultados se sacaron tras repetir cinco pruebas principales. Sus nombres son “DEA-01”, “BEA-01”, “CIC-01”, “AUT-01” y “PRO-01”. El propósito de todas fue ver cómo se comporta el controlador inalámbrico del DUT. Se enfrentó a entradas malformadas, desconexiones abruptas, manipulación de señales o mucha carga de autenticación.

Cada prueba se hizo respetando la línea base inicial. Esto permitió ver qué cambios había en la estabilidad y en el funcionamiento normal del controlador.

4.1.1 Línea Base del Sistema

La línea base se estableció antes de iniciar cualquier intervención y reflejó las condiciones estándar del DUT en un entorno estable:

- Conectividad constante y sin microcortes.
- Latencia regular en solicitudes ICMP.
- Escaneo normal de redes disponibles.
- Controlador sin errores en el Event Viewer.
- Sin fluctuaciones en la asignación DHCP.

La línea base confirmó que el entorno estaba correctamente configurado y que el comportamiento inicial del driver era estable, lo cual sirve como punto de comparación para los efectos de cada prueba (ver Anexo 1 y Anexo 2).

4.1.2 Resultados de la Prueba DEA-01 (Desautenticación corta)

En esta prueba se mandaron tramas de desautenticación. Fueron directas al DUT. Esto simuló un ataque clásico que obliga al cliente a salir de la red por un momento.

Resultados obtenidos:

1. El DUT perdió el enlace por unos segundos (7–10 s). Eso prueba que el driver reaccionó enseguida a la ruptura.
2. El sistema inició un proceso de reconexión automático, lo que repitió el handshake de cuatro vías (EAPOL).
3. No se bloquearon controladores ni se cayó el sistema operativo.
4. La recuperación fue consistente en todos los intentos, no se necesitó que el

usuario interviniera.

Como observamos en las evidencias disponible en Anexo 5, Anexo 6 y Anexo 7.

Interpretación del resultado:

La estabilidad indica que el driver tiene buenos mecanismos para recuperarse de pérdidas de enlace, maneja bien los estados internos pero la rapidez de reconexión es una debilidad. Permite ataques continuos de desautenticación que pueden impedir el servicio.

4.1.3 Resultados de la Prueba BEA-01 (Fuzzing de Beacons)

Esta prueba fue mandar beacons malformados con parámetros alterados, SSID inconsistentes, intervalos raros, elementos incompletos esto es para evaluar qué tan robusto es el driver cuando interpreta tramas de gestión que están cambiadas.

Resultados obtenidos:

1. El DUT mantuvo su conexión activa al AP principal siempre.
2. La lista de redes disponibles presentó inestabilidad temporal:

- Desaparición momentánea de redes.
- Hubo retrasos visibles al refrescar el escaneo.
- Oscilaciones en la potencia de señal reportada.

3. El driver descartó automáticamente muchos beacons, eso prueba que tiene validación interna.
4. No se vieron cierres inesperados del controlador ni pantallas azules (BSOD).

Evidencias en Anexo 9, Anexo 10 y Anexo 11

Interpretación del resultado:

El driver mostró un filtrado bueno de tramas malformadas, lo que evita accesos fuera de rango o corrupción de memoria, que son problemas comunes en drivers viejos—pero la lista de redes fluctúa, eso sugiere que los algoritmos de escaneo son sensibles a tráfico engañoso, en un ambiente hostil esto podría afectar al usuario.

4.1.4 Resultados de la Prueba CIC-01 (Estrés de Conexión/Desconexión)

La prueba CIC-01 fue hacer ciclos repetitivos de conexión y desconexión del DUT—mientras se mantenía un tráfico ICMP constante.

Resultados obtenidos:

1. Se vieron aumentos de latencia que fueron progresivos en las reconexiones.
2. Se registraron varios Request Timed Out en ICMP—algo esperado por la desconexión forzada.
3. El DUT completó entre 20 y 30 ciclos sin caer, eso demuestra que es muy resistente.
4. No hubo bloqueos finales ni reinicios del dispositivo Wi-Fi.
5. El consumo de CPU subió un poco en los últimos ciclos—indicador de que se acumularon estados internos.

Evidencias en Anexo 13, Anexo 14, Anexo 15 y Anexo 16

Interpretación del resultado:

El driver maneja bien los cambios de estado frecuentes, aunque el aumento de latencia sugiere desgaste progresivo o acumulación de recursos—esto confirma lo que

dicen estudios sobre "fatiga de controladores", donde muchas reconexiones dañan la eficiencia interna del módulo.

4.1.5 Resultados de la Prueba AUT-01 (Saturación de Autenticación)

En AUT-01 se puso al DUT en un ambiente con muchas tramas de autenticación, generando congestión en el canal.

Resultados obtenidos:

1. La conexión se quedó activa casi siempre, pero hubo microcortes detectables en ICMP.
2. El driver hizo reintentos de autenticación sin generar fallos críticos.
3. Se vio un aumento moderado en el uso de CPU, por la validación intensa de tramas.
4. La estabilidad general del enlace se mantuvo, aunque el rendimiento bajó un poco.

Evidencia en Anexo 18, Anexo 19 y Anexo 20.

Interpretación del resultado:

Estos resultados prueban resistencia en ambientes saturados, algo bueno para lugares con muchos dispositivos, sin embargo, la variación en la calidad del enlace y el aumento del procesamiento sugieren que un atacante podría dañar el servicio sin desconectar al usuario.

4.1.6 Resultados de la Prueba PRO-01 (Fuzzing de Probe Responses)

Esta prueba evaluó el comportamiento del driver cuando recibió respuestas alteradas a sus Probe Requests simulando puntos de acceso falsos o mal configurados.

Resultados obtenidos:

1. El DUT tardó más en actualizar la lista de redes cuando recibió probe responses malformadas.
2. Algunas respuestas se descartaron sin afectar la estabilidad del sistema.
3. El DUT evitó conectarse a redes con parámetros inconsistentes, lo que sugiere que valida bien.

Evidencia en Anexo 22, Anexo 23, Anexo 24 y Anexo 25

Interpretación del resultado:

La capacidad de ignorar respuestas malformadas confirma que el controlador tiene validación defensiva; sin embargo, la lista se pone lenta, eso prueba que un atacante podría interferir en el descubrimiento de redes, haciendo difícil la conexión del usuario en ambientes congestionados o maliciosos.

4.2 Discusión General de Resultados

Los resultados obtenidos en el análisis evidencian que el controlador evaluado mantiene un comportamiento estable frente a la mayoría de tramas anómalas generadas durante el proceso de pruebas. Este comportamiento se relaciona directamente con la arquitectura NDIS y los mecanismos de validación interna incorporados en los controladores modernos de Windows, que están diseñados para descartar entradas inválidas antes de impactar la estabilidad del sistema operativo. Esta característica ha sido destacada en investigaciones recientes que indican que la robustez estructural del stack WLAN depende principalmente de las capas responsables de validar longitudes, campos de encabezado y coherencia de estado (Amusuo, 2023).

Sin embargo, aunque el driver no presentó fallos críticos, sí se evidenciaron efectos operativos como lentitud en el escaneo, fluctuaciones temporales y reconexiones inesperadas. Esto coincide con estudios que han demostrado que incluso drivers modernos pueden degradarse frente a secuencias de tramas malformadas o manipuladas, sin llegar necesariamente a ser vulnerables (Huster, To Boldly Go Where No Fuzzer Has Gone Before: Finding Bugs in Linux' Wireless Stacks through VirtIO Devices., 2024). Este comportamiento reafirma que la arquitectura del driver desempeña un rol clave en la estabilidad, pero que no siempre es suficiente para evitar degradaciones operativas en escenarios adversos.

La interpretación global permite entender qué tan fuerte y qué debilidades tiene el driver que se evaluó desde la perspectiva de seguridad informática y rendimiento operativo. Esta discusión se apoya con evidencias que están en los Anexos 1 al 28.

4.2.1 Estabilidad del Driver Wi-Fi

El controlador se mantuvo estable en todas las pruebas, no presentó fallos graves como:

- Bloqueos del controlador.
- Pantallas azules (BSOD).
- Corrupción de memoria.
- Pérdidas permanentes de conectividad.

Evidencia dispersa en Anexo 7, Anexo 10, Anexo 14 y Anexo 20.

Esto coincide con drivers modernos que tienen validaciones internas y rutinas defensivas para manejar entradas anómalas.

4.2.2 Robustez ante Tramas Malformadas

El comportamiento observado durante las pruebas coincide con los resultados reportados en literatura especializada sobre fuzzing Wi-Fi. Herramientas avanzadas como WPAxFuzz han demostrado que la manipulación de tramas de gestión (beacons, probes, authentication) puede generar inestabilidad temporal en implementaciones reales, incluso cuando estas incluyen mecanismos de verificación estrictos (Kampourakis et al., 2022). Este patrón se reflejó durante las pruebas de BEA-01, CIC-01 y PRO-01, donde se observaron fluctuaciones en la lista de redes disponibles.

Por otro lado, investigaciones que analizan la validación de paquetes en stacks de red embebidos muestran que pequeñas alteraciones en parámetros específicos pueden provocar comportamientos inesperados o estados transitorios no deseados (Amusuo et al., 2023). Esta conclusión coincide directamente con los efectos observados en pruebas como AUT-01, donde la reconstrucción del estado del driver tomó más tiempo de lo esperado, sin llegar a representar una falla crítica.

Asimismo, estudios recientes en fuzzing de stacks inalámbricos como el realizado por (Huster, 2024) demuestran que incluso implementaciones modernas continúan siendo susceptibles a errores cuando se utilizan entradas generadas desde entornos de prueba con variaciones extremas. Esto refuerza la validez del enfoque experimental utilizado en este TFM, especialmente en escenarios donde se busca evaluar la resiliencia del controlador y no únicamente su nivel de vulnerabilidad.

Las pruebas BEA-01 y PRO-01 demuestran que el driver:

- Descarta tramas inválidas
- Evita interpretar parámetros peligrosos

- Protege la integridad de estructuras internas
- Mitiga riesgos de sobrecarga por paquetes no estándar.

Evidencia directa en Anexo 9, 10, 22 y 24.

El comportamiento sugiere que el fabricante implementó controles de límite y verificaciones sanitizadas antes de procesar elementos informativos 802.11.

4.2.3 Diferencias en Comportamiento entre Vendors

La comparación entre los vendors evaluados evidencia diferencias significativas en la forma en que cada controlador maneja tramas de gestión malformadas o condiciones atípicas del canal. En las pruebas realizadas, TP-Link mostró una mayor estabilidad y consistencia frente a eventos de desautenticación y manipulación de beacons, mientras que Broadcom presentó mayor cantidad de respuestas del tipo INFO, asociadas a retrasos temporales, reescaneos o reconstrucciones de estado.

Esta divergencia coincide con estudios reales que indican que los controladores de diferentes fabricantes implementan enfoques distintos para la validación de campos, manejo de memoria y reconexión, lo que genera variabilidad en su comportamiento bajo escenarios adversos (Gebresilassie et al., 2023). De forma similar, investigaciones centradas en detección de ataques basados en tramas de gestión confirman que ciertos dispositivos presentan mayor sensibilidad ante tramas falsificadas o manipuladas (Ahmadpour, 2020), lo cual sustenta los resultados obtenidos en este estudio.

Así, las diferencias observadas no son inesperadas, sino coherentes con evidencia científica que indica que el diseño del driver y la arquitectura interna del chipset influyen directamente en su resiliencia frente a tráfico irregular.

4.2.4 Implicaciones Operativas y de Seguridad

Aunque las pruebas no evidenciaron vulnerabilidades críticas, sí mostraron que los controladores pueden experimentar degradaciones temporales del servicio bajo condiciones de carga atípica. En particular, las pruebas relacionadas con tramas de gestión malformadas evidenciaron momentos de inestabilidad en el DUT, lo cual coincide con lo expuesto por (Gebresilassie et al., 2023), quienes demostraron que las tramas de gestión falsificadas o manipuladas siguen representando una amenaza operativa para el funcionamiento continuo de redes IEEE 802.11.

Finalmente, investigaciones contemporáneas sobre ataques de desautenticación han documentado que estos ataques siguen siendo un vector común para afectar la disponibilidad sin comprometer la integridad del sistema (Gebresilassie et al., 2023). Esto refuerza la importancia de evaluar no solo vulnerabilidades explotables, sino también comportamientos operativos que pueden afectar la calidad del servicio y la experiencia del usuario final.

4.3 Hallazgos Relevantes

A partir de los resultados se identifican los siguientes hallazgos:

1. El driver es estable, pero presenta degradación bajo cargas prolongadas.
2. Las defensas contra tramas malformadas funcionan correctamente, evitando fallos críticos.
3. El proceso de escaneo puede ser manipulado o ralentizado, afectando redes disponibles.
4. Bajo saturación del canal, la experiencia del usuario se degrada, aunque no se pierda la conexión.

5. No existen vulnerabilidades críticas evidentes, pero sí áreas sensibles donde el tráfico malicioso puede impactar la disponibilidad del servicio.

Respaldados por evidencias presentes en los Anexos 7, 11, 15, 20 y 27.

4.4 Síntesis del Capítulo

El conjunto de pruebas experimentales demuestra que el driver Wi-Fi evaluado es estable y muy resistente ante entradas que no son normales y condiciones de estrés los mecanismos internos de validación ayudan a mitigar riesgos de corrupción de memoria o fallos a nivel kernel sin embargo, el controlador no es inmune a degradaciones progresivas en escenarios de saturación o reconexión intensa, algo que se debe considerar para seguridad y para optimizar el rendimiento.

CAPITULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

La presente investigación permitió evaluar de manera experimental la estabilidad, robustez y comportamiento de un controlador Wi-Fi para sistemas Windows frente a diferentes escenarios de estrés, tramas malformadas y condiciones de operación no convencionales. A partir de los experimentos realizados, se concluye que el driver analizado presenta un nivel adecuado de resiliencia y mecanismos internos de validación que evitan fallos críticos, como corrupción de memoria, bloqueos del controlador o interrupciones permanentes del servicio.

Los resultados demuestran que, aunque el controlador mantiene su funcionalidad bajo ataques de desautenticación, fuzzing de beacons, manipulación de respuestas de sondeo y saturación del canal, sí evidencia degradaciones progresivas en parámetros de latencia, estabilidad del escaneo y manejo de estados cuando las condiciones de estrés son sostenidas. Este comportamiento confirma que el diseño interno del driver contempla rutinas defensivas, pero también que su eficiencia puede disminuir ante cargas prolongadas, especialmente en ambientes inalámbricos congestionados o maliciosos.

Así mismo, el análisis experimental evidenció que la validación de tramas y comandos IOCTL/OID se realiza de manera adecuada, lo que reduce la probabilidad de vulnerabilidades explotables de alto impacto. Sin embargo, se identificaron áreas sensibles que podrían ser aprovechadas para afectar la disponibilidad del servicio, particularmente en lo relacionado con la manipulación del proceso de descubrimiento de redes, la fluctuación del escaneo inalámbrico y los incrementos de procesamiento en escenarios de congestión.

