Escuela Politécnica Superior

Análisis, diseño y desarrollo de una aplicación para la realización automática de pentesting

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
José Gaspar Cano Esquibel

Tutor/es:
Francisco José Mora Gimeno

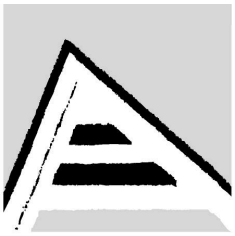
Junio 2019

Universitat d'Alacant
Universidad de Alicante

UNIVERSIDAD DE ALICANTE
TRABAJO DE FIN DE MÁSTER

Análisis, diseño y desarrollo de una aplicación para la realización automática de pentesting

Autor: José Gaspar Cano Esquibel
Tutor: Francisco José Mora Gimeno



Universitat d'Alacant
Universidad de Alicante

Junio 2019

Versión del documento
2019.06.01

Licencia

Motivación, justificación y objetivo general

La realización de las pruebas de penetración es un tema de máxima vigencia y actualidad, las pruebas de penetración son una exigencia diaria y primordial para garantizar la seguridad de los sistemas de información.

Uno de los trabajos principales en el día a día de los administradores de sistemas es la automatización de las tareas que realizan frecuentemente de manera manual. La automatización aumenta la productividad y reduce la probabilidad de errores. Ante esto surge la idea del presente trabajo, la necesidad de automatizar las distintas etapas de un pentesting mediante herramientas Open Source que ya existen y son utilizadas ampliamente de forma manual e independiente. Por lo tanto, con este trabajo se propone el diseño y desarrollo de una aplicación que permita realizar todas las tareas de un test de penetración una forma automática e incluso planificada de forma periódica.

La justificación de este proyecto es la utilidad y garantía de uso, basada en una necesidad real que tienen las organizaciones, por lo que la automatización de este tipo de pruebas de penetración les proporciona un feedback continuo sobre el estado de seguridad de sus sistemas.

La originalidad de idea parte en unir las etapas del pentesting hasta la explotación, utilizando herramientas de propósito general (Open Source) dentro del ámbito de la seguridad informática y la auditoria, de forma que cada herramienta no pierde su funcionalidad de ejecución y conserva su ciclo de actualización, al mismo tiempo que se pueden incorporar otras herramientas, como añadir las nuevas funcionalidades que puedan ofrecer.

Por lo tanto, el objetivo de este trabajo final de máster es el análisis, diseño y desarrollo de una aplicación que permita la automatización de las distintas etapas presentes en las auditorías técnicas de seguridad o pentesting. La aplicación recibirá como entrada información relativa al objetivo, el dominio de la infraestructura, y empleará técnicas de auditoría activa y/o pasiva, con el fin de implementar de manera automática distintas fases del proceso de pentesting.

Agradecimientos

Mi especial agradecimiento al director del Máster de Ciberseguridad y de este PFM
Francisco Mora, y a la entidad para la que trabajo el Banco de Sabadell
por haber apostado por este Máster junto a la UA.

Dedicatoria

A mi Madre, Esperanza.

Cada paso hace camino y cada piedra hace pared.

Jaime Cano García

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En 1971 con este mensaje "Soy una enredadera... ¡atrápame si tú puedes!" (1), que empezó a aparecer de repente en varios ordenadores de la red ARPANET (la antecesora de Internet), encontramos lo que se considera el primer virus informático, al que se denominó Creeper (enredadera). No era un programa malicioso, no pretendía causar daño alguno, y simplemente viajaba de un nodo a otro de la red replicándose a sí mismo y mostrando su mensaje. Escrito por Bob Thomas, infectaba ordenadores DEC PDP-10 con sistema operativo TENEX. Esto seria el revulsivo que sirvió para la creación del primer antivirus de la historia en 1972, apareció Reaper (segadora), su misión eliminar a Creeper de los ordenadores por los que se había propagado, imitaba su actitud, viajando y replicándose por los nodos de la red. Con el nacimiento de este primer antivirus, podemos hablar de los albores de una nueva disciplina en el mundo de las Tecnologías de la Información, la Ciberseguridad.

La seguridad informática o de las tecnologías de la información, conocida también como Ciberseguridad, es el área relacionada con la informática focalizada en la protección de las infraestructuras, comunicaciones y datos de los sistemas informáticos. La ciberseguridad comprende tanto software como hardware, así como todo elemento de las organizaciones que puesto en valor, su pérdida o compromiso, signifique una pérdida de servicio o disminución de este.

El objetivo de la ciberseguridad es por tanto su habilidad de identificar y eliminar vulnerabilidades existentes, y conseguir predecir futuras amenazas tratando de ser lo mas proactivos posibles, y al mismo tiempo fijar una política de seguridad que dirija la conveniencia y la coordinación de sus acciones para reducir los riesgos.

Según la RAE se establece Hacker como “persona experta en el manejo de computadoras, que se ocupan de la seguridad de los sistemas y que desarrollan técnicas de mejora”. Comúnmente este término es asociado a todo aquel experto que utiliza sus conocimientos para superar un problema asociado a la seguridad y mejorar la seguridad de los sistemas informáticos, empleando también el nombre de Hacker ético. Se debe distinguir de los piratas informáticos o ciberdelincuentes.

La siguiente figura muestra el incremento en la demanda de especialistas en Ciberseguridad desde el año 2014, para una de las empresas lideres en contratación de IT en España, Infojobs.

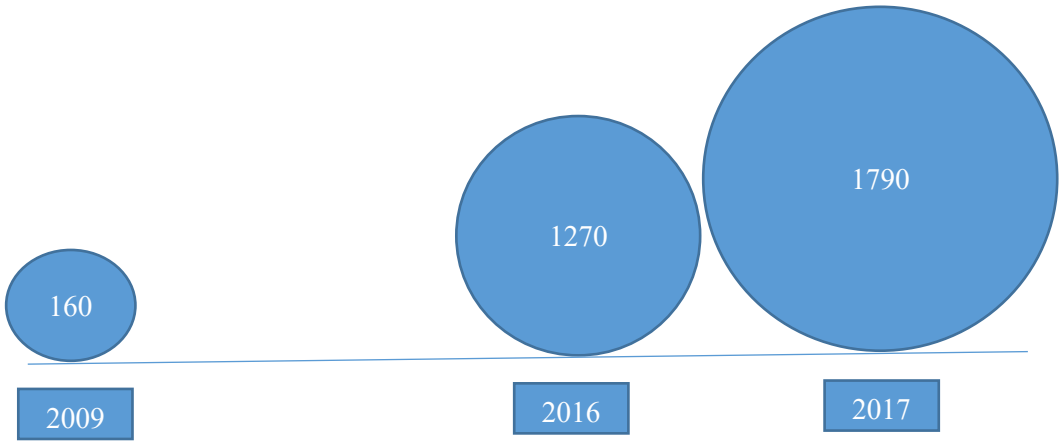


Figura 1. Demanda en Infojobs de especialistas Ciberseguridad (2)

El objetivo de los ciberdelincuentes es atacar los sistemas en la búsqueda de un beneficio propio, por el contrario, para un Hacker ético, el ataque de los sistemas les permite determinar las superficies de ataque y de esta forma plantear las posibles mejoras o soluciones a esas posibles vulnerabilidades, siguiendo siempre postulados éticos y legales. Sin embargo, los hackers y los ciberdelincuentes utilizan las mismas habilidades, conocimientos y herramientas. Al proceso que realiza una auditoría técnica de seguridad de los sistemas informáticos se denomina "pentesting" que una abreviatura de Prueba de Penetración (penetration testing).

Los conceptos de hacking ético y pentesting están relacionados con el propósito de la elaboración de informe técnico autorizado (legalmente) para descubrir y explotar satisfactoriamente los sistemas vulnerables, cuyo propósito es determinar su exposición ante un ataque informático real, con el objetivo final de corregir las vulnerabilidades encontradas para hacer los sistemas más seguros.

2. Estado del arte.

El Pentesting, o pruebas de penetración de sistemas informáticos puede ser definido como el estudio, autorizado por el propietario de sus sistemas y por lo tanto legal, de las vulnerabilidades presentes en un sistema objetivo. Esta batería de pruebas que evalúen los sistemas concluirá con un informe de la situación actual, junto con pruebas de explotación que evalúen la exposición real de los sistemas objetivos.

El proceso de prueba de penetración (pentesting), sigue una metodología que permite dividir el proceso en una serie de fases más pequeñas. En general, las pruebas de penetración se dividen en las siguientes etapas:

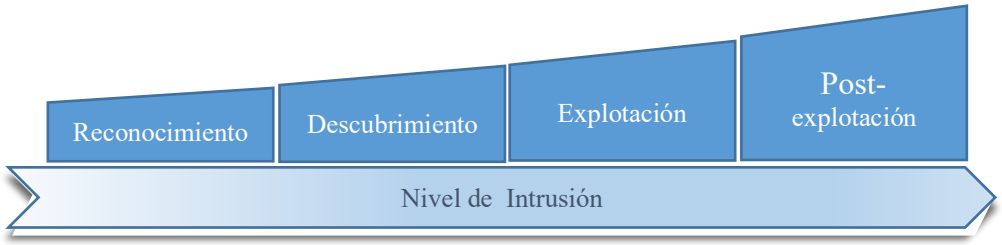


Figura 2. Etapas del Pentesting

Primera fase, reconocimiento, recogida de información, footprinting pasivo u OSINT (Open Source Intelligence). Se le denomina reconocimiento pasivo, porque las tareas que se realizan en ningún momento son intrusivas frente a la infraestructura de la organización que es objetivo de la recolección de la información, por lo que no suele ser detectada por los sistemas de seguridad de la empresa objetivo. La información que se recolecta del objetivo proviene de fuentes públicas disponibles en internet, es decir, el footprinting recoge información del objetivo a través de todos los directorios de información existentes en internet. La etapa de reconocimiento tiene una gran importancia dentro del pentesting puesto que es el punto de partida, donde más información se debe recoger y sobre la que se realizarán los análisis posteriores.

La segunda fase o descubrimiento activo, recoge la información directamente de la infraestructura de la empresa, por lo que sí puede ser detectado por los sistemas de seguridad. El objetivo es una recolección y exploración de la infraestructura objetivo, así como un análisis de vulnerabilidades.