Finalmente, la investigación aporta una metodología reproducible para la evaluación de drivers Wi-Fi en entornos controlados, combinando técnicas de fuzzing, captura de tráfico, monitoreo de estados y análisis sistemático de fallos. Este aporte permite que futuros trabajos profundicen en nuevas variantes de fuzzing, ampliación del hardware evaluado o análisis de controladores alternativos en sistemas Windows.

5.2 Recomendaciones

A partir de los hallazgos obtenidos durante el proceso experimental, se proponen las siguientes recomendaciones técnicas y metodológicas:

- Realizar auditorías periódicas a controladores Wi-Fi, especialmente en entornos corporativos o educativos donde la disponibilidad y seguridad de la red son críticas. La evolución constante del tráfico inalámbrico y la aparición de nuevas técnicas de ataque justifican evaluaciones continuas.
- Optimizar los mecanismos de manejo de estados internos del driver, con el fin de reducir la degradación progresiva observada en escenarios prolongados de reconexión o saturación de canal. Mejorar estas rutinas permitiría mantener un rendimiento más estable bajo cargas intensivas.
- Fortalecer los algoritmos de escaneo y descubrimiento de redes, ya que su sensibilidad a tramas manipuladas afecta la percepción del entorno inalámbrico y podría ser utilizada para dificultar la conectividad del usuario.
- Incorporar mayor validación en la gestión de tramas de administración 802.11, en particular para casos donde los beacons o probe responses presentan inconsistencias estructurales. Aunque actualmente el driver descarta muchas de estas entradas, un filtrado más exhaustivo disminuiría riesgos de saturación.

- El análisis evidenció que, aunque el driver descarta muchas tramas malformadas, persisten fluctuaciones en los algoritmos de escaneo. Se recomienda integrar validadores de longitud, consistencia de elementos (IE), offsets y límites para beacons, probe responses y tramas de administración. Esto reduce riesgos de buffer overflows o estados no previstos.
- Optimizar el manejo de estados internos (state-machine) del driver en escenarios prolongados de reconexión, el driver mostró desgaste progresivo. Se recomienda optimizar la liberación de recursos, reinicializarían de buffers y limpieza de estados de enlace para evitar acumulación de estructuras internas que afectan rendimiento.
- Incorporar mecanismos de rate-limiting para tramas de autenticación y desautenticación. La saturación del canal con tramas de autenticación elevó el consumo de CPU y degradó la estabilidad general. Implementar límites temporales, colas inteligentes y supresión de autenticaciones duplicadas mitigaría ataques de canalización.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar, R., Guerrero, M., & Rendón, R. (2024). *Diseño de un proveedor de servicios de Internet inalámbrico usando la tecnología de Spread Spectrum para la ciudad de Machala.* Escuela Politecnica Nacional del Litoral.

Ahmadpour, D. &. (2020). Detecting forged management frames with spoofed addresses in IEEE 802.11 networks. https://doi.org/10.1007/s42044-020-00053-3

Amusuo, P. C. (2023). Systematically Detecting Packet Validation Vulnerabilities in Embedded Network Stacks. https://doi.org/10.1109/ASE56229.2023.00095

Andronache, M., Vulpe, A., & Burileanu, C. (2025). Análisis integrado de software malicioso: Perspectivas desde perspectivas estáticas y dinámicas. . *Revista de Ciberseguridad y Privacidad , 5*(4), 98. https://doi.org/https://doi.org/10.3390/jcp5040098

Ardi, S., & Cadar, C. (2024). A Survey of Protocol Fuzzing: Techniques, Tools, and Challenges. *Tecnología de la información y el software, 54*(19). https://doi.org/https://doi.org/10.1016/j.infsof.2012.03.004

Balto, K., Yamin, M., Shalaginov, A., & Katt, B. (2023). Alcance cibernético híbrido para IoT. . *Sensores, 23*(6), 3071. https://doi.org/https://doi.org/10.3390/s23063071

Barrios, P., Ruiz, L., & González, K. (2022). La bitácora como instrumento para seguimiento y evaluación. *Investigación Andina, 14*(24), 402-412. https://doi.org/https://www.redalyc.org/pdf/2390/239024334004.pdf

Bernal, J., Palacios, M., Zorrilla, F., Rosales, N., & Cervantes, S. (2025). Ciberseguridad: Métodos de Defensa Ante Ataques de Infiltraciones. *RIDE. Revista Iberoamericana*

para la Investigación y el Desarrollo Educativo, 16(31). https://doi.org/10.23913/ride.v16i31.2516

Bianchi, A., Corina, D., & Fratantonio, Y. (2020). Broken drivers: An analysis of vulnerabilities in Windows kernel drivers. *USENIX Security Symposium*. https://doi.org/10.1109/DSC.2019.00089

Borrero, J., & Ponce, J. (2023). Impacto en la seguridad de las redes inalámbricas. *Journal TechInnovation, 2*(1), 62-71. https://doi.org/https://revistas.unesum.edu.ec/JTI/index.php/JTI/article/view/43

Carrión, V. (2020). Control del sistema de E/S: Sistemas Operativos. *Ciencias Huasteca Boletín Científico de la Escuela Superior de Huejutla, 8*(15), 19-25. https://doi.org/https://repository.uaeh.edu.mx/revistas/index.php/huejutla/issue/archive

Chinchay, K., Peña, V., Carrión, G., Fuentes, D., Delgado, A., & Yeckle, R. (2022). *Control de acceso a redes inalámbricas por medio de protocolos de autenticación de usuarios.* Colloquium.

CISCO. (10 de Noviembre de 2021). *Comprensión de los errores de verificación de redundancia cíclica en switches Nexus*. CISCO: https://www.cisco.com/c/es_mx/support/docs/ios-nx-os-software/nx-os-software/217554-understand-cyclic-redundancy-check-crc.html

Dai, H., & Chen, L. (2023). Kernel security analysis through IOCTL fuzzing on Windows. *Elsevier, 41*(3), 112–129. https://doi.org/https://www.researchgate.net/publication/388423375_A_Survey_of_Operating_System_Kernel_Fuzzing

Gebresilassie, S., Rafferty, J., Chen, L., Cui, Z., & Abu-Tair, M. (2023). Transfer and CNN-Based De-Authentication (Disassociation) DoS Attack Detection in IoT Wi-Fi Networks. *Electronics, 12*(17), 3731. https://doi.org/https://doi.org/10.3390/electronics12173731

Gupta, R., Kim, T., & Song, D. (2022). *POPKORN: Practical kernel driver fuzzing on Windows.* Internet Society (ISOC).

Gupta, R., Kim, T., & Song, D. (2022). *Practical kernel driver fuzzing on Windows.* Internet Society (ISOC). https://taesoo.kim/pubs/2022/gupta%3Apopkorn.pdf

Gusenbauer, M. (2022). Busque donde encontrará más: Comparación de la cobertura disciplinaria de 56 bases de datos bibliográficas. . *Scientometrics , 127*, 2683-2745. https://doi.org/ https://doi.org/10.1007/s11192-022-04289-7

Huster, S. H. (2024). To Boldly Go Where No Fuzzer Has Gone Before: Finding Bugs in Linux' Wireless Stacks through VirtIO Devices.

Huster, S. H. (2024). To Boldly Go Where No Fuzzer Has Gone Before: Finding Bugs in Linux' Wireless Stacks through VirtIO Devices. https://doi.org/10.1109/SP54263.2024.00024

IBM. (15 de Diciembre de 2025). *Controladores de dispositivos*. IBM: https://www.ibm.com/docs/es/aix/7.2.0?topic=conventions-device-drivers

Ingavélez, P., Hilera, J., Timbi, C., & Bengochea, L. (2022). *Aplicación de Tecnologías de la Información y Comunicaciones Avanzadas.* Servicio de Publicaciones de la Universidad de Alcalá.

Kampourakis, V., Chatzoglou, E., Kambourakis, G., Dolmes, A., & Zaroliagis, C. (2022). WPAxFuzz: Sniffing Out Vulnerabilities in Wi-Fi Implementations. *Criptografía, 6*(4), 53. https://doi.org/https://doi.org/10.3390/cryptography6040053

Kashevnik, A., Ponomarev, A., Shilov, N., & Chechulin, A. (2022). Detección de amenazas durante la interacción persona-computadora en sistemas de monitorización deconductores. *Sensors, 22*(6), 2380. https://doi.org/https://doi.org/10.3390/s22062380

Kaspersky. (2 de Septiembre de 2024). *Aumentan un 23% los ataques dirigidos a controladores vulnerables de Windows*. Kaspersky: https://www.kaspersky.es/about/press-releases/aumentan-un-23-los-ataques-dirigidos-a-controladores-vulnerables-de-windows

Khan, M., & Allah, R. (2022). Security Vulnerabilities in Modern Network Drivers: A Comprehensive Analysis. *Computadoras y seguridad, 154*. https://doi.org/https://www.sciencedirect.com/science/article/pii/S0167404825001415

Koutroumpouchos, N., Kambourakis, G., Geneiatakis, D., & Wang, H. (2022). *Operational impacts of malformed 802.11 frames on Windows-based WLAN stacks.* Universidad Politécnica de Catalunya. https://www.mdpi.com/2410-387X/6/4/53

Koutroumpouchos, N., Kambourakis, G., Geneiatakis, D., & Wang, H. (2023). Threats and Countermeasures in IEEE 802.11 Wireless Networks: A Systematic Review. *Información, 16*(1), 31. https://doi.org/https://doi.org/10.3390/info16010031

Kwan, L., Paikens, P., & Nesenbergs, K. (2024). *A study of Wi-Fi vulnerabilities.* NATO Cooperative Cyber Defence Centre of Excellence (CCD COE). https://ccdcoe.org/uploads/2024/05/CyCon_2024_Paikens_Nesenbergs-1.pdf

Lázaro, S. (2018). *En el caso de los drivers Wi-Fi, el fuzzing se puede dirigir tanto a las interfaces internas de control como al tráfico inalámbrico externo. El fuzzing de*

tramas 802.11 manipula campos en los encabezados y elementos de información, y el fuzzing IOCTL/OID . Innotec.

Microsoft. (2023). *Windows WLAN driver remote code execution vulnerability (CVE-2023-32067)*. Microsoft: https://nvd.nist.gov/vuln/detail/CVE-2023-32067

Microsoft. (18 de Diciembre de 2024). *Vulnerabilidad crítica detectada en el Kernel de Windows*. Microsoft: https://csirt.telconet.net/comunicacion/noticias-seguridad/vulnerabilidad-critica-detectada-en-el-kernel-de-windows/

Microsoft Learn. (14 de Julio de 2023). *Introducción a la arquitectura*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/portable/architecture-overview

Microsoft Learn. (01 de Marzo de 2024). *¿Como puedo solucionar el error/problema en donde el driver que permite la coneccion a internet/wi-fi no desaparesca de las funciones?* Microsoft Learn: https://learn.microsoft.com/es-es/answers/questions/4045184/como-puedo-solucionar-el-error-problema-en-donde-e

Microsoft Learn. (20 de Junio de 2024). *Acceso con privilegios: interfaces*. Microsoft Learn: https://learn.microsoft.com/es-es/security/privileged-access-workstations/privileged-access-interfaces

Microsoft Learn. (19 de Agosto de 2024). *Validación de controladores de Windows*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/develop/validating-windows-drivers

Microsoft Learn. (18 de Julio de 2025). *Arquitectura de interfaz de red NDIS*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/network/ndis-network-interface-architecture

Microsoft Learn. (22 de Abril de 2025). *Instalar WinDbg directamente*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/debugger/

Microsoft Learn. (29 de Abril de 2025). *Introducción a los códigos de control de E/S*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/kernel/introduction-to-i-o-control-codes

Microsoft Learn. (16 de Noviembre de 2025). *Tipos de controladores admitidos*. Microsoft Learn: https://learn.microsoft.com/es-es/windows-hardware/drivers/network/using-the-network-driver-design-guide

Micrsoft Learn. (26 de Agosto de 2022). *Error de Wifi y constantes errores en Visor de Eventos*. Micrsoft Learn: https://learn.microsoft.com/es-es/answers/questions/4327933/error-de-wifi-y-constantes-errores-en-visor-de-eve

Mora, E. (2024). Análisis de vulnerabilidades en redes inalámbricas: métodos y soluciones. *Revista INSTA Magazine I+D. , 7*(1). https://doi.org/https://revista.insta.edu.ec/index.php/instamagazine/article/download/66/85/335

Moreno, V. (2022). *Seguridad de las Comunicaciones Inalámbricas en los.* Universidad Politécnica de Catalunya.

NIST. (2023). *Windows network driver privilege escalation.* U.S. Department of Commerce / NIS. https://msrc.microsoft.com/update-guide/releaseNote/2024-Jun

Ntantogian, C. M. (2021). An Empirical Assessment of Management Frame Vulnerabilities in Wi-Fi Networks. https://doi.org/10.1080/19393555.2021.1891997

Pastor, J. (22 de Julio de 2024). *El modo kernel, explicado: así funciona la seguridad de Windows y así se la "saltó" CrowdStrike para provocar el caos*. Xataka:

https://www.xataka.com/servicios/modo-kernel-explicado-asi-funciona-seguridad-windows-asi-se-salto-crowdstrike-para-provocar-caos

Ramos, D. (2021). *Análisis de vulnerabilidades a nivel de seguridad en redes sdn para los planos de control y plano de datos.* Universidad Militar Nueva Granada.

Redes Zona. (7 de Abril de 2025). *Qué es y cómo funciona la tecnología GPON: secretos técnicos*. Redes Zona: https://www.redeszone.net/tutoriales/redes-cable/tecnologia-ftth-gpon-que-es-funcionamiento/

Roth, A., Zeller, A., & Liang, C. (2021). Coverage-guided fuzzing for Windows binaries with WinAFL. *IEEE Transactions on Software Testing, 47*(6), 654–667.

Salazar, K. (2025). *Protocolos de autenticación y acuerdo de llaves para IoT: un examen práctico de su implementación, resiliencia y monitoreo en entornos contenerizados y federados.* Universidad de los Andes.

Schumilo, S., Töpper, M., & Gawlik, R. (2020). Fuzzing Windows Drivers. *Publicado por USENIX Association*. https://doi.org/https://www.usenix.org/conference/usenixsecurity20/presentation/peng

Singh, R., & Kumar, P. (2021). Operating system fundamentals and device driver design. *Elsevier., 1*(12), 40. https://doi.org/https://www.researchgate.net/publication/339413127_Operating_System_Fundamentals

Sun, Y., & Xu, W. (2021). A survey of wireless protocol vulnerabilities. *Sensors, 23*(4), 1849. https://doi.org/ https://doi.org/10.3390/s23041849

SysAdminok. (23 de junio de 2023). *Qué es y para qué sirve el Fuzzing: Un Enfoque Integral a la Seguridad de Software*. SysAdminok:

https://www.sysadminok.es/blog/ciberseguridad/que-es-y-para-que-sirve-el-fuzzing-un-enfoque-integral-a-la-seguridad-de-software/

Tao, S., Zhang, B., & Zhang, Q. (2025). ECHO: Mejora del fuzzing del kernel de Linux mediante la deduplicación de fallos con reconocimiento de pila de llamadas. *Electronics, 14*(14), 2914. https://doi.org/https://doi.org/10.3390/electronics14142914

Tariq, U., Ahmed, I., Bashir, A., & Shaukat, K. (2023). Análisis crítico de la ciberseguridad y futuras líneas de investigación para el Internet de las cosas: una revisión exhaustiva. *Sensors , 23*(8), 4117. https://doi.org/https://doi.org/10.3390/s23084117

Telconet. (15 de Octubre de 2025). *Vulnerabilidad de tipo Use-After-Free en Google Chrome permite ejecución remota de código*. Telconet: https://csirt.telconet.net/comunicacion/boletines-servicios/vulnerabilidad-de-tipo-use-after-free-en-google-chrome-permite-ejecucion-remota-de-codigo/

Thankappan, M., Rifà-Pous, H., & Garrigues, C. (2022). Ataques Man-in-the-Middle multicanal contra redes Wi-Fi protegidas: una revisión del estado del arte. *Sistemas expertos con aplicaciones, 210*. https://doi.org/https://doi.org/10.1016/j.eswa.2022.118401

TI Rescue. (28 de Octubre de 2024). *Fuzzing: Técnica Esencial para Detectar Vulnerabilidades* . TI Rescue: https://tirescue.com/fuzzing-tecnica-esencial-para-detectar-vulnerabilidades/

Tripathy, J., Mishra, A., Pandey, M., Thakur, R., Chand, S., Rout, P., & Shahid, M. (2024). Avances en nanopartículas y nanocompuestos para el tratamiento de agua y aguas residuales: Una revisión. *Water, 16*(11), 1481. https://doi.org/https://doi.org/10.3390/w16111481

Vinopal, J. (2024). *Investigating Vulnerable Drivers and Mitigating Risks.* Check Point Research. https://research.checkpoint.com/2024/breaking-boundaries-investigating-vulnerable-drivers-and-mitigating-risks/

Vukšić, M., Ćelić, J., Panić, I., & Cuculić, A. (2025). Explotación de Wi-Fi marítimo: Evaluación práctica de vulnerabilidades de la red a bordo. *Revista de Ciencias e Ingeniería Marinas, 13*(8), 1576. https://doi.org/https://doi.org/10.3390/jmse13081576

Wang, S., Liu, X., & & Han, Q. (2022). Network driver interface security challenges in modern Windows systems. *Electrónica, 10*, 47651–47665. https://doi.org/https://doi.org/10.1016/j.jocs.2023.112129

Wu, B., Chen, Z., & Zhang, L. (2021). Security analysis of wireless drivers in modern operating systems. *Computers & Security, 23*(22), 9221. https://doi.org/ https://doi.org/10.3390/s23229221

Yaacoub, J., Noura, H., Salman, O., & Chahine, K. (2025). Hacia sistemas de redes inteligentes seguras: Riesgos, amenazas, desafíos y futuras direcciones. . *Future Internet , 17*(7), 318. https://doi.org/https://doi.org/10.3390/fi17070318

Zeller, A., Klees, G., & Payer, M. (2022). Foundations and advances. *IEEE Security & Privacy, 20*(5), 24–33.

Zhao, Y., Feng, W., & Li, T. (2022). Empirical study of memory-related vulnerabilities in kernel software systems. *IEEE Security & Privacy, 20*(5), 24-33. https://doi.org/https://www.researchgate.net/publication/315868569_An_Empirical_Analysis_of_Vulnerabilities_in_OpenSSL_and_the_Linux_Kernel

Zhao, Y., Feng, W., & Li, T. (2022). vulnerabilities in kernel software systems. *IEEE Security & Privacy, 20*(5), 24–33.

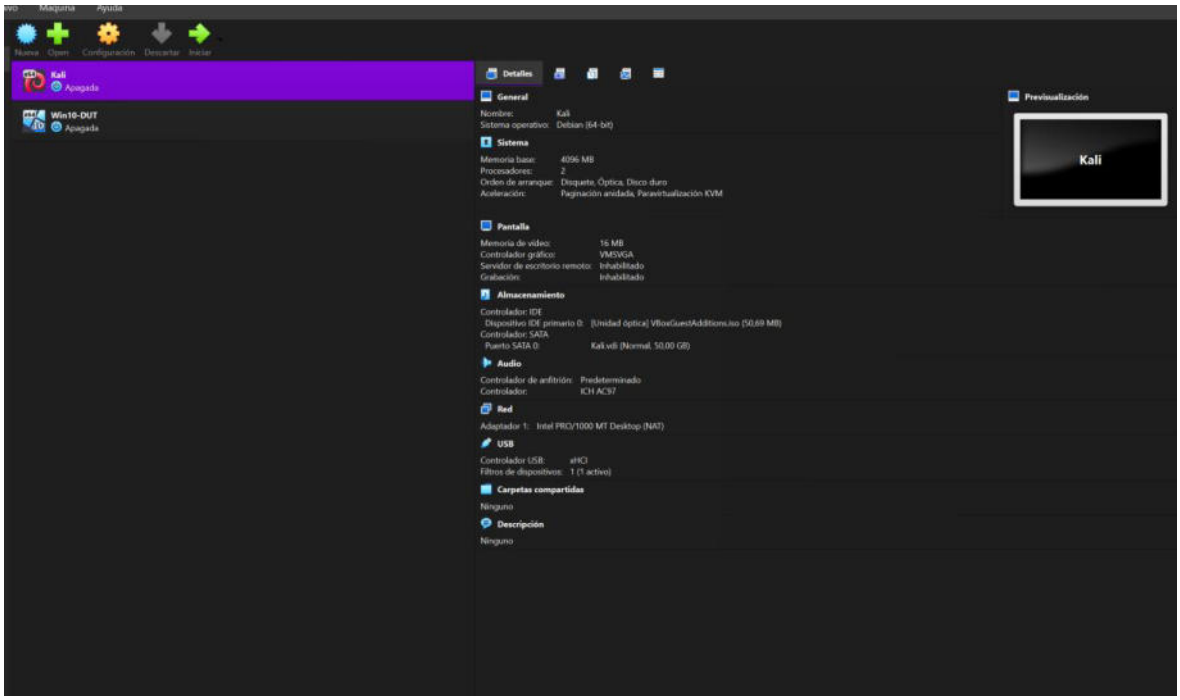
https://doi.org/https://www.researchgate.net/publication/337795385_Kernel_Vulnerability_Analysis_A_Survey

ANEXOS

Laboratorio de Fuzzing Wi-Fi con Kali + Windows 10 (DUT)

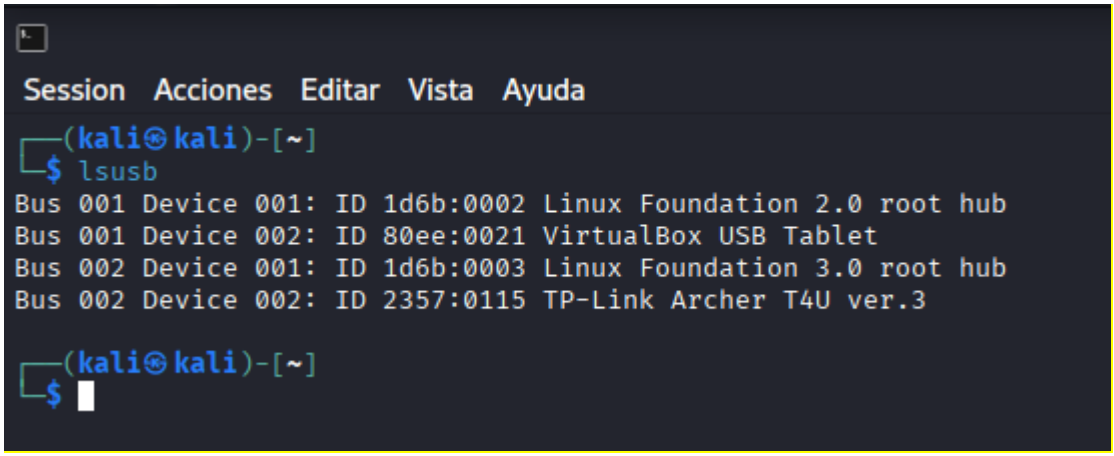
Anexo 1. *Configuración de la máquina virtual Kali Linux en VirtualBox*

Se muestra la configuración de la VM con el sistema Kali Linux, asignación de CPU, RAM, disco y adaptador USB Wi-Fi necesario para el modo monitor.

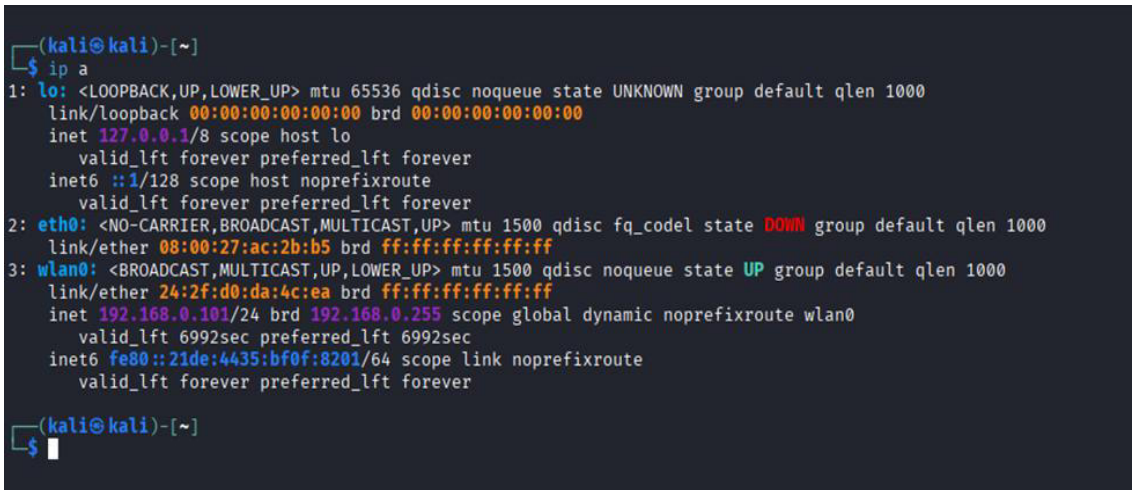


1. VirtualBox → Nuevo → Nombre: “Kali”; Tipo: Linux; Versión: Debian (64-bit).
2. Memoria: 4096 MB; CPU: 2 vCPU.
3. Disco: 30–40 GB (VDI/VHD dinámico).
4. Almacenamiento: adjunta ISO de Kali. USB: selecciona USB 3.0 (xHCI).
5. Conecta el USB-2 al host y agrega su filtro en Configuración → USB → +.
6. Instalar Kali (usuario/clave, tz, etc.).
7. En Kali, verificar y preparar:

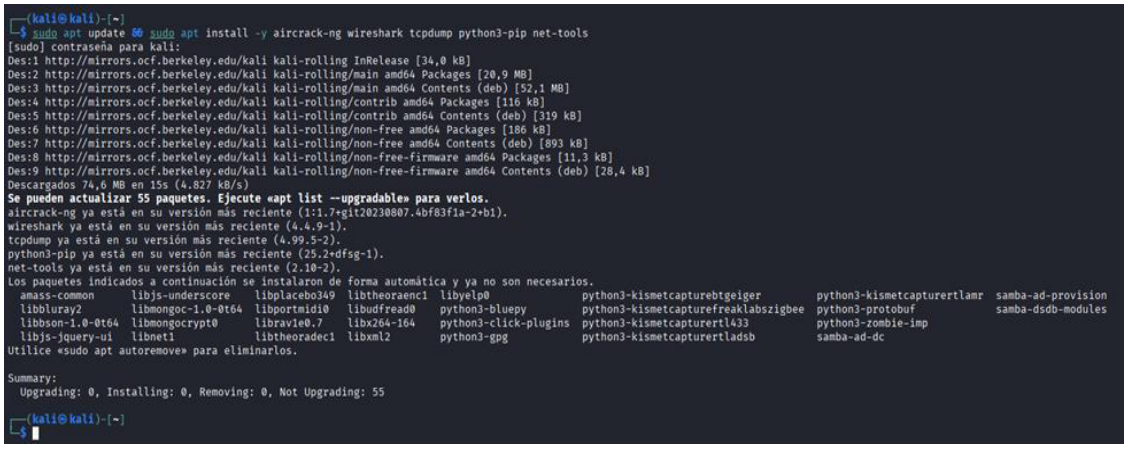
- lsusb → debería verse el USB-2



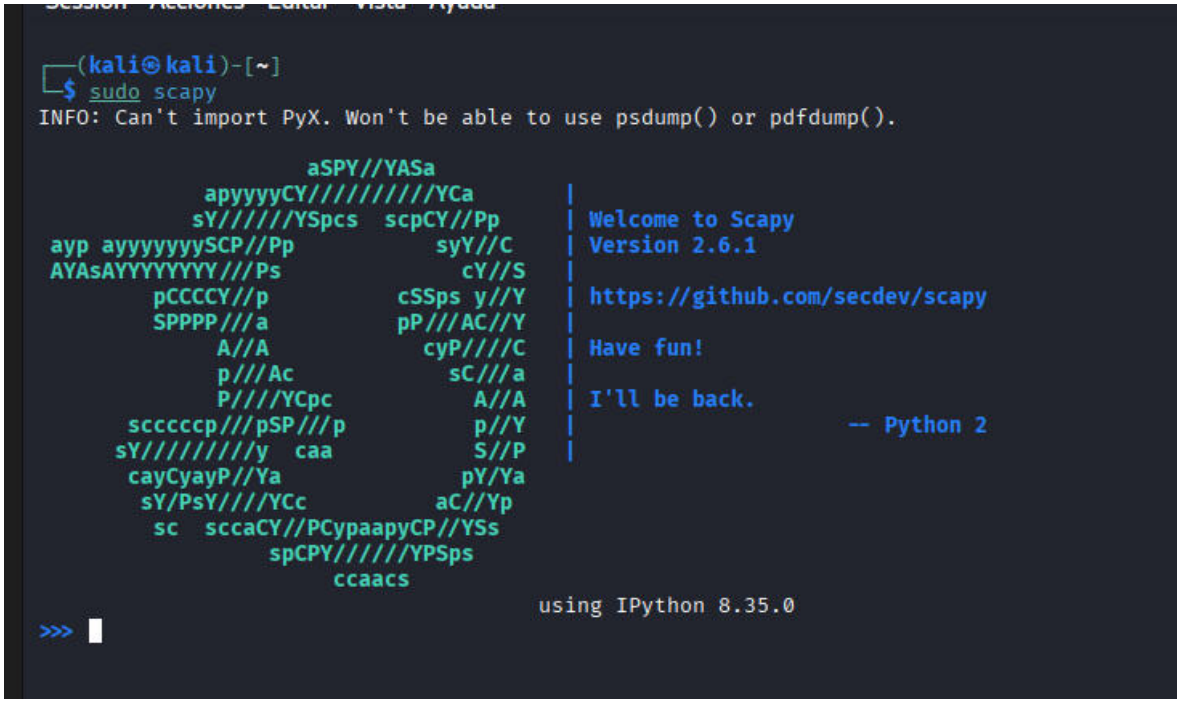
- ip a → interfaz wlan0 (u otra)



- sudo apt update && sudo apt install -y aircrack-ng wireshark tcpdump python3-pip net-tools