La tercera fase es la explotación del sistema objetivo, esta es una etapa puramente intrusiva y hace uso de la información obtenida en las fases anteriores, así como las vulnerabilidades descubiertas para así realizar un ataque, y conseguir acceso al sistema objetivo. El éxito de la explotación dependerá de la calidad de la información obtenida en las etapas anteriores, así como de la identificación de las vulnerabilidades encontradas en la infraestructura del objetivo.

Existe una ultima fase, post-explotación, cuyo objetivo es seguir explotando un sistema al que ya tenemos acceso, es decir mantener la persistencia del acceso conseguido, para lo cual necesitamos elevar los privilegios obtenidos (si no se consiguió previamente), penetrar en una subred interna (pivoting), cubrir rastros (en la medida que sea posible evitar que seas detectado y mantener la persistencia de acceso al sistema).

Etapa	Denominación	Nivel de Intrusión	Detectable
1	Reconocimiento	Pasivo	No
2	Escaneo	Activo	Si
3	Explotación	Activo	Si
4	Post-Explotación	Activo	Si

Tabla 1. Detección/Intrusión de Pentesting

El grado de conocimiento del objetivo determina el tipo de pentesting a realizar, en este sentido los podemos clasificar en tres tipos:

- Caja blanca: La información con la que se realiza el pentesting sobre el sistema objetivo es completa, este tipo pruebas las realizan en las organizaciones los departamentos de auditoria interna de manera regular para calibrar objetivos.
- Caja gris: La información con la que se realiza el pentesting sobre el sistema objetivo esta limitada sobre la infraestructura. Esta prueba esta diseñada para auditar las pruebas de penetración del personal de la empresa que tiene conocimiento parcial del sistema objetivo
- Caja Negra: La prueba de penetración se realiza sin ningún tipo de información sobre los sistemas objetivos, este tipo de pentesting está diseñado el punto de vista de la auditoria externa y trata de imitar el comportamiento previsible de un atacante externo malicioso.

2.1. Reconocimiento

El reconocimiento es la primera etapa del pentesting, donde se persigue recoger la mayor parte de información posible y que esta sirva de base para las siguientes etapas, mediante técnicas no intrusivas en los sistemas objetivos, es decir, se accede a la información publica disponible en internet, y en ningún momento se accede a los sistemas de la empresa. De esta manera, esta etapa se considera de reconocimiento pasivo y no es detectable por los sistemas de seguridad de la empresa objetivo del análisis.

La cuantía y el tipo de información que se puede conseguir es numerosa, podemos obtener entre otros los siguientes datos de esta fase de reconocimiento.

De Infraestructuras

- Topología de Red, dominio y subdominios
- Servicios de red existente: web, mail, ftp
- Tecnología empleada para desplegar servicios

Organizativa

- Jerarquía de personal de la empresa: CEO, empleados, etc.
- Oficinas
- Teléfonos
- Emails corporativos

Personal

- Relación de empleados
- Aficiones de los empleados, así como donde han estudiado, amigos, etc.
- Relaciones de los empleados con terceros

En este reconocimiento pasivo se intenta conseguir, a través de herramientas OSINT, la información descrita de fuentes públicas.

Existen muchas formas de realizar el reconocimiento pasivo, dentro del vasto número de herramientas en esta etapa podríamos citar:

- Información Administrativa de Infraestructuras en registro públicos, “https://whois.ws”
- Búsquedas avanzadas en Google/Bing aplicado directivas específicas.
- Búsqueda en Base de datos de dispositivos, como Shodan (3)
- Herramientas especializadas como MALTEGO (4) que muestra la información gráficamente y Robtex, que en teoría permite realizar footprinting activo como si fuera pasivo
- Ingeniería de redes sociales (linkedin, Facebook, Twitter) a través de buscadores

2.2. Descubrimiento

El objetivo de la etapa de descubrimiento es el análisis de todas las vulnerabilidades presentes dentro de las infraestructuras de la empresa objetivo, esta información será explotada posteriormente.

El origen de partida de esta etapa es la información proveniente de la fase anterior, que era un reconocimiento pasivo, a partir de información publica. En esta etapa podemos recopilar más información sobre las infraestructuras de la empresa objetivo como IPs públicas, servidores activos, servicios e información sobre usuarios.

En general, los pasos a seguir para realizar la etapa de descubrimiento de la red objetivo son los siguientes:

- Obtención de equipos activos
- Descubrimiento de puertos abiertos en dichos equipos
- Descubrimiento de los servicios en ejecución
- Obtención del sistema operativo de los equipos
- Obtención de información perimetral
- Realización de un análisis de vulnerabilidades

Después de esta etapa, el analista de seguridad dispone de toda la información de la empresa objetivo relativa a los equipos activos de la empresa accesibles remotamente, sus sistemas operativos, los servicios que están ejecutando, los puertos que tienen abiertos, si existe o no seguridad perimetral y un listado con las vulnerabilidades encontradas en la infraestructura.

En esta etapa se emplean, principalmente, dos tipos de herramientas: un escáner de red y un analizador de vulnerabilidades. El escáner de red por excelencia es NMAP, mientras que OpenVAS y Nessus (5) constituyen dos ejemplos de analizadores de vulnerabilidades.

2.3. Explotación

En función de la información obtenida en las fases anteriores, reconocimiento y descubrimiento, habremos obtenido los objetivos y las posibles vulnerabilidades, relacionadas básicamente con debilidades de configuración y de la tecnología empleada (desbordamiento de buffer, errores en protocolos). Por lo que el principal vector de entrada en la explotación de los sistemas, son estas debilidades presentes las implementaciones del sistema operativo, aplicaciones y servicios.

La aplicación estándar como plataforma de explotación es Metasploit Framework que permite uso no comercial (6), a diferencia de Core Impact.

Metasploit Framework permite que se ejecute un payload asociado a un ataque (5), que en la mayoría de las ocasiones tiene acciones que tienen como objetivo la obtención de una shell o intérprete de comandos, de manera directa o inversa. Los sistemas de defensa cuentan con firmas muy exactas de los payloads por defecto, que los detectan con mucha facilidad, abortando su ejecución.

2.4. Post-Explotación

La fase de Post-explotación está formada por las tareas que se llevan acabo después de haber explotado un sistema, y constituyen el objetivo real de los atacantes maliciosos. Esta fase está fuera del objetivo de una prueba de penetración puesto que la obtención de los activos concretos es objetivo del atacante, pero se realiza para mostrar el impacto asociado a las brechas de seguridad de los sistemas de la empresa.

Meterpreter es la herramienta principal que permite pasar de la explotación a la post-explotación de una forma casi directa. Meterpreter es un payload muy avanzado que se puede ejecutar después de explotar un sistema (8).

Esta etapa esta fuera ámbito del proyecto, puesto que este trabajo final de máster sólo contempla el objetivo básico de un pentesting que es la explotación o acceso a los sistemas.

3. Objetivos

El Pentesting puede ser definido como el estudio autorizado por el propietario, y por lo tanto legal, de las vulnerabilidades presentes en la infraestructura informática de una empresa. Estas pruebas de penetración de sistemas informáticos proporcionan una prueba o evidencia de la vulnerabilidad de los sistemas informáticos, frente a un ataque real malintencionado realizado para comprometer estos sistemas. El pentesting nos ofrece la posibilidad de corregir las vulnerabilidades encontradas, evitando el ataque real, en la medida de lo posible.

El principal objetivo de este proyecto es el diseño y desarrollo de una aplicación que realice de manera automática varias etapas de un pentesting. La aplicación, a partir de objetivos determinados, podrá ejecutar automáticamente el proceso de pentesting completo: reconocimiento, descubrimiento y explotación. Este proyecto en todo momento se aborda desde el punto de vista del hacking ético, de forma que el informe elaborado nos permita realizar mitigación de las vulnerabilidades detectadas.

Este pentesting automático debe cubrir las tres primeras etapas de una evaluación técnica de seguridad: reconocimiento, descubrimiento y explotación. La explotación es el objetivo principal.

Dado que la consideración de partida está relacionada con el hacking ético y legal, en una situación real se necesitaría permiso explícito y por escrito mediante contrato de la empresa que requiere las pruebas de penetración antes de poder iniciar cualquier actividad. El ámbito y alcance de la prueba debe quedar claro en este contrato de auditoria.

Dentro del contrato se debe especificar que acciones están permitas realizarse y cuales no. Los métodos y estrategias deben ser los mismos que emplearía un atacante malicioso, por lo tanto, usaremos herramientas de seguridad y análisis forense ampliamente conocidas y utilizadas por atacantes maliciosos.

Los resultados de las pruebas de penetración, así como cualquier tipo de información asociada, perteneciente al cliente y no podrá ser divulgada fuera de la organización del cliente que encargo las pruebas.

El objetivo secundario del trabajo es la generación, también de manera automática, de un informe de auditoría con las vulnerabilidades detectadas, así como evidencias de las explotaciones realizadas en el sistema objetivo.

Las conclusiones, así como los puntos de mejora o soluciones a las deficiencias encontradas, que suelen forma parte del contrato estándar de unas pruebas de penetración, quedaran fuera del ámbito de este informe automático.

4. Metodología

El conjunto de los procedimientos para la gestión de todos los componentes de un proyecto se constituye como la metodología empleada para su realización. En la siguiente sección describiremos la metodología de desarrollo para este proyecto que tiene como objetivo principal, la creación de una herramienta para la automatización de las diferentes etapas en las pruebas de penetración y explotación.

4.1. Metodología de desarrollo

La metodología de desarrollo se ha encajado en un modelo incremental de funcionalidades, es decir, se han elegido unas pocas herramientas por cada etapa de forma que podamos validar el modelo general de automatización, que es el propósito de este proyecto. Con este planteamiento será sencillo ir añadiendo funcionalidades o incrementar las ya existentes simplemente añadiendo herramientas a cada etapa y siguiendo el modelo existente.

La metodología incremental se ha seguido tanto vertical como horizontalmente. Incremento vertical porque se ha desarrollado primero el módulo de reconocimiento, después se ha integrado el de descubrimiento y, finalmente, se ha añadido el módulo de explotación. Incremento horizontal porque en cada una de las etapas o módulos anteriores se pueden ir añadiendo nuevas herramientas, extendiendo la funcionalidad de la aplicación.