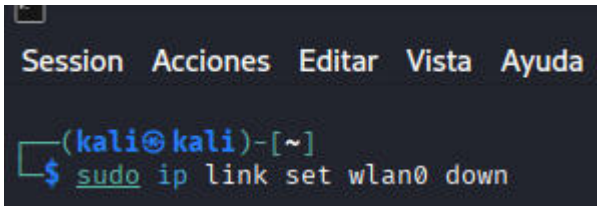


- sudo apt update |sudo apt install -y python3-scapy | python3 -c "import scapy, sys; print('scapy', scapy.__version__, 'python', sys.version.split()[0])" | sudo scapy

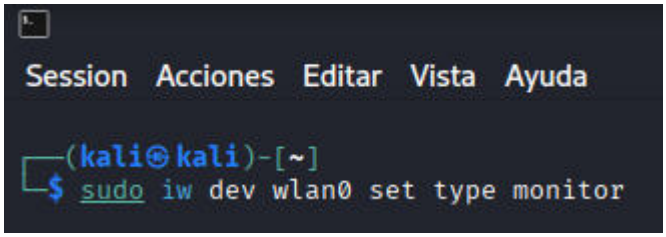


Modo monitor e inyección (prueba rápida):

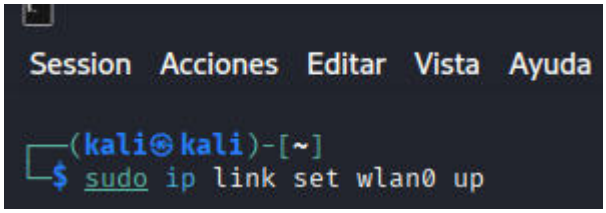
sudo ip link set wlan0 down



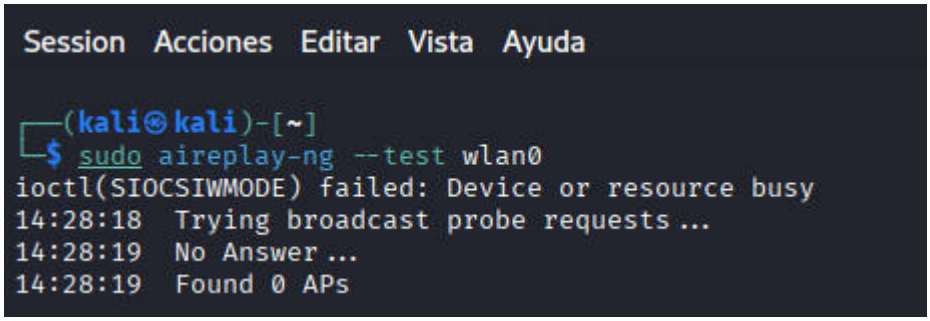
sudo iw dev wlan0 set type monitor



sudo ip link set wlan0 up

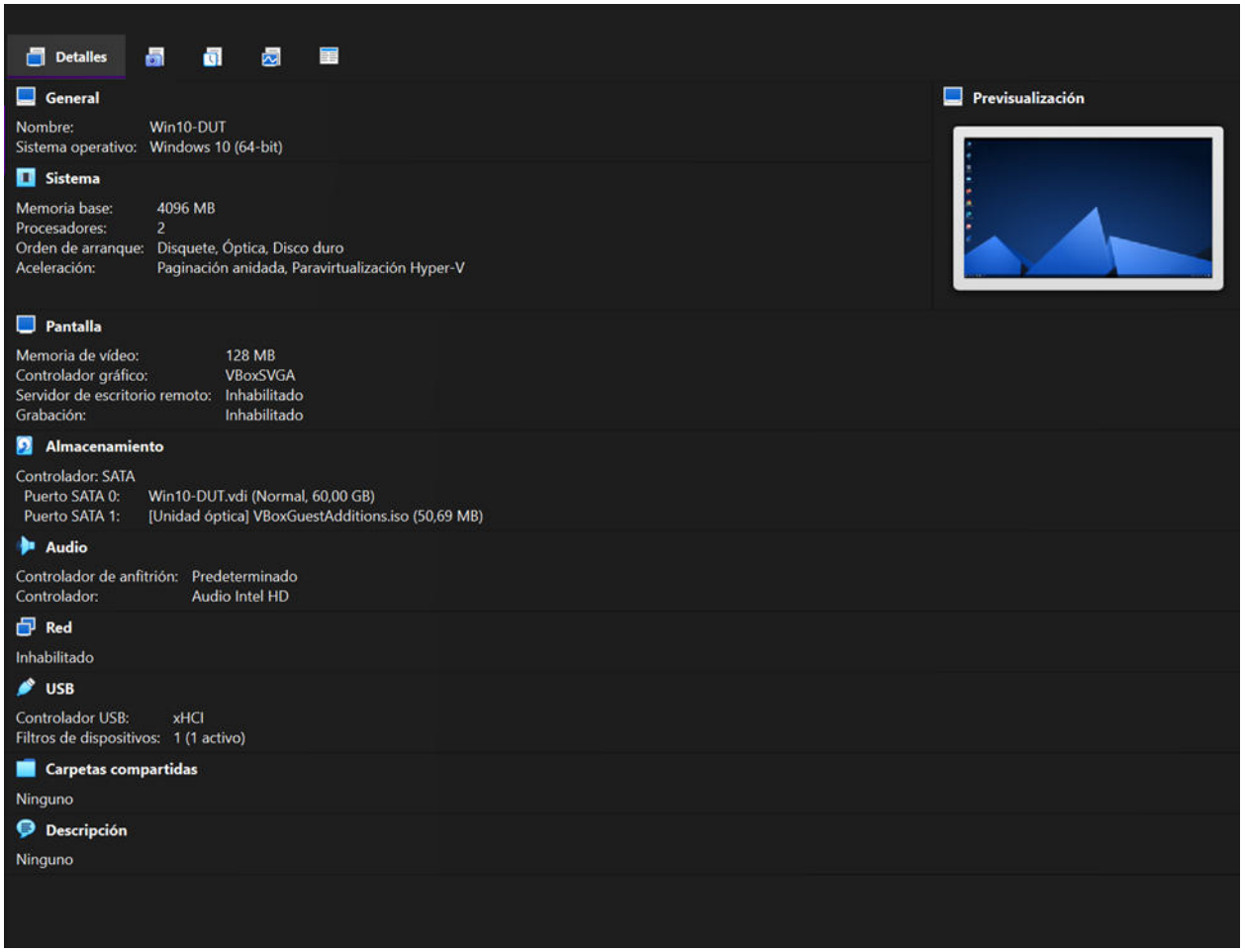


sudo aireplay-ng --test wlan0



Anexo 2. *Configuración de la máquina virtual Windows 10 (DUT)*

Se muestra la configuración del equipo bajo prueba (DUT) en VirtualBox, incluyendo memoria asignada, procesadores, disco virtual y controlador USB utilizado para el adaptador Wi-Fi.



Anexo 3. *Estado inicial de la interfaz Wi-Fi del dispositivo bajo prueba (DUT)*

Salida del comando netsh wlan show interfaces en Windows-DUT, donde se observa la conexión inalámbrica estable previa al inicio de las pruebas. Se muestran parámetros clave como SSID, canal, velocidad, tipo de red, autenticación y calidad de señal, los cuales constituyen la línea base del experimento.

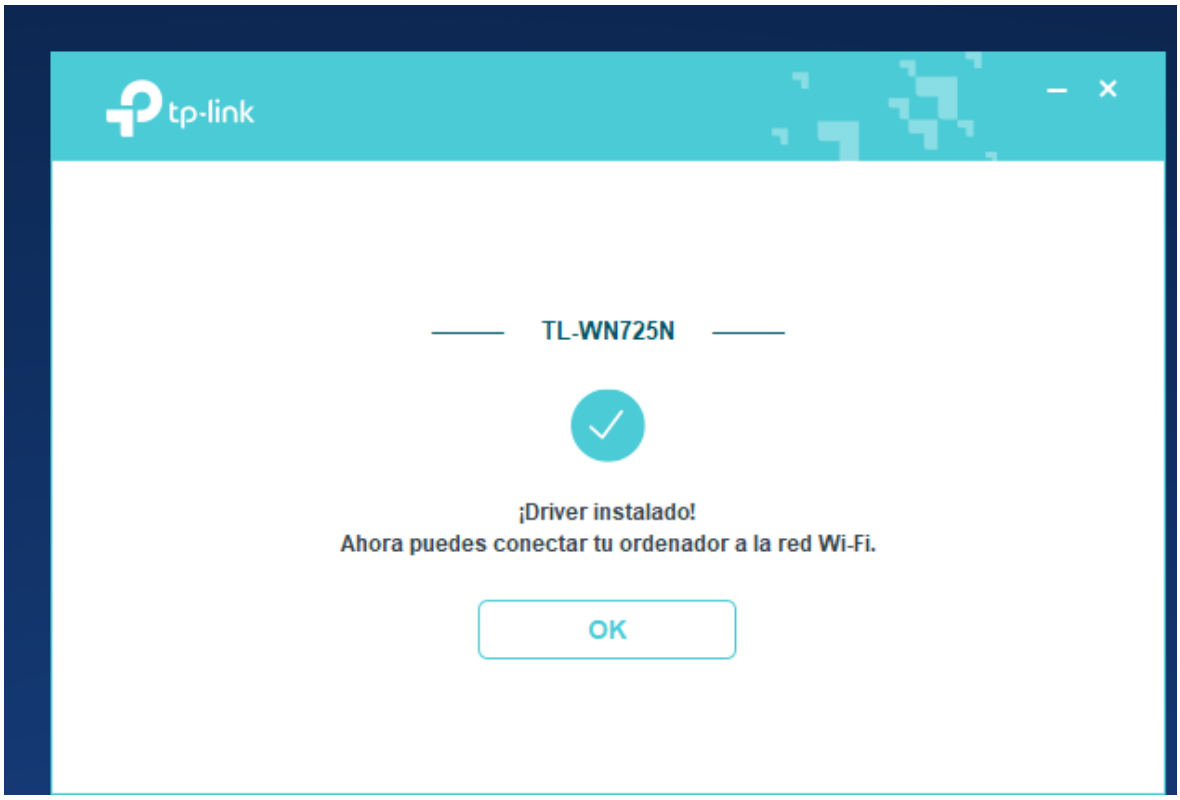
1. Nuevo → Nombre: “Win10-DUT”; Tipo: Windows 10 (64-bit).

2. Memoria: 4096 MB; CPU: 2 vCPU; Disco: 40–60 GB.

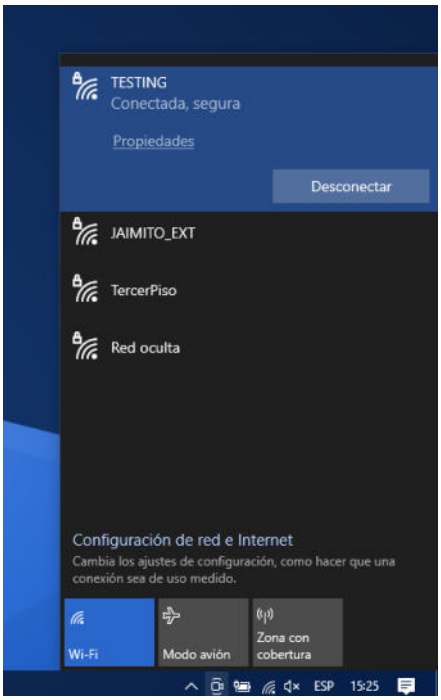
3. Almacenamiento: ISO de Windows 10. USB: 3.0 (xHCI).

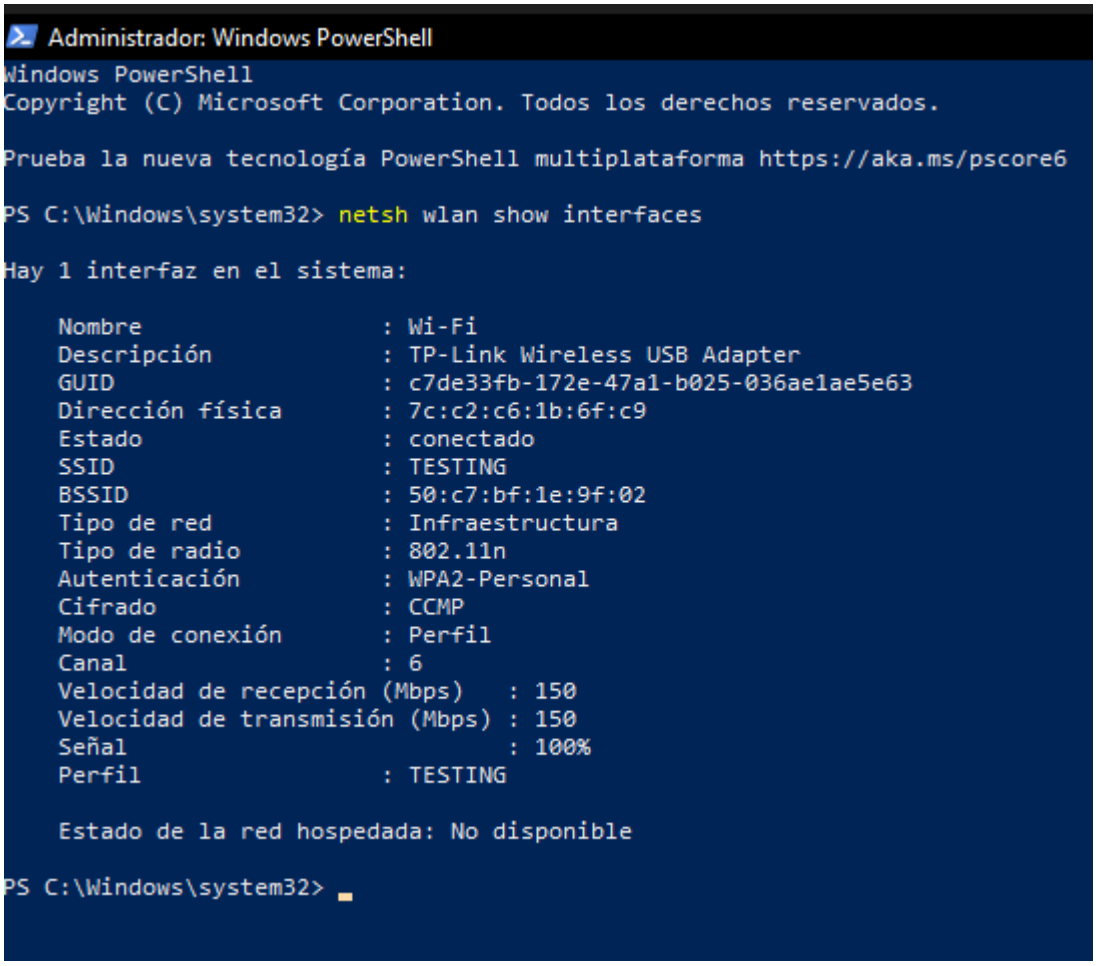
4. Conectar el USB-1 (el DUT) y agregar filtro USB en la VM Win10.