En todo momento se ha buscado un modelo puramente funcional y que puede evolucionar de una manera estable, que cubra plenamente el alcancé de este proyecto.

5. Desarrollo e implementación

Para abordar la realización de este proyecto, ha sido requerido el despliegue de una infraestructura tecnológica, necesaria tanto para el desarrollo como para la ejecución, así como el uso de un conjunto de herramientas para la implementación y desarrollo. En las siguientes secciones describimos el sistema operativo, el sistema gestor de BBDD, las herramientas para cada etapa, el lenguaje y las librerías empleadas para cubrir la penetración, la explotación y la generación del informe implementadas.

5.1. Infraestructura tecnológica

Para la realización del proyecto se ha contado con la distribución GNU Linux Kali, líder dentro en el ámbito de la seguridad informática, tanto en el campo del pentest como en el campo del análisis forense. A la distribución base KALI se le han añadido los elementos necesarios para consecución del proyecto.

El lenguaje de programación empleado es Python, escogido por ser es uno de los mas empleados para la creación de herramientas de sistemas, algo que garantiza la continuidad del proyecto. Para el entono de desarrollo se ha utilizado el IDE de Python WinPersonal, por su sencillez y su entono de depuración amigable, que simplifican las condiciones de desarrollo y depuración.

A continuación, se mostrarán las características del entorno de desarrollo y ejecución en la creación del proyecto, incluyendo: sistemas operativos, el lenguaje empleado, librerías, herramientas, base de datos, IDE y sistemas de testeo utilizados.

5.1.1. Kali

El sistema operativo empleado es un GNU/Linux, concretamente la distribución Kali, por ser una de distribuciones lideres en Pentest (9). Kali es la distribución sucesora de la distribución GNU/BackTrack, pero basado en el Core Debian, esta diseñada básicamente para la seguridad y la auditoria informática y esta mantenida por la compañía Ofensive Security LTD.

Figura 3. Logo Kali Linux (10)

La distribución Kali Linux trae preinstalados más de 600 programas, puede ser usada desde un Live CD, live-usb o como sistema operativo principal. Kali se distribuye en imágenes ISO compiladas para diferentes arquitecturas (32/64 bits y ARM), existen imágenes compatibles con los principales hypervisores de virtualización.

Para garantizar un entorno seguro de desarrollo, el equipo de Kali, esta formado por un reducido numero de personas que interactúan con los repositorios oficiales, todos los paquetes estan firmados por los desarrolladores que los compilaron. Los paquetes también serán firmados en sus repositorios utilizando GNU Privacy Guard, para garantiza su autenticad.

Desarrollador Offensive Security	kali.org
Modelo de desarrollo	Código abierto
Lanzamiento inicial	13 de marzo de 2013
Última versión estable	2019.1 (Rolling)
Núcleo	Linux, FreeBSD, Hurd
Tipo de núcleo	Monolítico, Micro
Interfaz gráfica predeterminada	GNOME, KDE, Xfce o LXDE
Plataformas admitidas	x86, x86-64 y ARM
Sistema de gestión de paquetes	dpkg
Método de actualización	APT (varias interfaces disponibles)
Licencia	GPL con programas y
Estado actual	En desarrollo
Idiomas	Multilingüe
Predecesor:	BackTrack

Tabla 2. Resumen de Kali (11)

5.1.2. Python

Python es un lenguaje de programación interpretado, orientado a objetos, imperativo y que soporta la programación funcional, usa tipado dinámico y es multiplataforma. Su filosofía es su sintaxis, que tiende a facilitar que el código sea ante todo legible. Administrado por la Python Software Foundation fue creado en 1991 por Guido Van Rossen.

Figura 4. Logo Python (12)

La facilidad de extensión de Python es también uno de sus objetivos de diseño, permite escribir nuevos módulos fácilmente en C o C++. Python dispone de una numerosa librería estándar de módulos para multitud de usos de propósito general, y con un especial atractivo para la creación de programas de Networking (13) (14) y seguridad, la tabla siguiente muestra un resumen de características del Python.

Desarrollador	Python Software Foundation www.python.org
Extensiones	py, .pyc, .pyd, .pyo, .pyw, .pyz
Paradigma	orientado a objetos, imperativo, funcional, reflexivo
Diseñado por	Guido van Rossum (1991)
Última versión	3.7.3
Sistema de tipos	Fuertemente tipado, dinámico
Implementaciones	CPython, IronPython, Jython, PyPy, Pygame, ActivePython
Dialectos	Stackless Python, RPython
Influido por	ABC, ALGOL , C, Haskell, Icon, Lisp,Modula3, Perl, Smalltalk
Ha influido a	Boo, Cobra, D, Falcon, Genie, Groovy, Ruby, JavaScript, Cython
Sistema operativo	Multiplataforma
Licencia	Python Software Foundation License

Tabla 3. Resumen de Python (15)

En febrero de 2009 se lanzó una nueva versión de Python la 3.0 esta nueva versión incluye toda una serie de cambios que obligan a reescribir el código de versiones previas.

Como resumen, Python combina una sintaxis clara, multitud herramientas dentro de su biblioteca estándar y la posibilidad de usar lenguajes de bajo nivel como C y C++, esto nos permite interactuar con numerosas bibliotecas. Esto describe su versatilidad, potencia y utilización como muestra la siguiente tabla, por lo que se ha convertirlo en uno de los lenguajes mas utilizados después del Java y el C/C++.

May 2019	May 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.005%	-0.38%
2	2		C	14.243%	+0.24%
3	3		C++	8.095%	+0.43%
4	4		Python	7.830%	+2.64%
5	6	▲	Visual Basic .NET	5.193%	+1.07%
6	5	▼	C#	3.984%	-0.42%
7	8	▲	JavaScript	2.690%	-0.23%
8	9	▲	SQL	2.555%	+0.57%

Tabla 4. Índice TIOBE (16) de utilización mundial de Leguajes de Programación

5.1.3. SQLite

SQLite es un gestor relacional de bases de datos compatible con ACID, escrito en C , esta biblioteca requiere menos de 300Kb, es un proyecto de dominio público creado por Richard Hipp.

Figura 5. Logo SQLite (20)

Como diferencia principal con otros gestores de BBDD, SQLite no esta basado en modelo cliente-servidor, por lo que no es un proceso independiente. Es una biblioteca que se enlaza con el programa y forma parte de su código y su hilo de ejecución, la funcionalidad a SQLite se obtiene a través de llamadas a su API (subrutinas y funciones). Con esto se consigue unos tiempos de latencia muy bajos a las llamadas a la librería, así como la eficiencia de la comunicación entre procesos, este planteamiento de trabajo es valido cuando se requieren pocos accesos a una BBDD (simultáneos). La estructura de la base de datos (definiciones, tablas, índices, y datos), forman parte de un único fichero en la misma maquina donde se ejecuta la aplicación.

Este diseño tan simple se consigue mediante el bloqueo de un único fichero durante toda la transacción. Desde la versión 3 el limite de tamaño esta en 2 Terabytes , y se admiten campos del tipo BLOB.

Modelo de desarrollo	Dominio Público
Desarrollador	D. Richard Hipp
Autor	D. Richard Hipp
Lanzamiento inicial	agosto de 2000
Última versión estable	3.28.0
Género	RDBMS
Programado en	C
Sistema operativo	Multiplataforma
Tamaño	~275 kiB
Licencia	Dominio público

Tabla 5. Resumen de SQLite (21)

5.1.4. Herramientas y librerías

Las herramientas utilizadas constituyen el núcleo generador de la información en las diferentes etapas de las pruebas de penetración y explotación, las librerías son el nexo para su utilización e interconexión, así como el medio para la generación del informe final.

5.1.4.1. MessagePack

MessagePack es un formato binario para serializarían de datos, permite serializar estructura de datos simples como listas y diccionarios y existen implementaciones oficiales para múltiples lenguajes y plataformas.

Figura 6. Logo MessagePack (17)

Messagepack es el formato de intercambio que emplean algunos servidores de RPC para comunicarse con su cliente, por su velocidad y eficiencia.

Autor	Sadayuki Furuhashi
Ultima versión	0.5.7
Programado en	Varios lenguajes
Disponible para	C, C++, C#, D, Erlang, Go, Haskell, Java, JavaScript, Lua, OCaml, Perl, PHP, Python,
Sistema operativo	Multiplataforma
Funcionalidad	Intercambio de datos
Licencia	Apache License
Website	msgpack.org

Tabla 6. Resumen de MessagePack (18)

A diferencia de JSON que es un standard de facto en intercambio de datos, MessagePack es mas rápido y necesita transmitir menos información, ya que para formatos numéricos pequeños utiliza un solo byte, así como paras las cadenas pequeñas requeriré unos pocos bytes. La siguiente figura compara la eficiencia en la transmisión de datos entre JSON y Messagepack, para un mismo mensaje, se emplean 18 bytes en Messagepack frente a 27 en JSON.

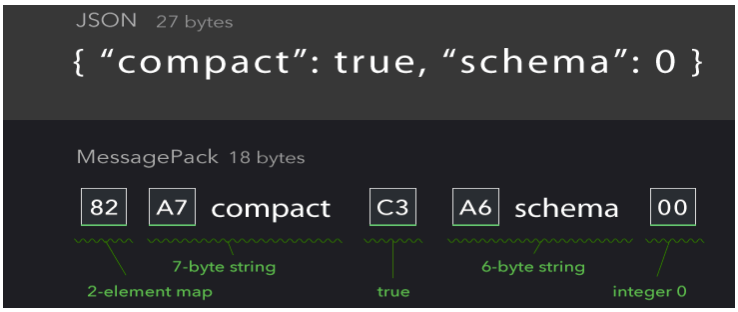


Figura 7. Comparación JSON vs MessagePack (17)

5.1.4.2. ReportLab

Reporlab es una Librería que permite crear documentos directamente en el formato de documento portables de Adobe (PDF). Esta escrita utilizando el lenguaje de programación Python. Su versatilidad le permite crear gráficos, tablas, insertar imágenes, anotación y diferentes objetos. El formato PDF es el estándar global para documentos electrónicos, es compatible con la impresión de alta calidad.

Figura 8. Logo Report Lab (19)

Las especificaciones de PDF son complejas (tienen mas de 600 páginas), es un binario indexado que es imposible de escribir directamente, ya que se deben proporcionar compensaciones de bytes precisas. Esto hace que sea más difícil de generar que HTML, la mayoría de los documentos PDF son producidos por las herramientas como Acrobat de Adobe, JAWS o PDF Creator, que funcionan como 'controladores de impresión', que generaran el PDF como si se tratase de una impresión.

La biblioteca ReportLab crea directamente PDF en función de sus comandos, esto le permite funcionar a alto nivel, con un motor completo que diseña documentos completos con tablas, cuadros gráficos y todo tipo de objetos, a continuación, mostramos un resume de la librería empleada.

Python Software Foundation www.reportlab.com	
Paradigma	orientado objetos, imperativo, funcional, reflexivo
Apareció en	2001
Diseñado por	ReportLab, Inc
Última versión	3.5 (2018)
Implementación	Python
Sistema operativo	Multiplataforma
Licencia	Python Software Foundation License

Tabla 7. Resumen de ReportLab (19)

5.1.4.3. KaliTools

La distribución Kali Linux contiene una gran cantidad de herramienta pertenecientes a diferentes ámbitos dentro de los campos de la seguridad y del análisis forense. Estas herramientas están recogidas en la distribución de Kali, pero están soportados en los repositorios de cada herramienta.

Figura 9. Logo Kali Tools (22)

Las KaliTools constituyen la lista de herramientas y utilidades de la distribución Kali Linux, abordan desde la recogida de la información y la explotación hasta la generación de informes finales. Esto permite a los profesionales de la seguridad IT contar con múltiples funcionalidades necesarias, para cubrir su desempeño, las Kali Tools cubren las siguientes categorías.

- Information Gathering
- Sniffing & Spoofing
- Vulnerability Analysis
- Exploitation Tools
- Password Attacks
- Wireless Attacks
- Forensics Tools
- Maintaining Access
- Hardware Hacking
- Web Applications
- Stress Testing
- Reverse Engineering
- Reporting Tools

Dentro de todas estas categorías y cubriendo las etapas que forman parte de este proyecto se cuentan las siguientes mostradas en la siguiente tabla.

Information Gathering (Etapa 1)			
• ace-voip • Amap • APT2 • arp-scan • Automater • bing-ip2hosts • braa • CaseFile • CDPSnarf • cisco-torch • copy-router-config • DMitry • dnmap • dnsenum • dnsmap • DNSRecon • dnstracer • dnswalk • Ghost Phisher	• GoLismero • goofile • hping3 • ident-user-enum • InSpy • InTrace • iSMTP • lbd • Maltego Teeth • masscan • **Metagoofil** • DotDotPwn • enum4linux • enumIAX • EyeWitness • Faraday • Fierce	• Firewalk • fragroute • fragrouter • Miranda • nbtscan-unixwiz • Nikto • Nmap • ntop • OSRFramework • p0f • Parsero • Recon-ng • SET • SMBMap • smtp-user-enum	• snmp-check • SPARTA • sslcaudit • SSLsplit • sslstrip • SSLyze • Sublist3r • THC-IPV6 • **theHarvester** • TLSSLed • twofi • Unicornscan • URLCrazy • Wireshark • WOL-E • Xplico
Vulnerability Analysis (Etapa 2)			
• BBQSQL • BED • cisco-auditing-tool • cisco-global-exploiter	• Doona • DotDotPwn • HexorBase • jSQL Injection • Lynis	• Oscanner • Powerfuzzer • sfuzz • SidGuesser • SIPArmyKnife • sqlmap	• tnscmd10g • unix-privesc-check • Yersinia

• cisco-ocs • cisco-torch • copy-router-config	• **Nmap** • ohrwurm • openvas	• Sqlninja • sqlsus • THC-IPV6	
Exploitation Tools (Etapa 3)			
• Armitage • Backdoor Factory • BeEF • cisco-auditing-tool • cisco-global-exploiter	• cisco-ocs cisco-torch • Commix • crackle • exploitdb • jboss-autopwn	• Linux Exploit Suggester • Maltego Teeth • **Metasploit Framework** • **MSFPC**	• RouterExploit • SET • ShellNoob • sqlmap • THC-IPV6 • Yersinia

Tabla 8. Resumen de Herramientas Kali (22) para Etapas 1,2 y 3

5.2. Implementación

La realización de las especificaciones técnicas necesarias, sobre la infraestructura tecnológica descrita en la sección anterior, son la base de la implementación realizada. Los medios utilizados se describirán en la próxima sección, así como el entorno de desarrollo que constituyen el desarrollo de la aplicación de automatización realizada en este TFM. También describiremos las herramientas necesarias en cada etapa, los mecanismos para poder recopilar toda la información generada en cada etapa y el medio para realizar la explotación.

Se detallarán los módulos necesarios y desarrollados para realizar la integración con sus interfaces para tal propósito. Se describirá el flujo de ejecución de cada etapa y su paso a la siguiente etapa, la escalabilidad en el diseño para la inclusión de nuevos módulos y herramientas, que permitirían añadir nuevas funcionalidades. Para finalizar detallaremos la generación del informe desarrollado como una recopilación de la información de las vulnerabilidades descubiertas y evidencias de la explotación, junto a su manifiesto de uso y utilización.

5.2.1. Entorno de desarrollo

El conjunto de procedimientos y herramientas que se utilizan para generar, probar y depurar un programa, constituyen el entorno de desarrollo. En las siguientes secciones describiremos los IDE (Integrated Development Environment) empleados, que proporciona servicios que facilitan el desarrollo, edición de código fuente, y de depuración, de una manera integrada y amigable.

5.2.1.1. IDE Python Personal Wing

Personal Wing, es un entorno integrado para desarrollo de Python amigable y con un entorno de depuración, que facilita el desarrollo y ejecución de programas en Python. Existen versiones de Personal Wing que se ejecutan en Windows, Mac OS X y Linux

Figura 10. Logo IDE Wings (27)

Desarrollador	Wingware
Lanzamiento	2000
Ultima versión	6.1.5
Escrito en	Python, Cython, C, C++
Sistema operativo	Windows, OS X, Linux
Funcionalidad	IDE para Python
Licencia	Propietario
Website	www.wingware.com

Tabla 9. Resumen de IDE Wing (27)

5.2.1.2. IDE SQLite Studio

El IDE SQLite Studio es la herramienta de desarrollo que posibilita la interacción con el sistema gestor de BBDD de SQLite, la consulta de los datos visualmente y la generación de las consultas de datos utilizadas el desarrollo de este proyecto.

Figura 11. Logo SQLite Studio (28)

Este IDE es una herramienta muy amigable (grafica) y totalmente funcional para cubrir todas las necesidades del proyecto, la siguiente tabla muestra información de SQLite Studio.

Desarrollador	SalSoft Pawel Salawa
Lanzamiento	2007
Versión estable	3.2.1
Sistema operativo	Windows, OS X, Linux
Funcionalidad	IDE para SQLlite
Licencia	PGPLv3
Website	https://sqlitestudio.pl

Tabla 10. Resumen de SQLite Studio (28)

5.2.2. Desarrollo

La aplicación desarrollada en este Trabajo Fin de Máster, denominada AutoPenExploit, es la suma de las abreviaturas de Automático, Penetración y Exploit. En las siguientes secciones describiremos la estructura de la aplicación y su implementación.

5.2.2.1. Herramientas integradas

La aplicación AutoPenExploit, tal y como se ha comentado está compuesta por una serie de módulos, que se muestran en la siguiente tabla, así como la tecnología empleada, el tipo de enlace que se empleó desde la aplicación Python con cada herramienta y el interfaz para su integración.

Paso	Etapa	Modulo	Programa			
			Aplicación	Versión	Call	Interfaz
1	1	Python_TheHarvester	TheHarvester	3.0.6	Shell	XML
2	1	Python_Metagoodfil	Metagoodfil	2.2	Shell	HTML
2	2	Python_NMAP	NMAP	7.70	Shell	XML
3	3	Python_Metasploit	Metasploit	5.0.20	RPC	MessagePack

Tabla 11. Resumen del herramientas integradas y interfaz utilizado

5.2.2.1.1. The Harvester (Etapa 1)

TheHarvester, tiene como objetivo descubrir información pública de múltiples repositorios como buscadores, servidores de claves PGP, bases de datos de ordenadores como SHODAN. Siendo la información obtenida emails, subdominios, host, nombres de empleados, puertos abiertos, etc.

Esta herramienta está destinada a ayudar a los analistas de seguridad en la primera etapa de las pruebas de penetración, para comprender la huella digital del cliente en Internet. También es útil para cualquier persona que quiera saber qué puede ver un atacante sobre su organización.

Source	https://github.com/laramies/theHarvester/
Repositorio Kali	http://git.kali.org/gitweb/?p=packages/theharvester.git;a=summary
Autor	Christian Martorella
Licencia	GPLv2

Tabla 12. Resumen de TheHaverter (22)

5.2.2.1.2. Metagoofil (Etapa 1)

Metagoofil es una herramienta de recopilación de información diseñada para extraer metadatos de documentos públicos (pdf, doc, xls, ppt, docx, pptx, xlsx) que pertenecen a una empresa objetivo.

Metagoofil realizará una búsqueda en Google para identificar y descargar los documentos en el disco local y luego extraerá los metadatos con diferentes bibliotecas como Hachoir, PdfMiner. Con los resultados, generará un informe con nombres de usuario, versiones de software y servidores o nombres de máquinas que ayudarán a los evaluadores de seguridad en la primera fase de recopilación de información

Source	http://www.edge-security.com/metagoofil.php
Repositorio Kali	http://git.kali.org/gitweb/?p=packages/metagoofil.git;a=summary
Autor	Christian Martorella
Licencia	GPLv2

Tabla 13. Resumen de Metagoofil (22)

5.2.2.1.3. NMAP (Etapa 2)

Nmap (Network Mapper) es una herramienta de código abierto y con licencia gratuita para el descubrimiento de redes, auditorías de seguridad, inventario de redes, inventarios de actualización de servicios y la supervisión del tiempo de actividad de los equipos informáticos. Nmap utiliza paquetes de IP sin procesarlos, para determinar qué hosts están disponibles en la red, qué servicios (nombre de aplicación y su versión), qué sistemas

operativos (y versiones de SO) están ejecutando, qué tipo de filtros de paquetes / cortafuegos están en uso (15), entre otras opciones.