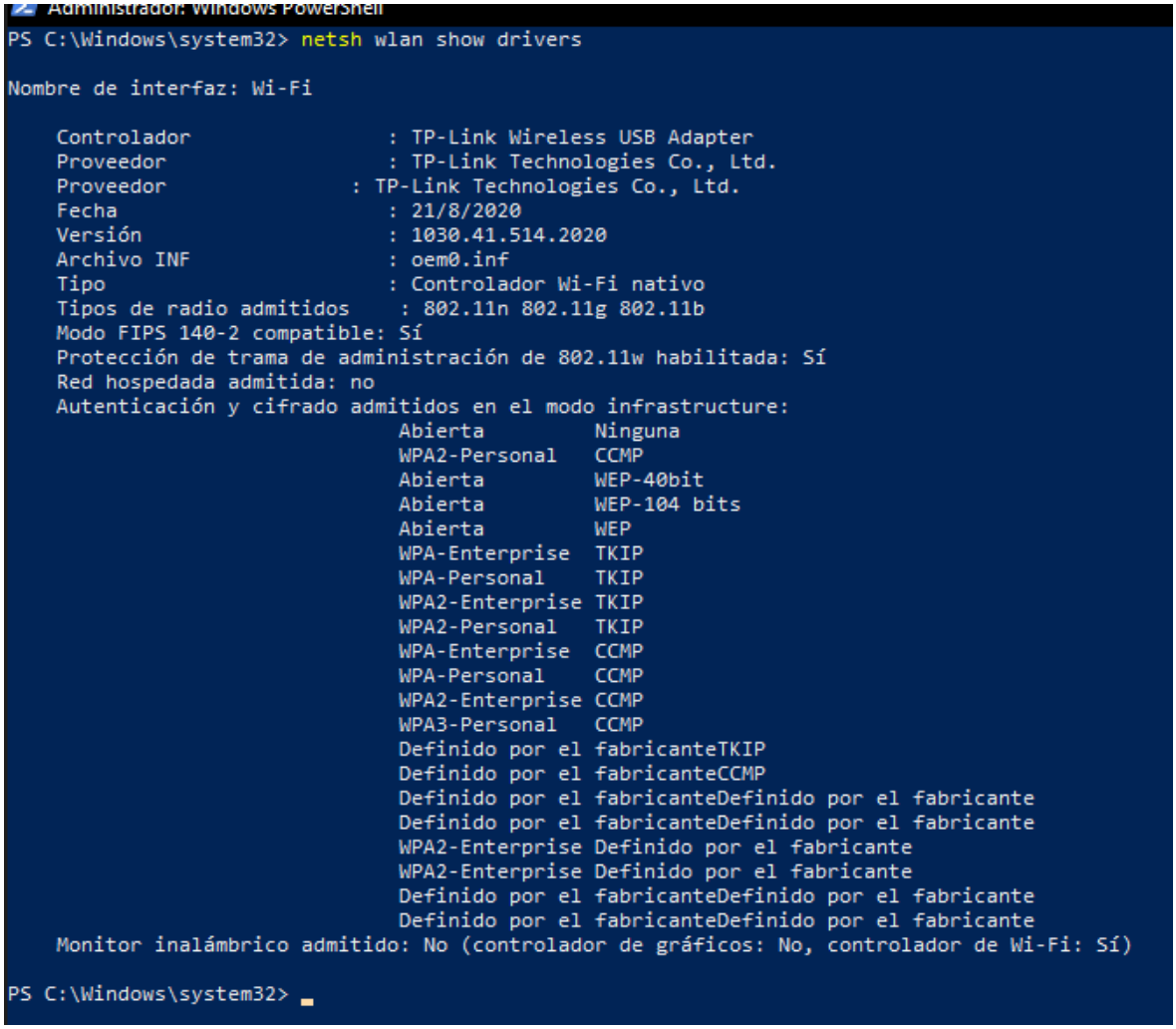
5. Instalar driver oficial del adaptador Wi-Fi dentro de la VM.



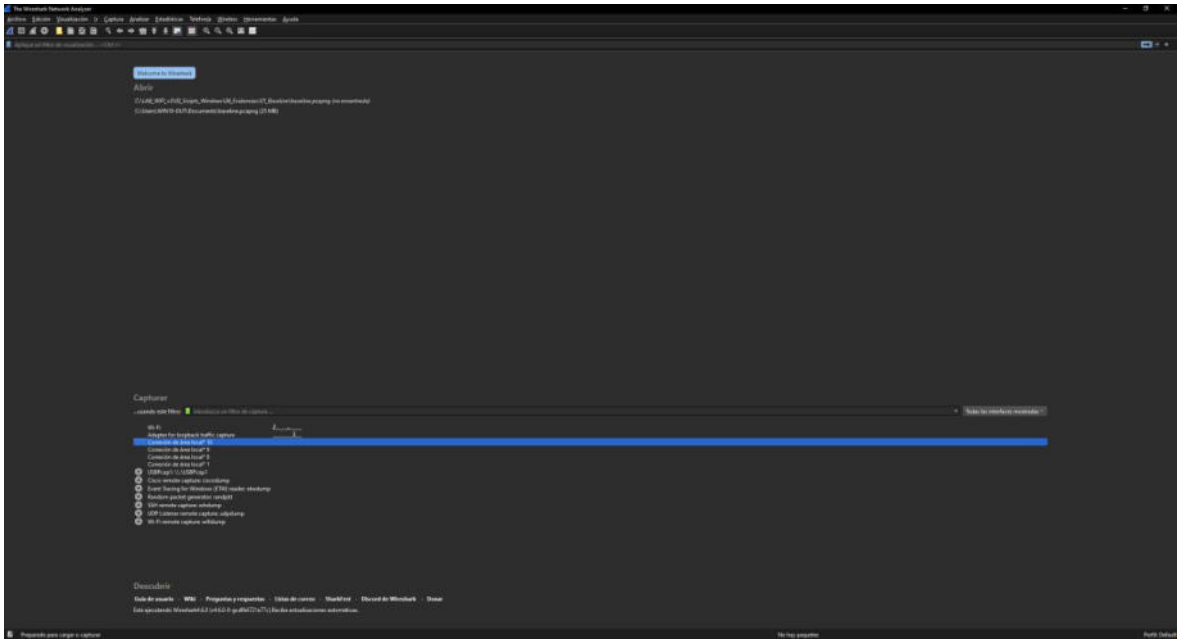
6. Conectar el DUT al router (el SSID de laboratorio).




Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6
PS C:\Windows\system32> netsh wlan show interfaces
Hay 1 interfaz en el sistema:
Nombre : Wi-Fi
Descripción : TP-Link Wireless USB Adapter
GUID : c7de33fb-172e-47a1-b025-036ae1ae5e63
Dirección física : 7c:c2:c6:1b:6f:c9
Estado : conectado
SSID : TESTING
BSSID : 50:c7:bf:1e:9f:02
Tipo de red : Infraestructura
Tipo de radio : 802.11n
Autenticación : WPA2-Personal
Cifrado : CCMP
Modo de conexión : Perfil
Canal : 6
Velocidad de recepción (Mbps) : 150
Velocidad de transmisión (Mbps) : 150
Señal : 100%
Perfil : TESTING
Estado de la red hospedada: No disponible
PS C:\Windows\system32>

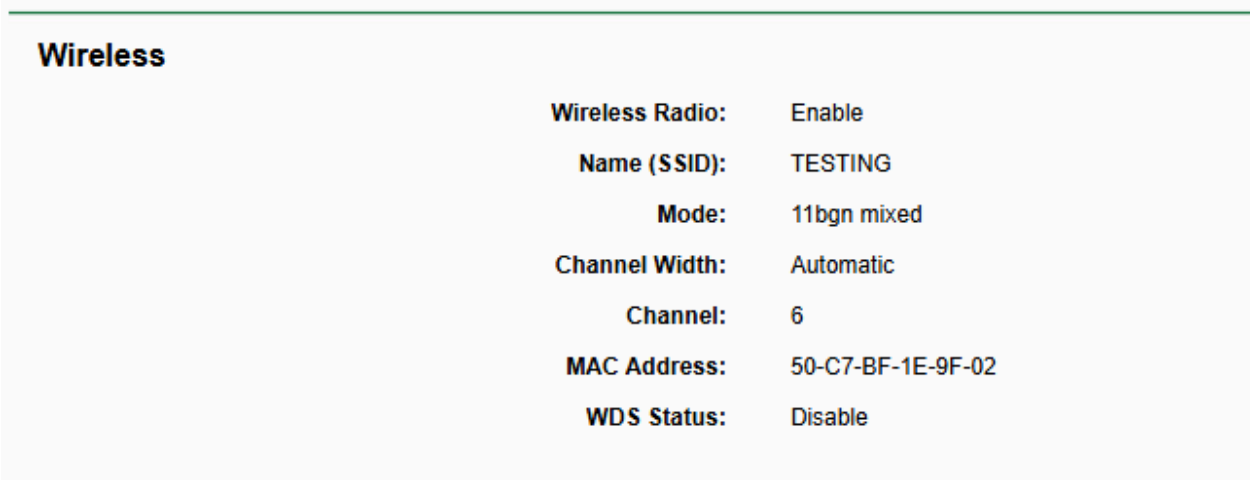


Instalar Wireshark (marca Npcap).



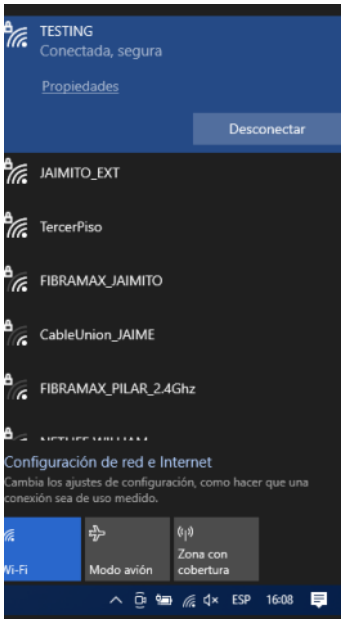
Poner el Router en canal fijo

1. Entrar al panel del router.
2. Fijar un canal específico (p. ej., 6 en 2.4 GHz o 36/40/44/48 en 5 GHz).
3. Tomar nota del SSID, BSSID y canal.

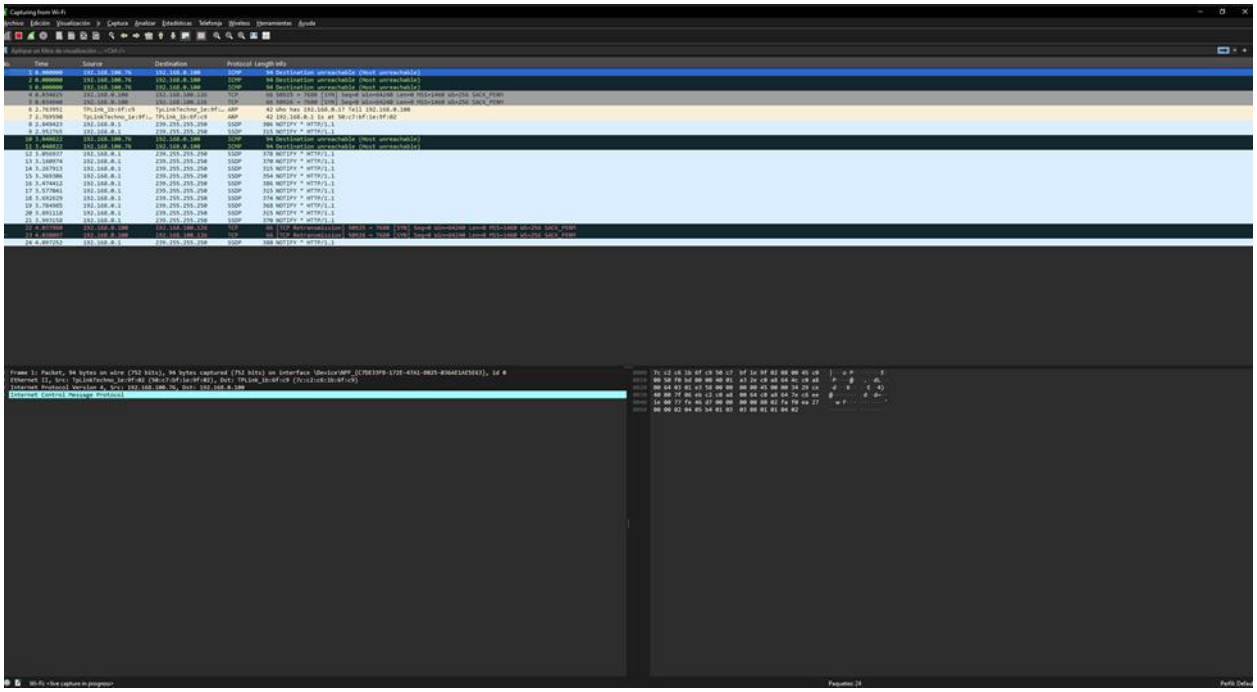


Anexo 4. *Captura de tráfico normal en Wireshark durante la línea base*

Wireshark registra paquetes ICMP, DNS y tráfico general correspondientes al funcionamiento normal de la interfaz Wi-Fi del DUT antes de aplicar las pruebas de fuzzing. Esta captura sirve como referencia para comparar el comportamiento del sistema frente a los ataques ejecutados posteriormente.

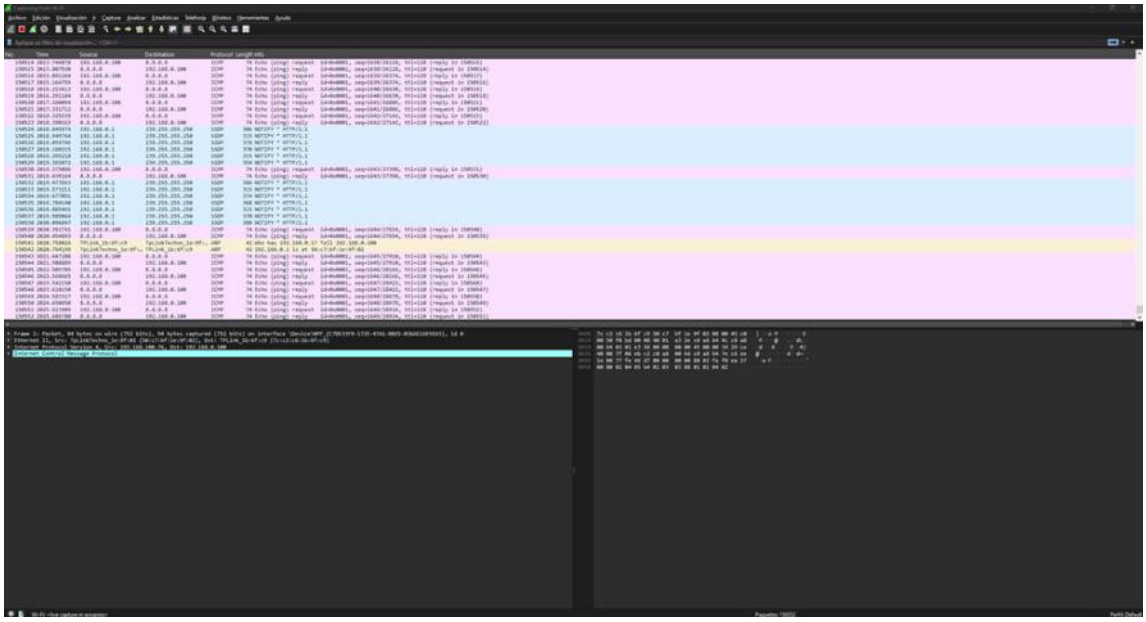


Abrir Wireshark y empezar a capturar en la interfaz Wi-Fi.