Fue diseñado para escanear rápidamente grandes redes, pero funciona bien contra hosts individuales. Nmap se ejecuta en la mayoría de los sistemas operativos, y los paquetes de binarios oficiales están disponibles para Linux, Windows y Mac OS X. Además del ejecutable clásico de Nmap de línea de comandos, la suite Nmap incluye una GUI avanzada y un visor de resultados (Zenmap), una herramienta flexible de transferencia de datos, redireccionamiento y depuración (Ncat), una utilidad para comparar resultados de escaneo (Ndiff) y una herramienta de análisis de generación y respuesta de paquetes (Nping).

Source	http://nmap.org/
Repositorio Kali	http://git.kali.org/gitweb/?p=packages/nmap.git;a=summary
Autor	Fyodor
Licencia	GPLv2

Tabla 14. Resumen de NMAP (22)

5.2.2.1.4. Metasploit y MSGRPC Service (Etapa 3)

Metasploit es una plataforma de pruebas de penetración que permite encontrar, explotar y validar vulnerabilidades. Proporciona la infraestructura, el contenido y las herramientas para realizar pruebas de penetración y extensas auditorías de seguridad y, gracias a la comunidad de código abierto y al propio equipo de contenido de Rapid7, se agregan nuevos módulos de forma regular, lo que significa que el último exploit está disponible tan pronto como se publique.

Source	https://Metasploit.com/
Web Site	http://git.kali.org/gitweb/?p=packages/Metasploit-framework.git
Autor	Rapid7
Licencia	BSD-3-clause

Tabla 15. Resumen de Metasploit (22)

Metasploit Framework (16) permite que aplicaciones externas se puedan conectar e interactuar utilizado el Plugin MSGRPC, esto posibilita el acceso a los módulos y los exploits integrados en Metasploit. Este complemento genera una instancia de un servidor RPC local, ofreciendo todas las funcionalidades del framework de Metasploit, su funcionamiento se basa en serializar los mensajes del servidor RPC usando el formato MessagePack. Desde Python será posible emplear Metasploit usando el cliente *msgpack* y la biblioteca *python-msfrpc,* que es responsable de encapsular el intercambio de paquetes entre el servidor MSGRPC y el programa cliente que usa msgpack. De esta manera, es posible realizar una integración entre cualquier programa de Python y el framework de Metasploit, mediante el interfaz msgrpc y una poderosa API (17)

La carga del plugin se hace dentro de la consola del Metasploit, tal y como se muestra en la siguiente figura, especificado usuario, contraseña, utilización o no de SSL, y la dirección IP del servidor RPC que atenderá las peticiones.

Figura 12. Carga del Servicio msgrp del framework de Metasploit

Este servidor nos permitirá conectarnos con el MSF y poder realizar la explotación (26), así como ejecutar el payload a través del cual ejecutaremos un Shellcode en el sistema objetivo, y con esto obtendremos las evidencias de explotación (7) que formaran parte del informe como una vulnerabilidad real y evidenciada.

5.2.2.2. Composición de los Módulos

La aplicación AutoPenExploit esta desarrolla en Python y compuesta por una serie de módulos principales. Los módulos pertenecen a alguna de las etapas de un test de penetración analizadas anteriormente y ofrecen distintas funcionalidades. En la siguiente figura se pueden observar los distintos módulos y la etapa a la que pertenecen.

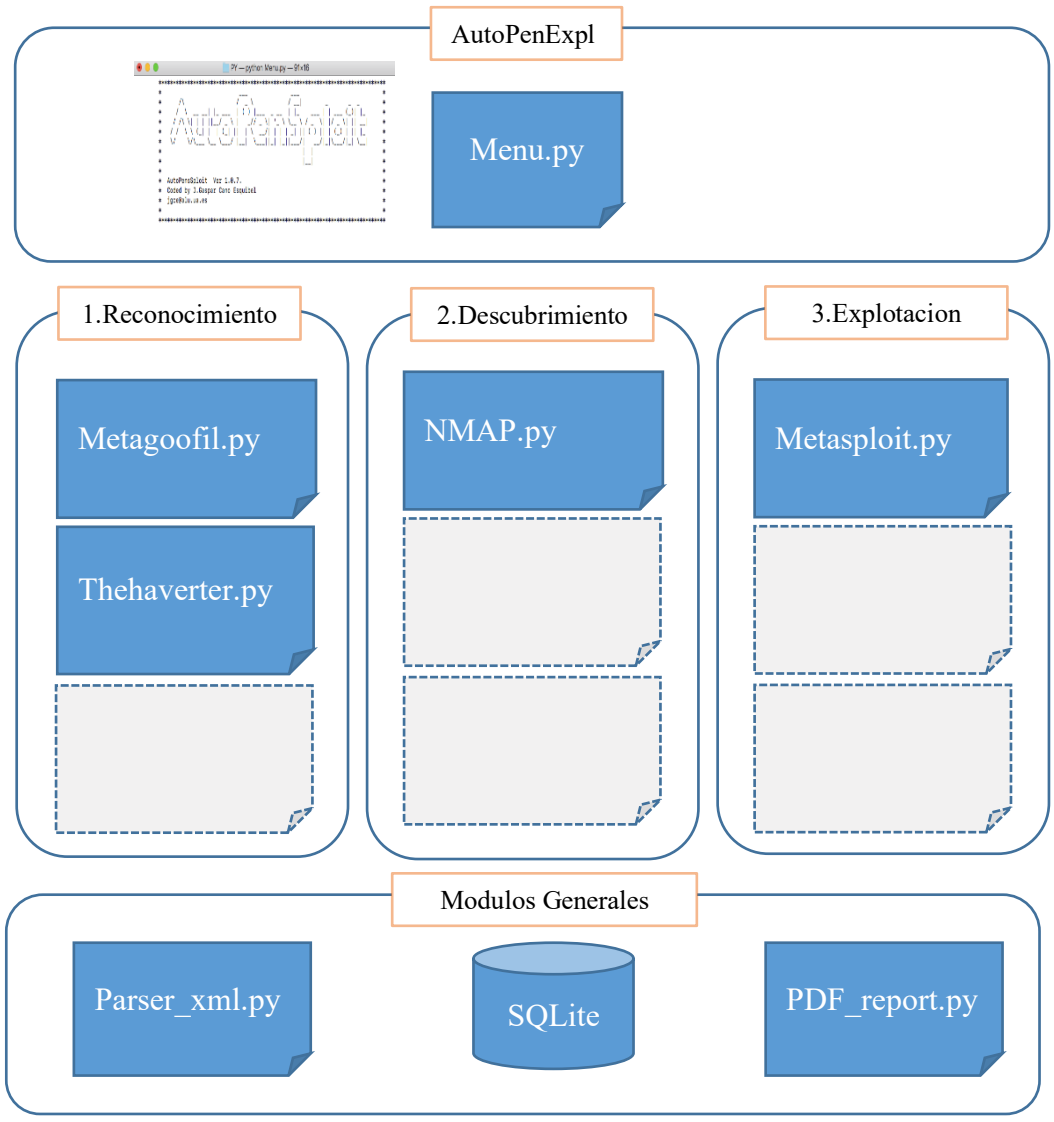


Figura 13. Componentes fundamentales de Aplicación AutoPenExploit

5.2.2.3. Implementación de los módulos

En las siguientes secciones describimos brevemente la implementación en Python de los módulos descritos anteriormente: la ejecución de las herramientas, la extracción de la información con su interfaz correspondiente, el almacenamiento en la BBDD, la explotación y la generación del informe.

5.2.2.3.1. Ejecución

Sección de código en Python que muestra el detalle de la ejecución para cada herramienta en un proceso independiente esperando su finalización.

```
60          comando = comando.replace('@path',self.path)
61          comando = comando.replace('@script',self.binario)
62          comando = comando.replace('@dominio',self.word)
63          comando = comando.replace('@fileresultado',self.fSalida)
64
65          print (comando)
66          args = shlex.split(comando)
67          s = subprocess.Popen(args, stdout=subprocess.PIPE)
68          p_status = s.wait()
```

Figura 14. Detalle de ejecución para una herramienta (metagoofil.py)

5.2.2.3.2. Interfaz

Sección de código del interfaz de HTML5 que realiza la extracción para la salida de la herramienta metagoofil y obtiene los elementos email, usuarios.

```
55      def resolve(self):
56          page = open(self.path+self.documento)
57          soup = BeautifulSoup(page.read(),"html5lib");
58
59          print('\[*] Parse Email item.');
60          for child in soup.find_all("li", attrs={"class" : "emailitem"}):
61              self.all_emails.append(child.text);
62
63          print('\[*] Parse User item.');
64
65          for child in soup.find_all("li", attrs={"class" : "useritem"}):
66              self.all_user.append(child.text);
```

Figura 15. Detalle de interfaz de HTML5 (metagoofil.py)

Sección de código del interfaz de XML que realiza la extracción para la salida de la herramienta theharvester, para los elementos email, hostname e IP.

```
88      def resolve(self):
89          tree = ET.parse(self.path+self.documento)
90          root = tree.getroot()
91
92          print('[*] Parse Hosts. \n');
93          for child in root.findall("host"):
94              self.all_ip.append(child.find('ip').text)
95              self.all_hosts_name.append(child.find('hostname').text);
96
97          print('[*] Parse Email. \n');
98          for child in root.findall("email"):
99                  self.all_emails.append(child.text);
00
```

Figura 16. Detalle de interfaz XML (thehaverter.py)