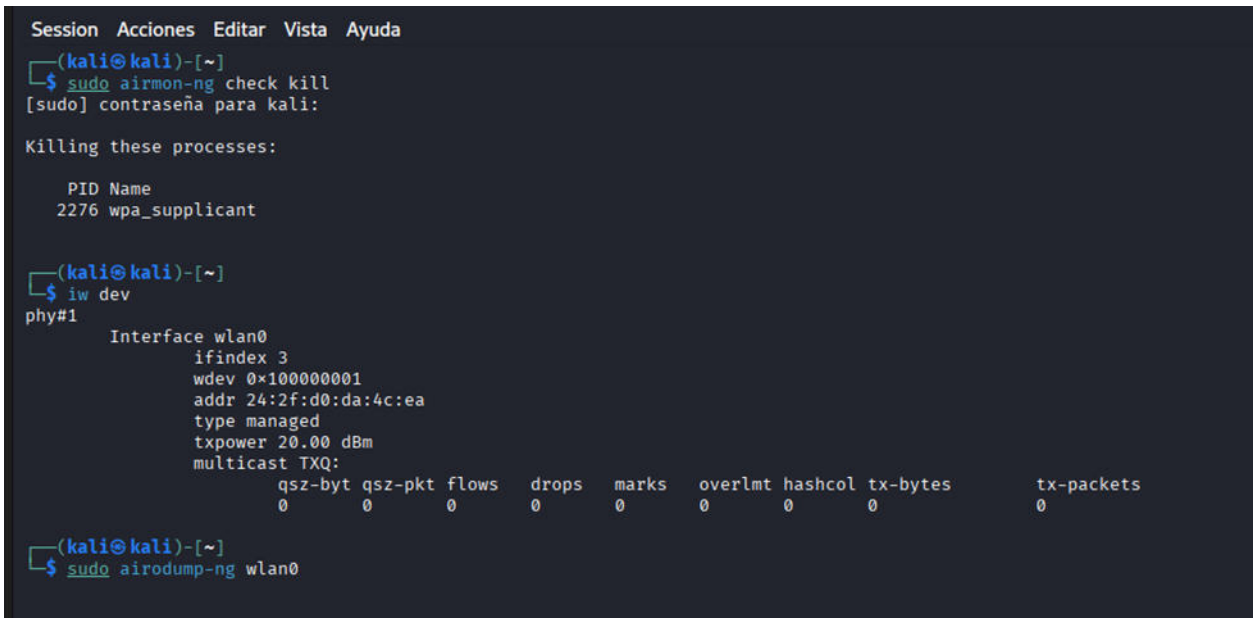
Anexo 5. *Captura de tráfico en Wireshark previa a las pruebas (estado normal del DUT)*

La captura muestra a Wireshark registrando tráfico inalámbrico del DUT en condiciones normales, antes de la ejecución de las pruebas de desautenticación y fuzzing. Esta trazabilidad sirve como referencia para comparar el comportamiento posterior del controlador bajo ataque.



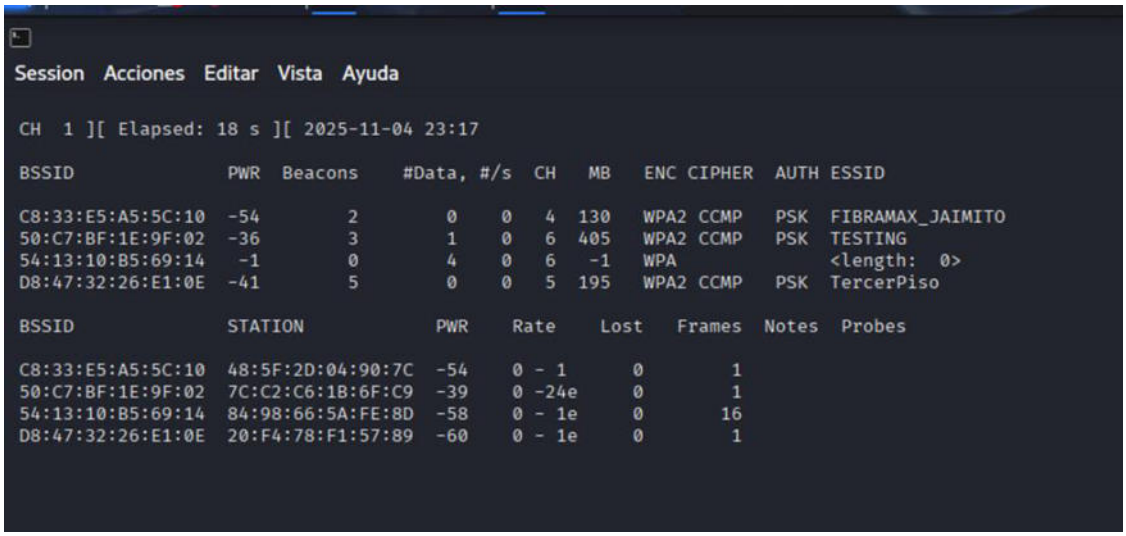
Anexo 6. *Configuración de la interfaz inalámbrica en modo monitor en Kali Linux*

En la terminal de Kali se observa la preparación de la interfaz wlan0 para modo monitor, desactivando procesos previos y verificando los parámetros del dispositivo. Este paso es necesario para poder inyectar y capturar tramas 802.11 en la prueba DEA-01.



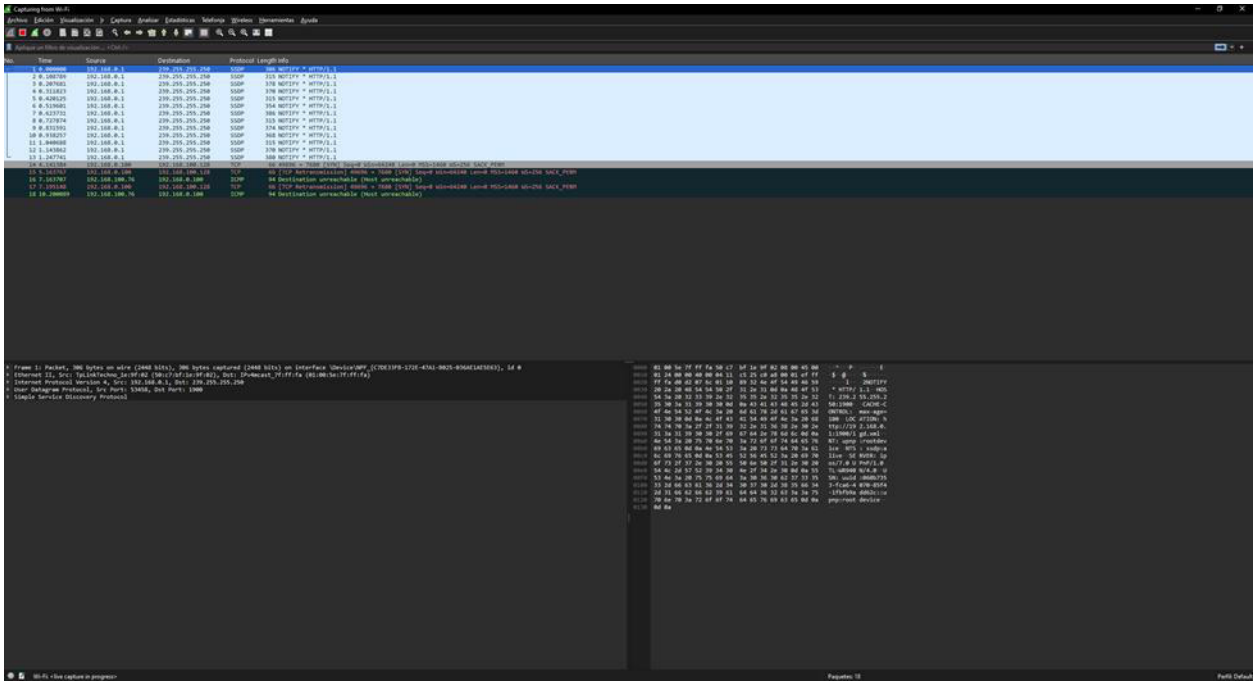
Anexo 7. *Identificación del AP objetivo y del canal mediante airodump-ng*

La herramienta airodump-ng lista las redes inalámbricas cercanas, mostrando el BSSID, el canal y la intensidad de señal del AP de laboratorio. Esta información se utiliza para dirigir correctamente las tramas de desautenticación hacia el DUT.



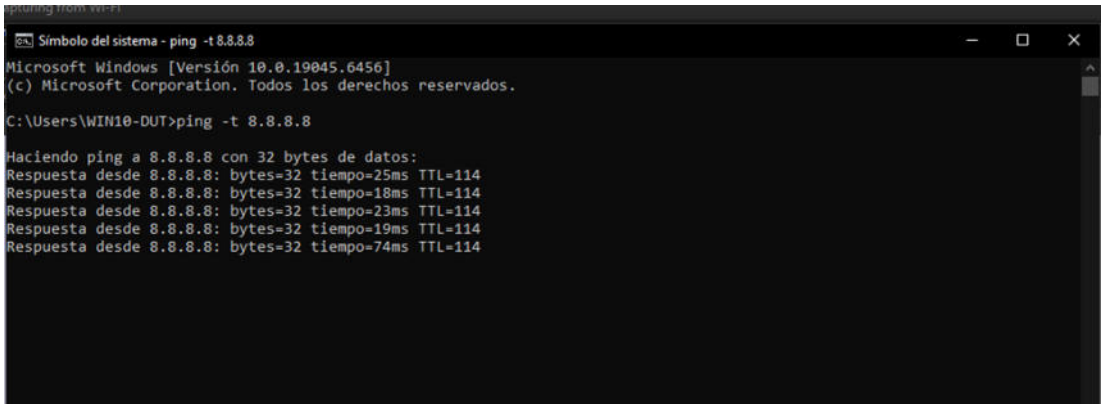
Anexo 8. *Tramas de desautenticación capturadas en Wireshark durante la prueba DEA-01*

Wireshark registra la aparición de tramas deauth enviadas desde Kali hacia el AP del DUT. Estas tramas provocan la desconexión breve del dispositivo, constituyendo la base del escenario de ataque DEA-01.



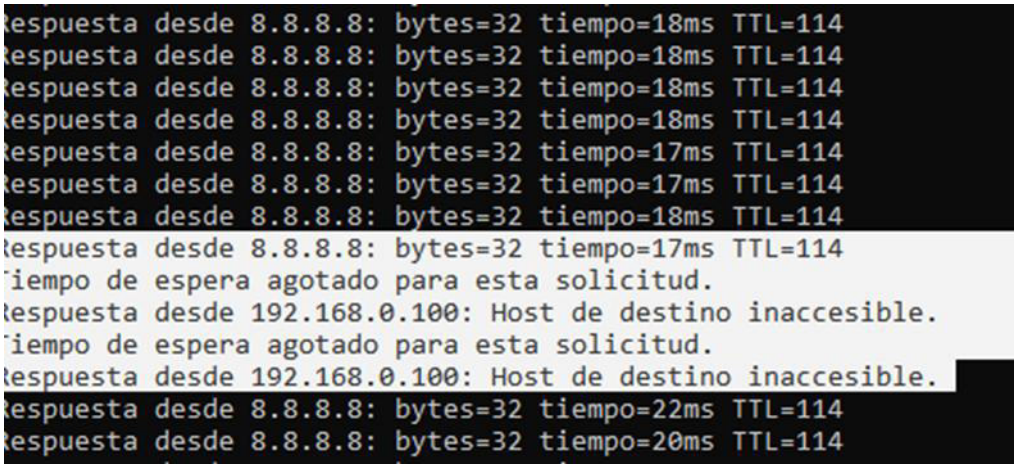
Anexo 9. *Interrupción temporal del ping durante el ataque de desautenticación*

La consola de Windows muestra la ejecución de un ping continuo hacia Internet. Durante la prueba DEA-01 se observan respuestas normales seguidas de una breve interrupción, evidenciando el impacto del ataque deauth sobre la conectividad del DUT.



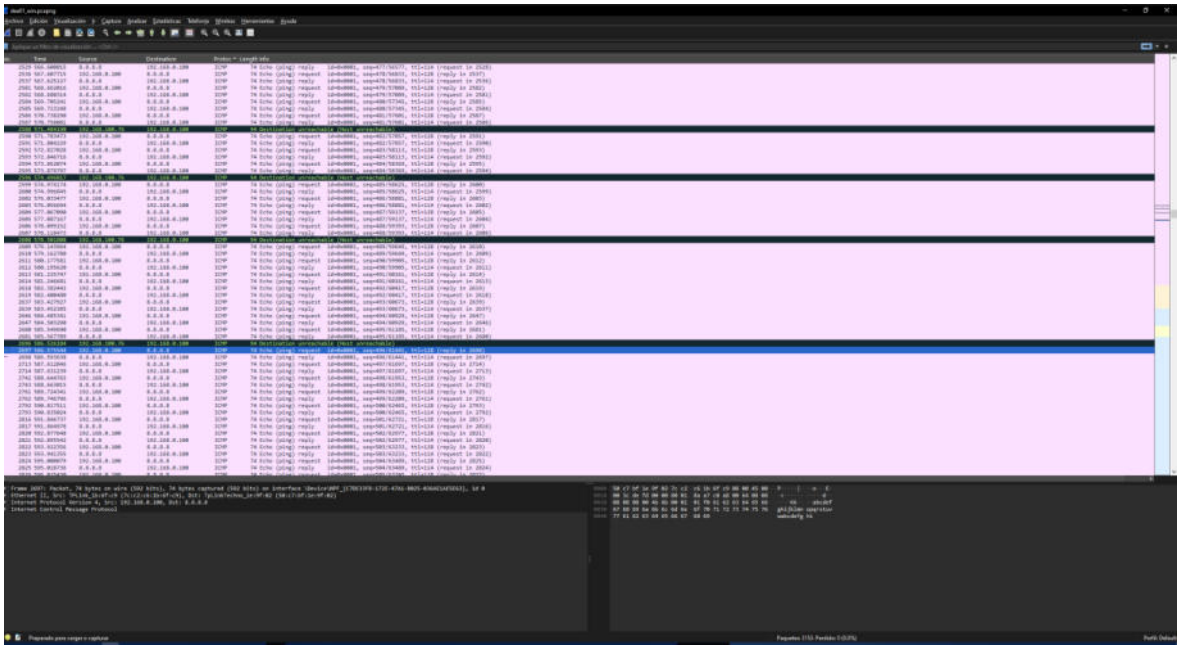
Anexo 10. *Pérdida y recuperación de conectividad ICMP tras el ataque DEA-01*

En la salida del comando ping se aprecian varios mensajes de “tiempo de espera agotado” seguidos de respuestas exitosas. Este patrón refleja la ventana de desconexión y el tiempo de recuperación del DUT después de recibir tramas de desautenticación.



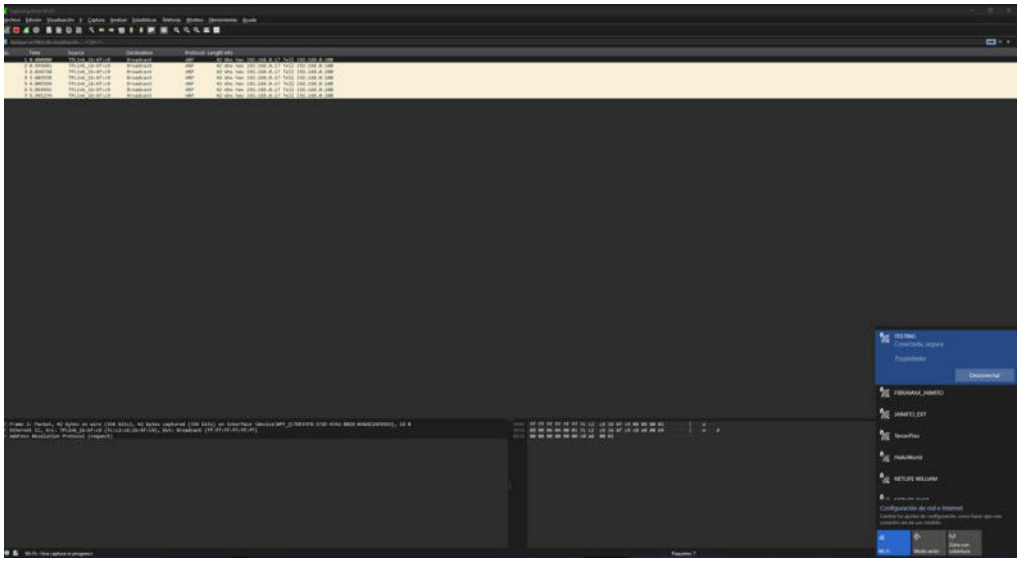
Anexo 11. *Secuencia de reconexión del DUT capturada en Wireshark*

Wireshark muestra el intercambio de tramas de gestión y datos (incluyendo EAPOL y tráfico IP) asociado al proceso de reconexión del DUT una vez finalizado el ataque. Esta evidencia complementa el comportamiento observado en el ping.



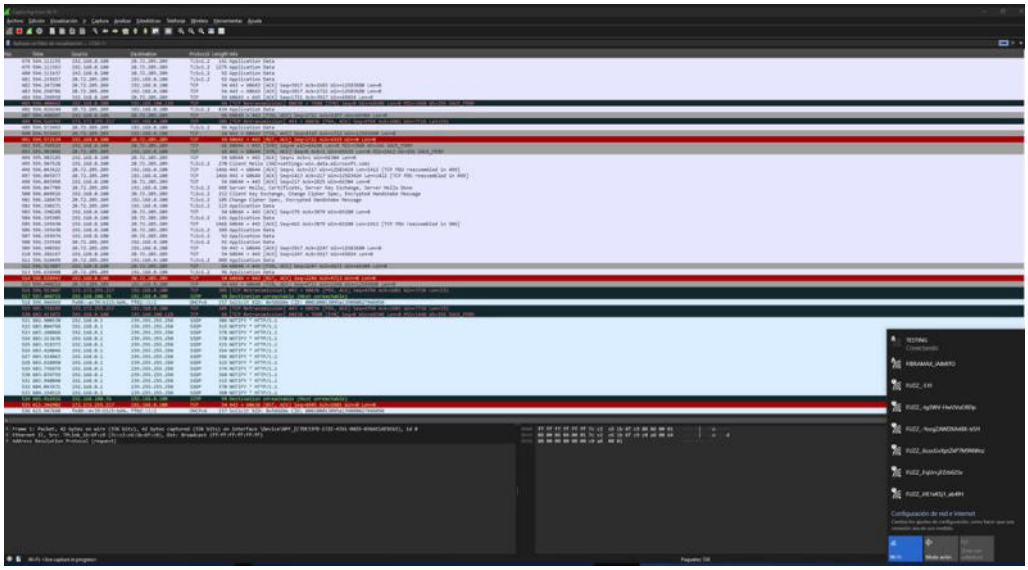
Anexo 12. *Ejecución del script de fuzzing de beacons y captura del tráfico resultante*

Se observa el entorno de Kali capturando tramas 802.11 mientras se ejecuta el script de fuzzing de beacons. La imagen evidencia el envío masivo de beacons alterados hacia el entorno del DUT, componente clave de la prueba BEA-01.



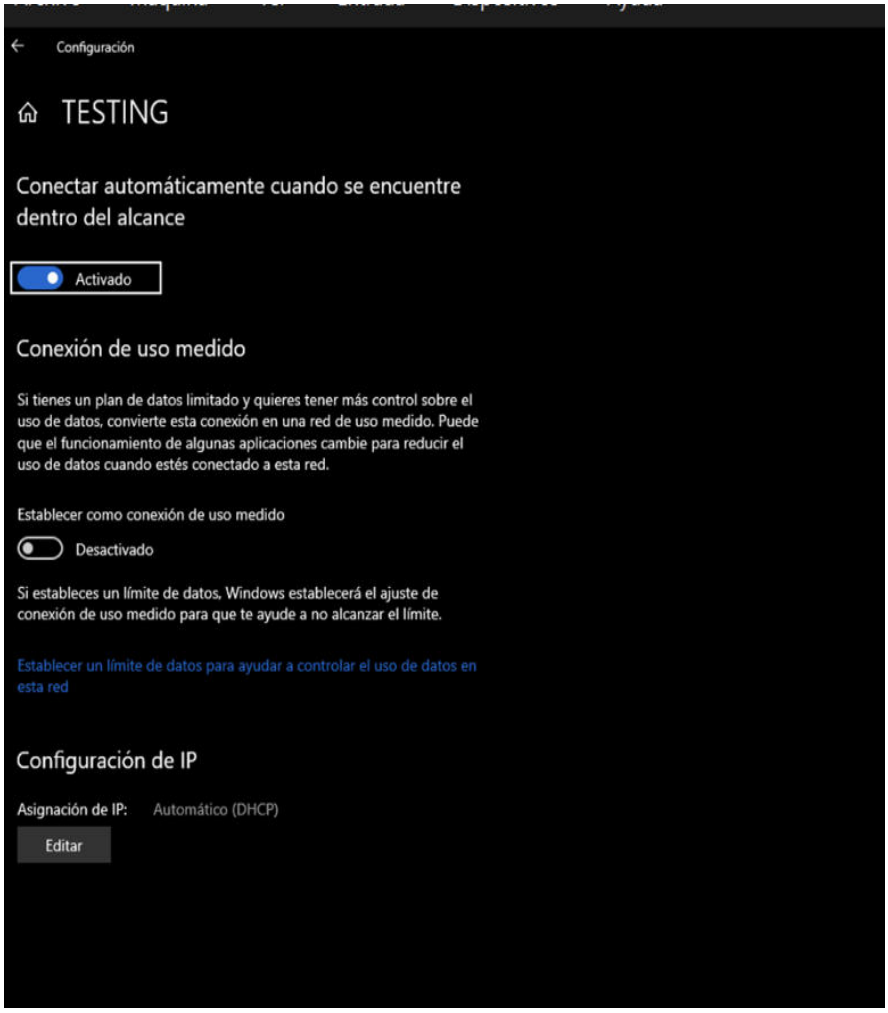
Anexo 13. *Tráfico anómalo de beacons durante la prueba BEA-01*

Wireshark registra múltiples beacons y otras tramas 802.11 generadas durante el fuzzing. La densidad y variación del tráfico permiten visualizar cómo se estresa el entorno inalámbrico del DUT ante beacons malformados.



Anexo 14. *Captura del estado del Wi-Fi en Windows (Propiedades de red)*

Captura que muestra el estado actual de la conexión Wi-Fi del DUT en Windows, incluyendo SSID, tipo de seguridad, configuración IP y parámetros del enlace.



Anexo 15. *PowerShell mostrando parámetros extraídos de netsh wlan show interfaces*

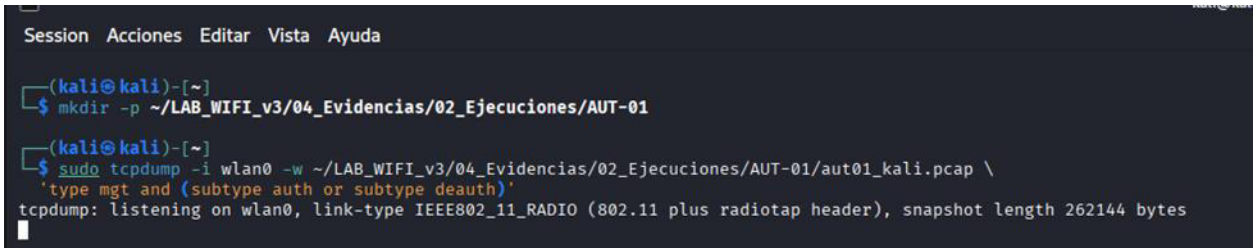
Salida de PowerShell donde se extraen valores clave del adaptador Wi-Fi mediante comandos de filtrado (Select-String), verificando estado y SSID conectados.

Anexo 16. *Captura 802.11 en Kali*

Preparación del entorno en Kali Linux para habilitar el modo monitor en la interfaz inalámbrica y liberar procesos que interfieren.

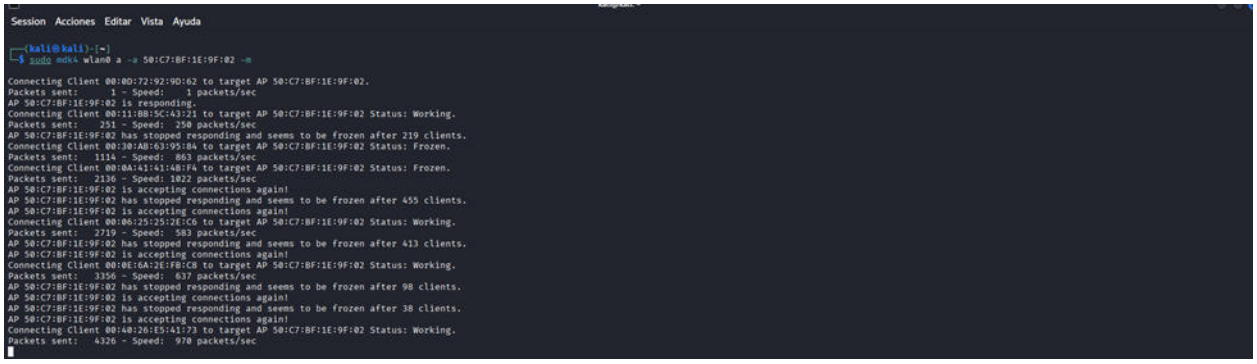
Anexo 17. *Captura 802.11 iniciada con tcpdump*

Ejecución de tcpdump capturando tráfico 802.11 crudo mientras se realiza la prueba AUT-01. Permite observar tramas de autenticación generadas por el ataque.



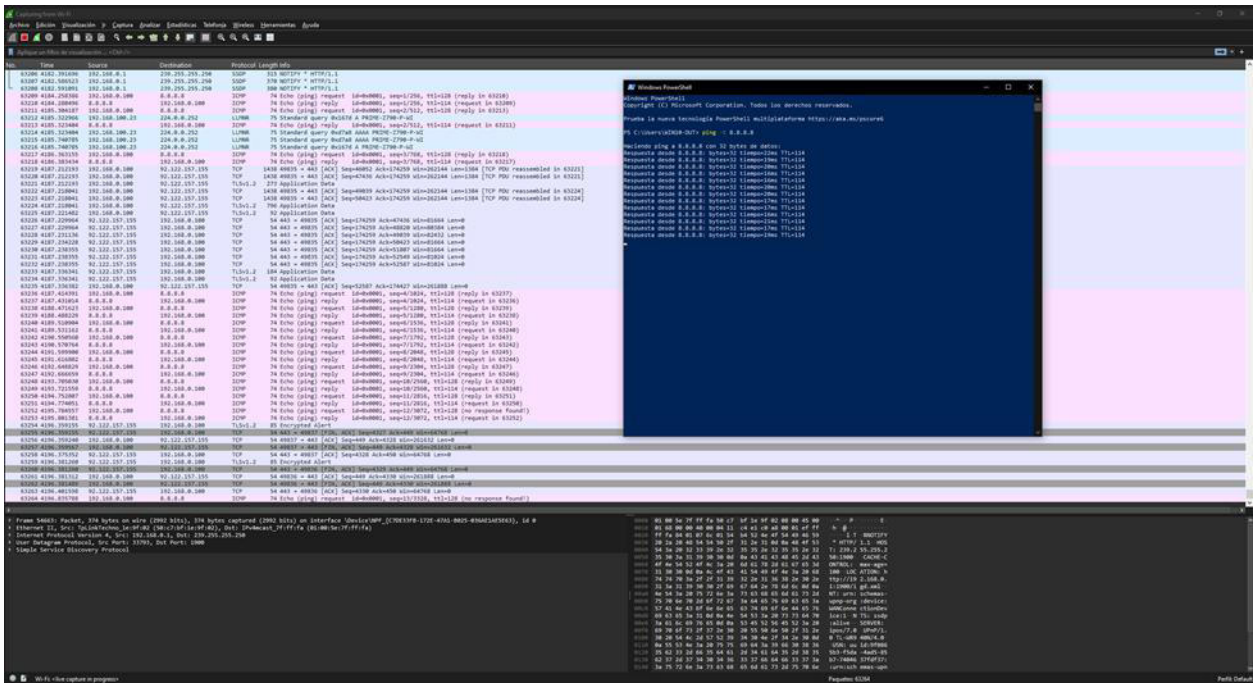
Anexo 18. *Ataque de autenticación con mdk4*

Ejecución del ataque mdk4 enviando múltiples tramas de autenticación hacia el AP. Esta captura muestra el inicio del flood antes de detenerse manualmente.



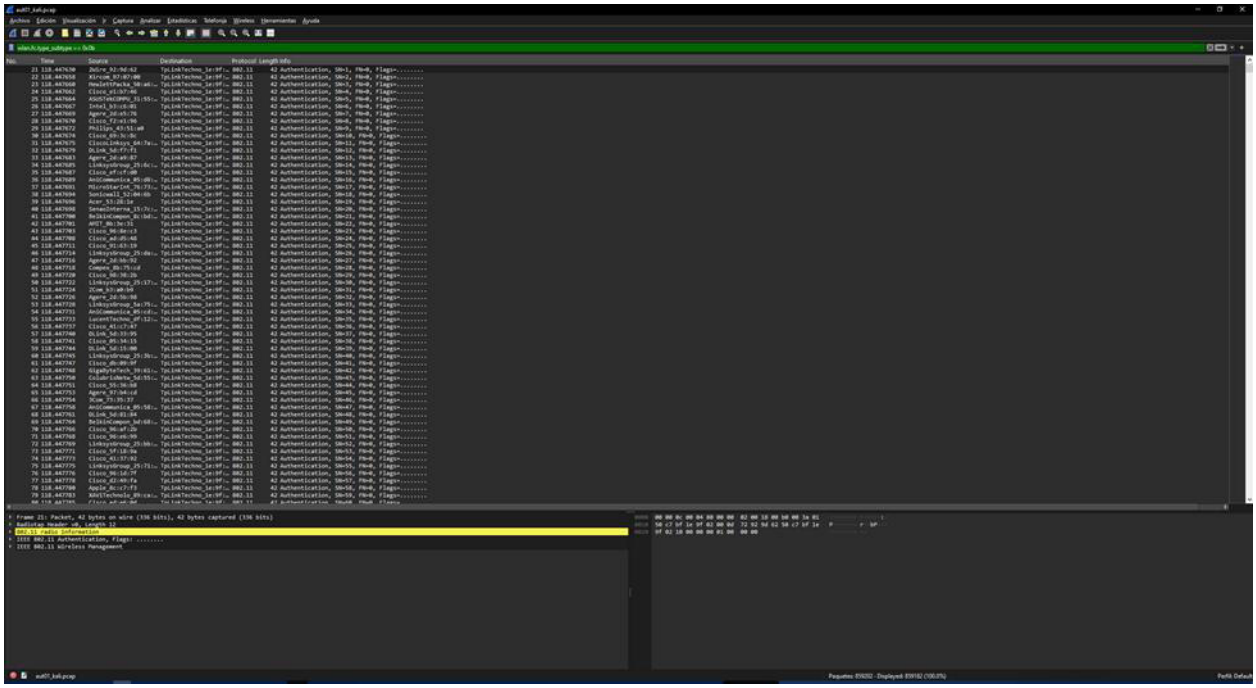
Anexo 19. *Wireshark (Windows) capturando + PowerShell haciendo ping*

Captura paralela en Windows mostrando tráfico ICMP mientras Wireshark registra el comportamiento del DUT durante el ataque.