5.2.2.3.3. Almacenamiento

Sección de código que describe como se realiza el almacenamiento de la información entre la interfaz de una herramienta y almacenamiento en BBDD.

```
195
196                 db = bbdd.stash_manager();
197                 db.do_init();
198                 db.store_all(self.etapa,self.orden,self.word, self.all_port, 'port','estado',self.all_port_status, self.source);
199                 db.store_all(self.etapa,self.orden,self.word, self.all_port, 'port','service_name',self.all_port_service_name, self.source);
200                 db.store_all(self.etapa,self.orden,self.word, self.all_port, 'port','service_product',self.all_port_service_product, self.source);
201                 db.store_all(self.etapa,self.orden,self.word, self.all_port, 'port','service_version',self.all_port_service_version, self.source);
202                 db.store_all(self.etapa,self.orden,self.word, self.all_port, 'port','service_cpe',self.all_port_service_cpe, self.source);
```

Figura 17. Detalle de almacenamiento en BBDD (NMAP.py)

5.2.2.3.4. Explotación

Sección de código que describe como se realiza la creación de la consola de MSF mediante el cliente Msfrpc de Metasploit, que permite la invocación de los comandos de la explotación.

```
207
208     def CrearConsola(self):
209       self.client = Msfrpc({})
210       self.client.login(self.m_login,self.m_password)
211
212       try:
213         result = self.client.call('console.create')
214       except:
215         sys.exit("[!] Error Creacion de consola MSF !")
216
217       console_id = result['id']
218       console_id_int = int(console_id)
219       print "consola:"+ console_id
220       self.m_console=console_id
221
```

Figura 18. Detalle creación cliente Msfrpc (Metasploit.py)

5.2.2.3.5. Informe

Sección de código que describe como se genera la portada del informe mediante invocación a la librería de ReportLab.

```
59
60  def myFirstPage(canv, doc):
61      canv.setPageCompression(1)
62      canv.setFont("Times-Bold", 26)
63      canv.drawCentredString(0.5 * A4[0], 10 * inch, "Análisis, diseño y desarrollo")
64      canv.drawCentredString(0.5 * A4[0], 9.5 * inch, "de una aplicación para la realización")
65      canv.drawCentredString(0.5 * A4[0], 9.0 * inch, "automática de pentesting")
66      canv.setFont("Times-Bold", 30)
67      canv.drawCentredString(0.5 * A4[0], 8.0 * inch, "AutoPenSploit")
68
```

Figura 19. Detalle creación portada informe (PDF_Report.py)

5.2.2.4. Ejecución de AutoPenExploit

La funcionalidad principal de la aplicación AutoPenExploit es la realización de las pruebas de penetración automatizada, para lo cual emplearemos las opciones del menú, primero introduciremos el dominio del análisis, en segundo lugar la ejecución (RUN) y por último la generación del Informe de auditoría.

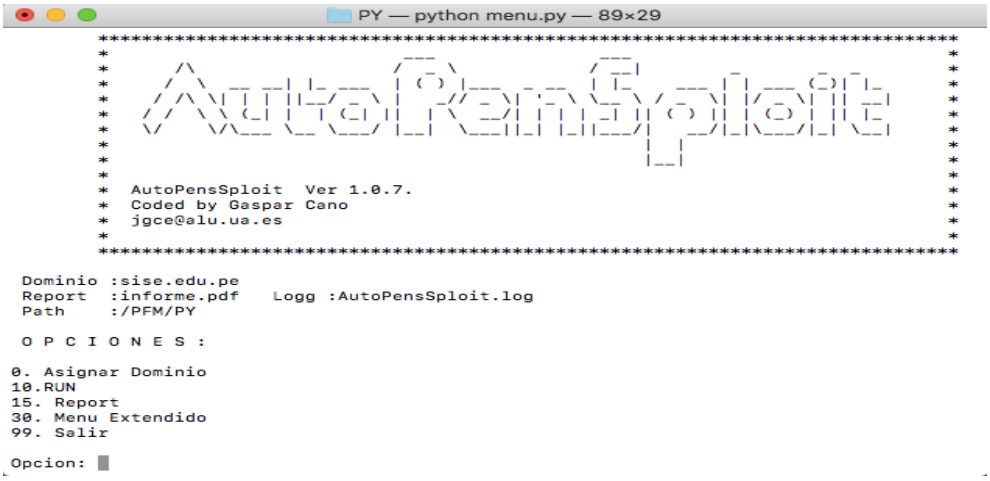


Figura 20. Menú Básico de AutoPenExploit

Existe un menú extendido, básicamente para tareas de depuración, que habilita la generación de todas las etapas por separado, como se puede apreciar en la figura siguiente:

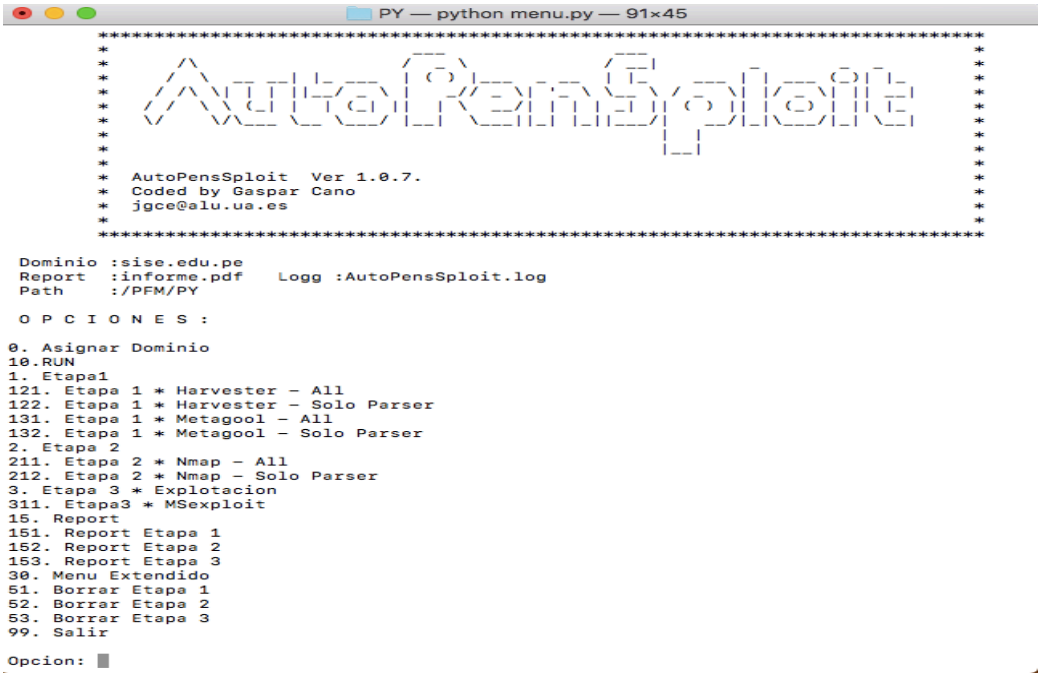


Figura 21. Menú Extendido de AutoPenExploit

5.2.2.5. Flujo de Ejecución

En la próxima figura se muestra el flujo de ejecución normal, de los componentes de las aplicaciones AutoPenExploit, y a continuación de esta se muestra como sería el flujo de ejecución con las inclusiones de nuevos módulos en desarrollos futuros.