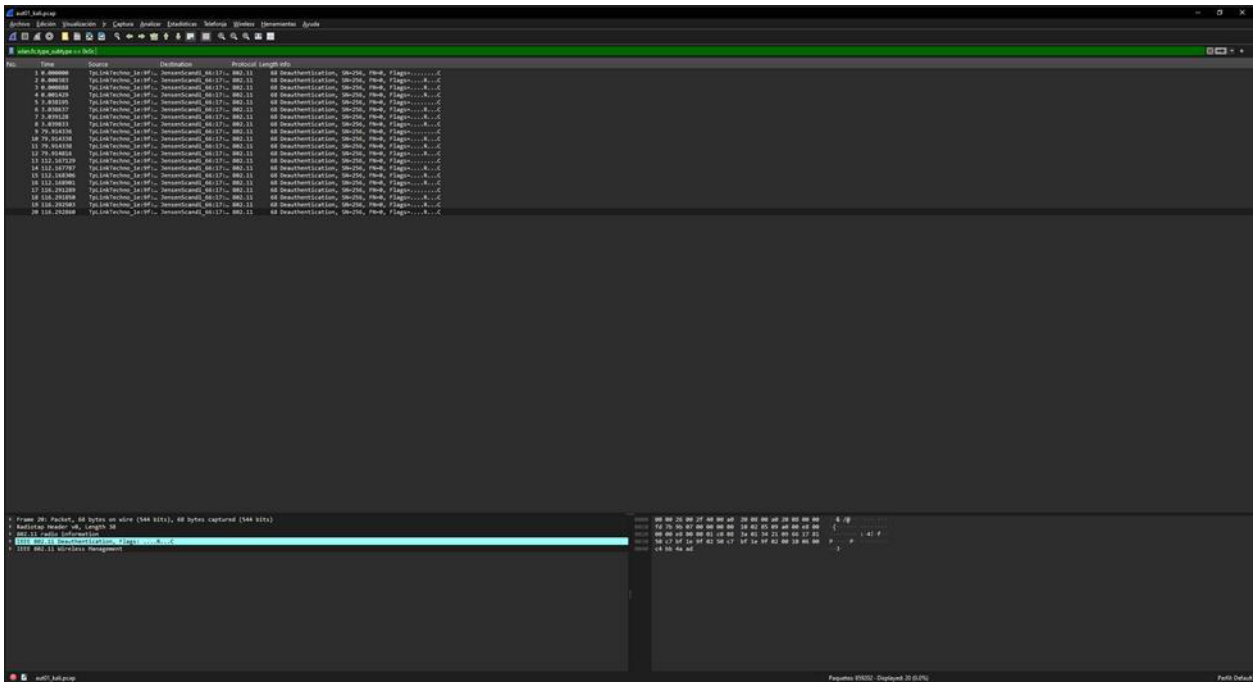
Anexo 20. *Filtro wlan.fc. type subtype == 0x0b (Authentication)*

Display filter para visualizar exclusivamente tramas de autenticación recibidas durante el flood.



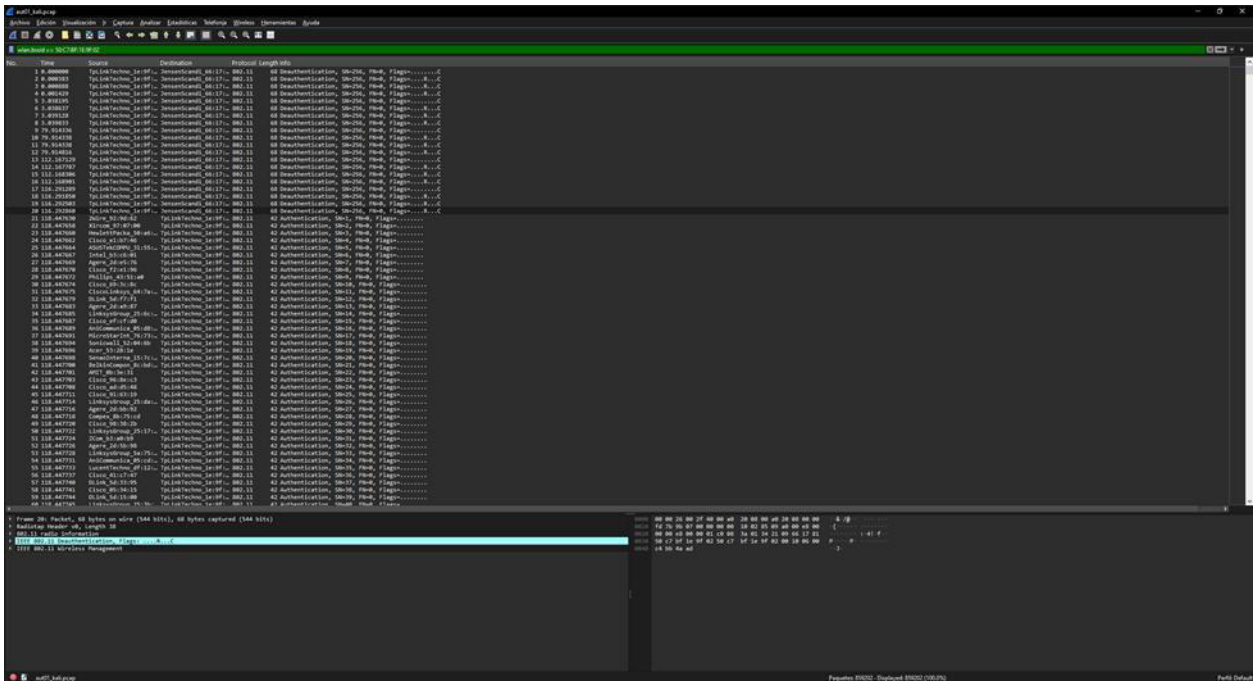
Anexo 21. *Filtro wlan.fc.type_subtype == 0x0c (Deauthentication)*

Filtro usado en Wireshark para verificar si el DUT recibe o genera tramas de desautenticación durante la prueba.



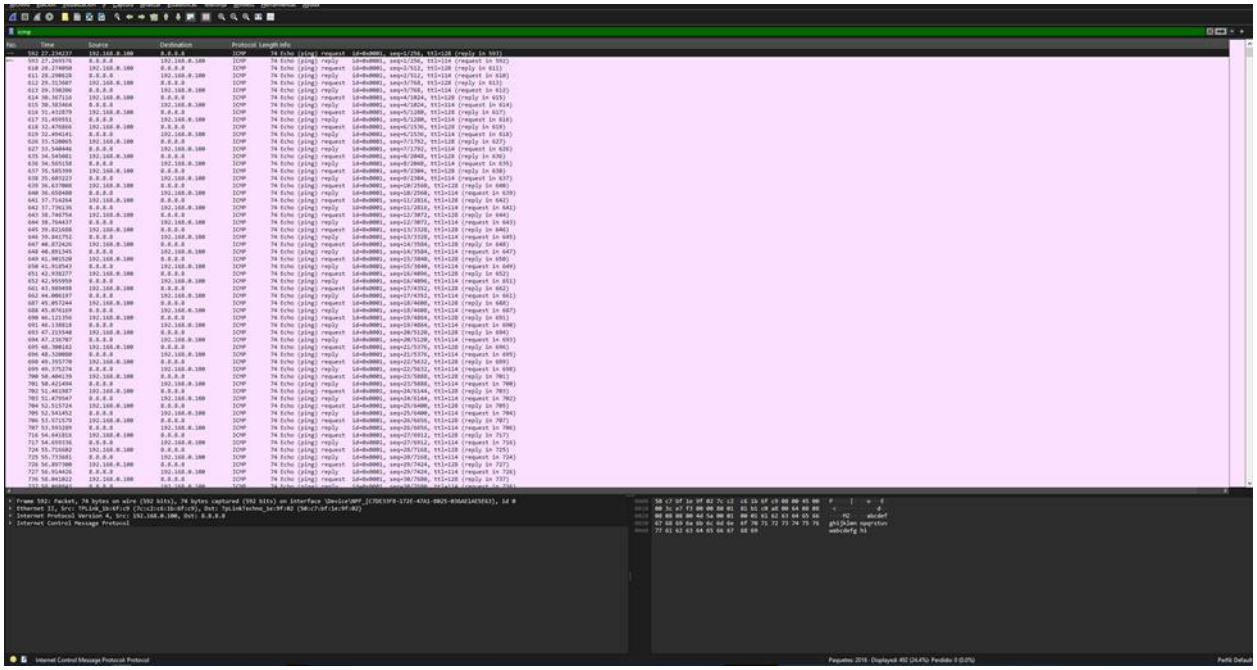
Anexo 22. *Filtro wlan.bssid == <BSSID>*

Filtro destinado a aislar únicamente el tráfico proveniente del punto de acceso objetivo.



Anexo 23. *Display de Wireshark en Kali (802.11 mgmt)*

Tramas de administración capturadas en Kali, usadas para análisis del comportamiento del DUT frente a respuestas manipuladas.

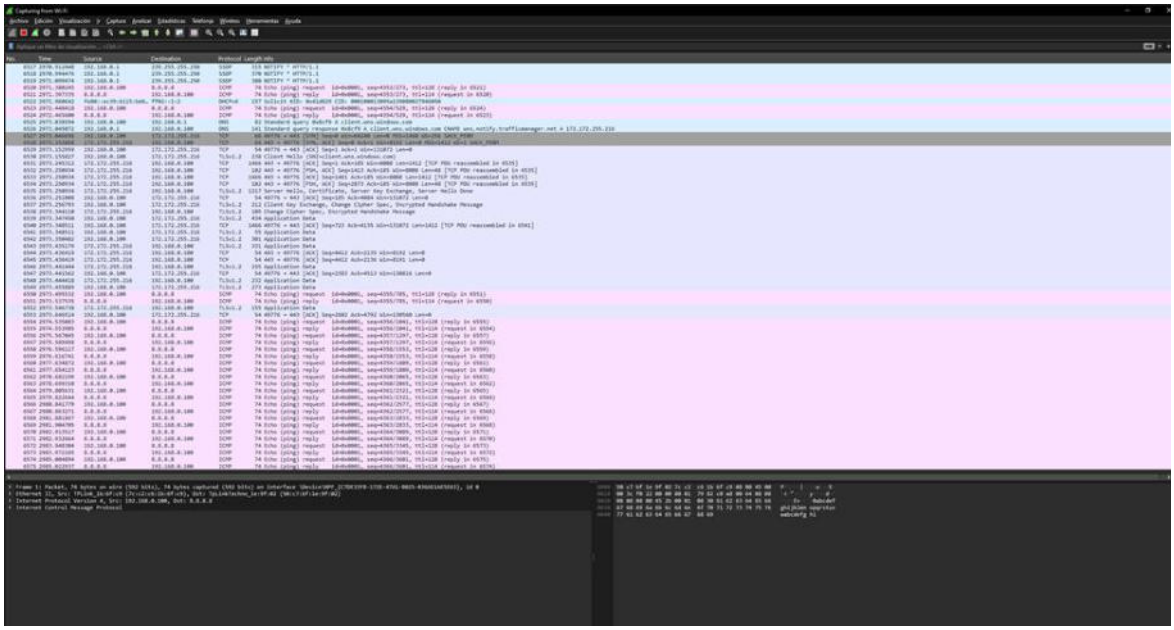


Anexo 24. *Kali — fijar canal + captura inicial*

Configuración del canal del adaptador en modo monitor más inicio de la captura pcap.

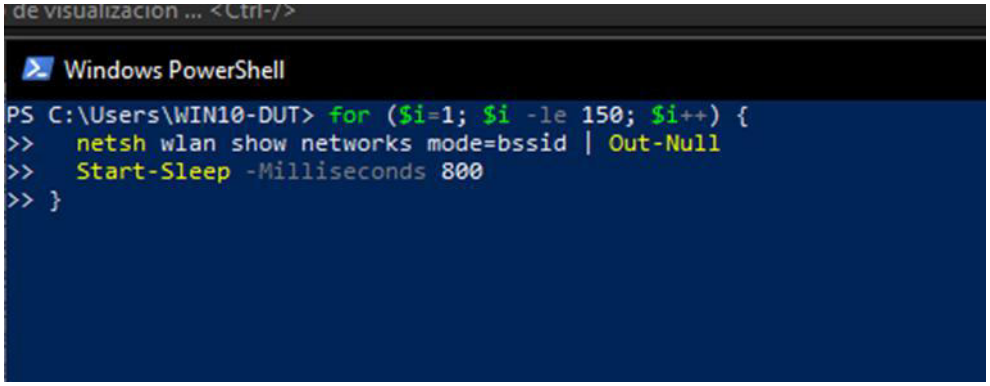
Anexo 25.*Windows generación forzada de Probe Requests*

Script de PowerShell que fuerza exploración activa (netsh wlan show net Works) para generar Probe Requests durante el fuzzing.



Anexo 26. *Kali ejecución del Fuzzer probe response*

Ejecución del script Python que envía Probe Responses alteradas para evaluar la robustez del DUT.



Anexo 27. *Pantalla de Kali ejecutando el Fuzzer (parámetros del script)*

Detalle de los parámetros utilizados por el Fuzzer (canal, SSID, MAC del DUT, cantidad de respuestas).

```
Session  Acciones  Editar  Vista  Ayuda

┌──(kali㉿kali)-[~]
└─$ sudo python3 ~/LAB_WIFI_v3/02_Scripts_Injector/proberesp_fuzz.py \
  --iface wlan0 \
  --ap 50:C7:BF:1E:9F:02 \
  --ssid TESTING \
  --chan 6 \
  --sta 7C:C2:C6:1B:6F:C9 \
  --count 200 \
  --burst 4

[i] Fijado a canal 6. Escuchando Probe Requests y respondiendo ...
```

Anexo 28. *PCAP de Kali filtrado*

Visualización en Wireshark del pcap resultante del fuzzing de Probe Responses, mostrando tramas respondidas y comportamiento del DUT.