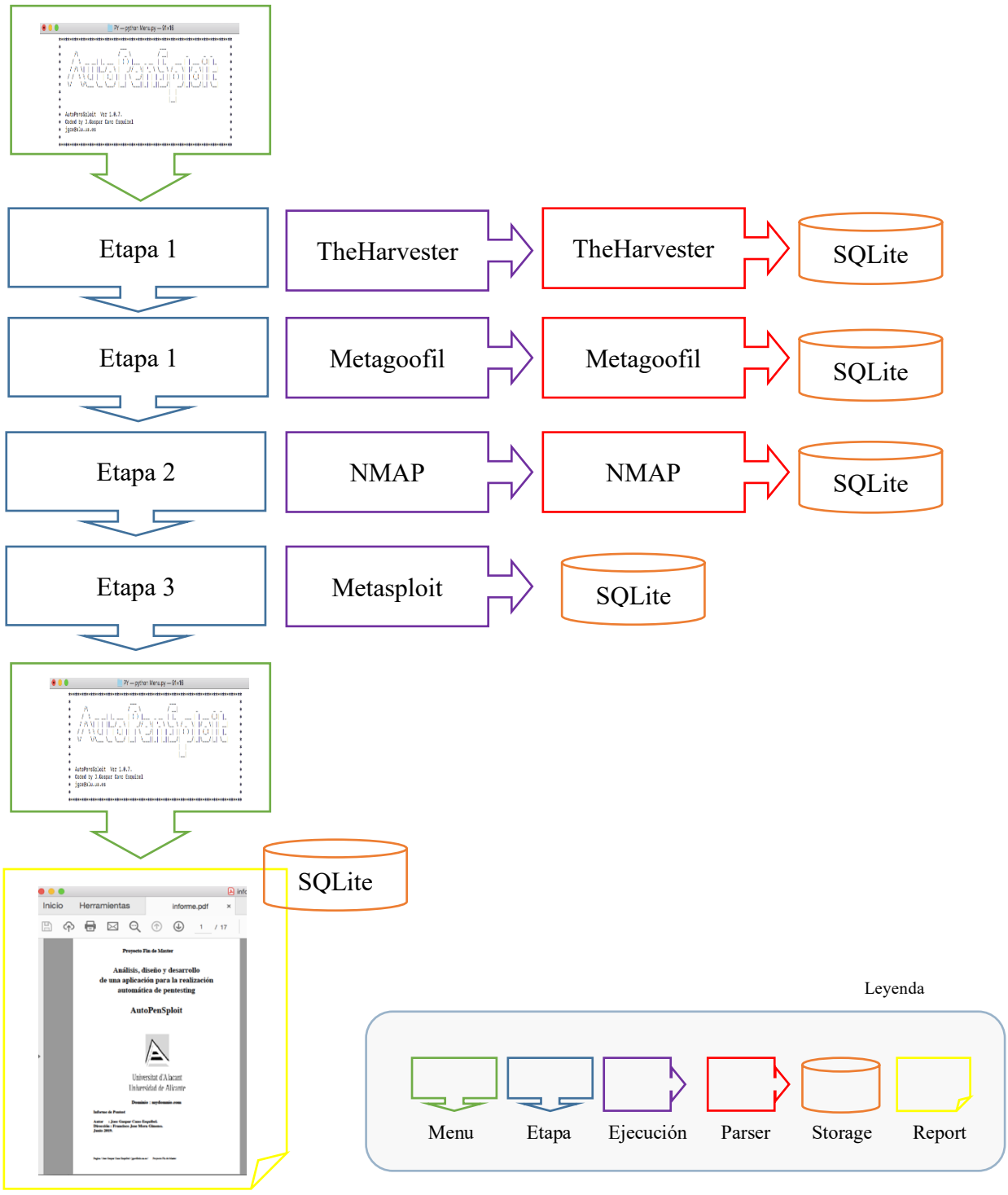


Figura 22.Flujo ejecución actual

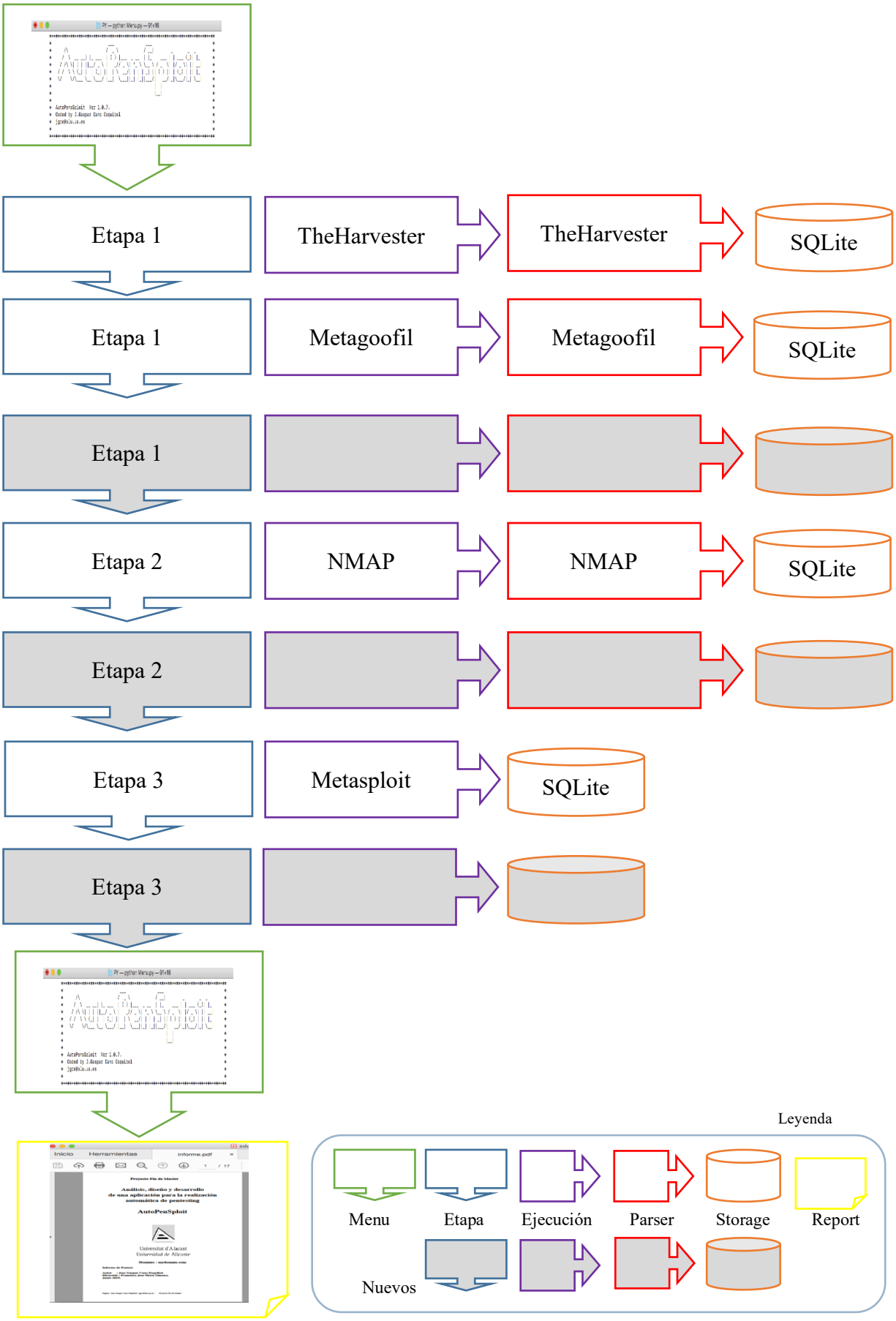


Figura 23.Flujo de ejecución previsto con nuevas herramientas

5.2.2.6. El Informe de AutoPenExploit

El informe del AutoPenExploit estaba marcado como un objetivo secundario, pero la presentación de este informe automatizado de vulnerabilidades detectadas, así como evidencia de las explotaciones realizadas en el sistema objetivo, permite cubrir la entrega de un informe final con las vulnerabilidades encontradas.

Las conclusiones, así como los puntos de mejora o soluciones a las deficiencias encontradas de estas pruebas de penetración, quedaran fuera del ámbito de este informe automático, ya que requieren una elaboración AdHoc que está fuera del ámbito de este proyecto. La siguiente figura describe la estructura del informe que se realiza, y en las siguientes subsecciones se detallan brevemente cada una de ellas.

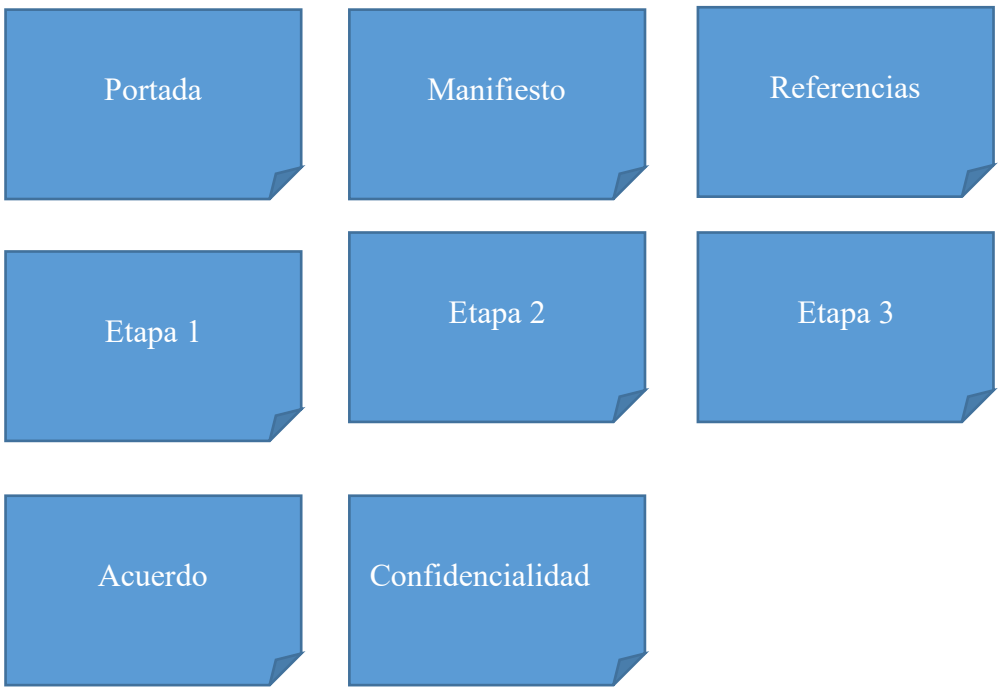
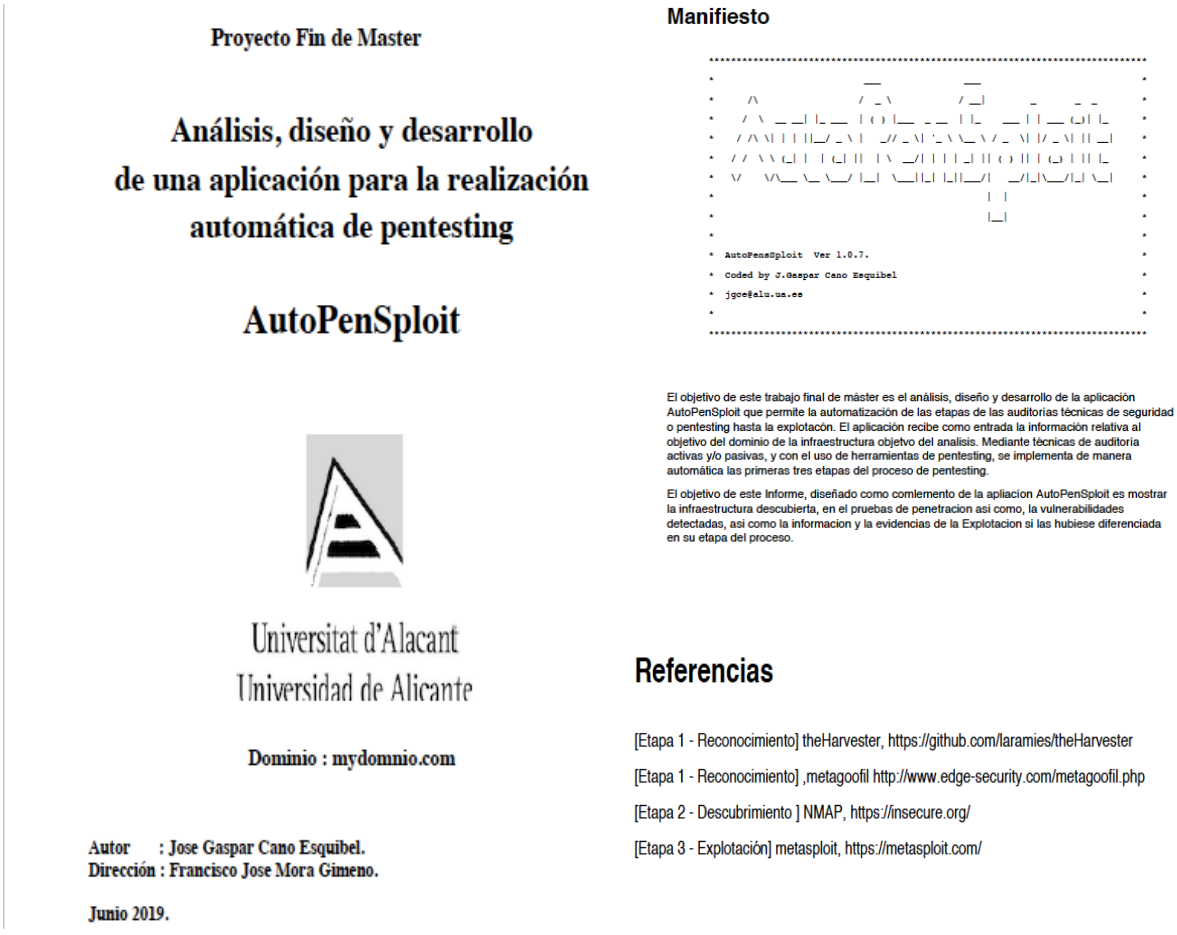


Figura 24. Composición del informe de AutoPenExploit

5.2.2.6.1. Portada, Manifiesto y Referencias

La portada del informe y el manifiesto, conforman la presentación y alcance del informe de penetración dentro de este proyecto, junto con las referencias que sirvan como información sobre las herramientas empleadas en la elaboración de este informe, como se muestra en detalle en la siguiente figura.



Proyecto Fin de Master

Análisis, diseño y desarrollo de una aplicación para la realización automática de pentesting

AutoPenSploit

Universitat d'Alacant
Universidad de Alicante

Dominio : mydomnio.com

Autor : Jose Gaspar Cano Esquibel.
Dirección : Francisco Jose Mora Gimeno.

Junio 2019.

Manifiesto

AutoPensSploit Ver 1.0.7.
Coded by J.Gaspar Cano Esquibel
jgce@alu.ua.es

El objetivo de este trabajo final de máster es el análisis, diseño y desarrollo de la aplicación AutoPenSploit que permite la automatización de las etapas de las auditorías técnicas de seguridad o pentesting hasta la explotacón. El aplicación recibe como entrada la información relativa al objetivo del dominio de la infraestructura objetvo del analisis. Mediante técnicas de auditoría activas y/o pasivas, y con el uso de herramientas de pentesting, se implementa de manera automática las primeras tres etapas del proceso de pentesting.

El objetivo de este Informe, diseñado como comlemento de la apliacion AutoPenSploit es mostrar la infraestructura descubierta, en el pruebas de penetracion asi como, la vulnerabilidades detectadas, asi como la informacion y la evidencias de la Explotacion si las hubiese diferenciada en su etapa del proceso.

Referencias

[Etapa 1 - Reconocimiento] theHarvester, https://github.com/laramies/theHarvester

[Etapa 1 - Reconocimiento] ,metagoofil http://www.edge-security.com/metagoofil.php

[Etapa 2 - Descubrimiento] NMAP, https://insecure.org/

[Etapa 3 - Explotación] metasploit, https://metasploit.com/

Figura 25. Detalle Portada, Manifiesto y Referencias

5.2.2.6.2. Etapa 1

La primera etapa el reconocimiento es donde se persigue recoger la mayor parte de información posible y que esta sirva de base para las siguientes etapas, siempre mediante técnicas no intrusivas. En el informe aparecerán en múltiples páginas, toda la información recopilada por las dos herramientas. En la siguiente figura mostramos detalle de la primera etapa

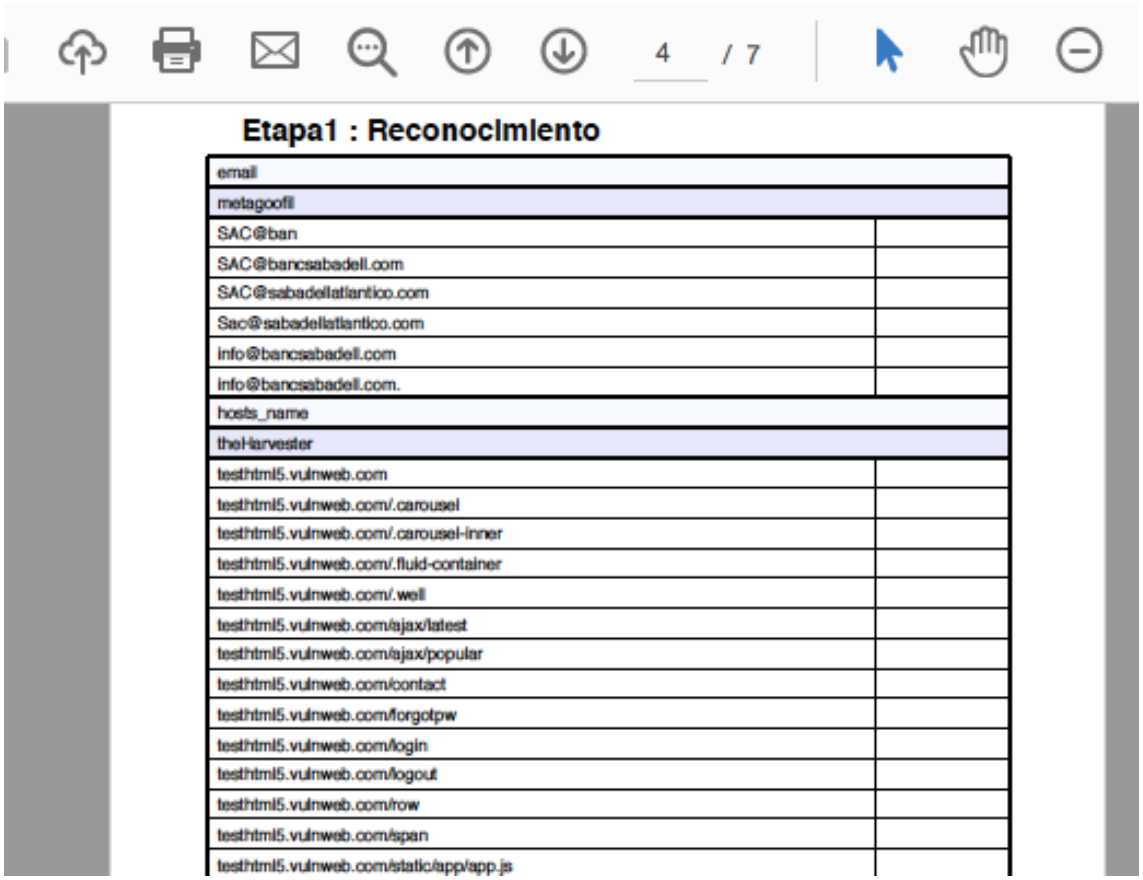


Figura 26 Detalle de la etapa 1 del Informe

En la figura anterior, se muestra por categoría (Email) y por herramienta de reconocimiento (metagoofil y theharvester) el detalle de la información recopilada, que podrá ser utilizada en las siguientes etapas de AutoPenExploit, o de forma manual con otras herramientas, o como auditoria para eliminar vulnerabilidades antes un ataque de ingeniería social para este ejemplo.

5.2.2.6.3. Etapa 2

Con la información proveniente de la fase anterior, que era un reconocimiento pasivo, la herramienta utilizada en esta etapa NMAP que nos proporciona información de infraestructura principalmente, puertos abiertos y versiones de sistemas operativo, mediante Figerprinting activo\ pasivo.

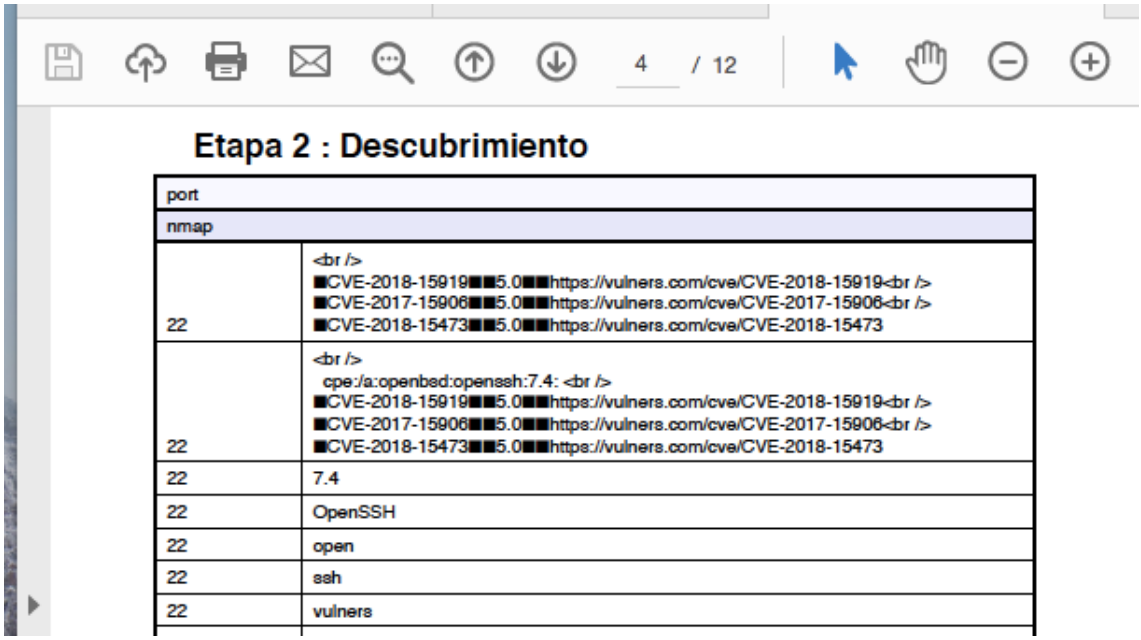


Figura 27 Detalle de la etapa 2 del Informe

Como se muestra en la figura anterior para la categoría (Port), a partir de la herramienta NMAP se muestra el estado de los puertos, por ejemplo, se describe el 22 como open. Al mismo tiempo asociado al puerto tenemos las vulnerabilidades mostradas con el vector estándar de la vulnerabilidad CVE (22), ya asociadas con la versión del sistema operativo reconocido. Esta información será necesaria para la siguiente etapa de explotación y como información de auditoría relevante asociada a vulnerabilidades detectadas, que deben ser mitigadas.

5.2.2.6.4. Etapa 3

En base a las vulnerabilidades descubiertas en la etapa anterior y en función de la información obtenida en las fases anteriores, se puede llevar a cabo la etapa de explotación. Metasploit Framework permite que se ejecute un payload asociado a vulnerabilidad descubierta.

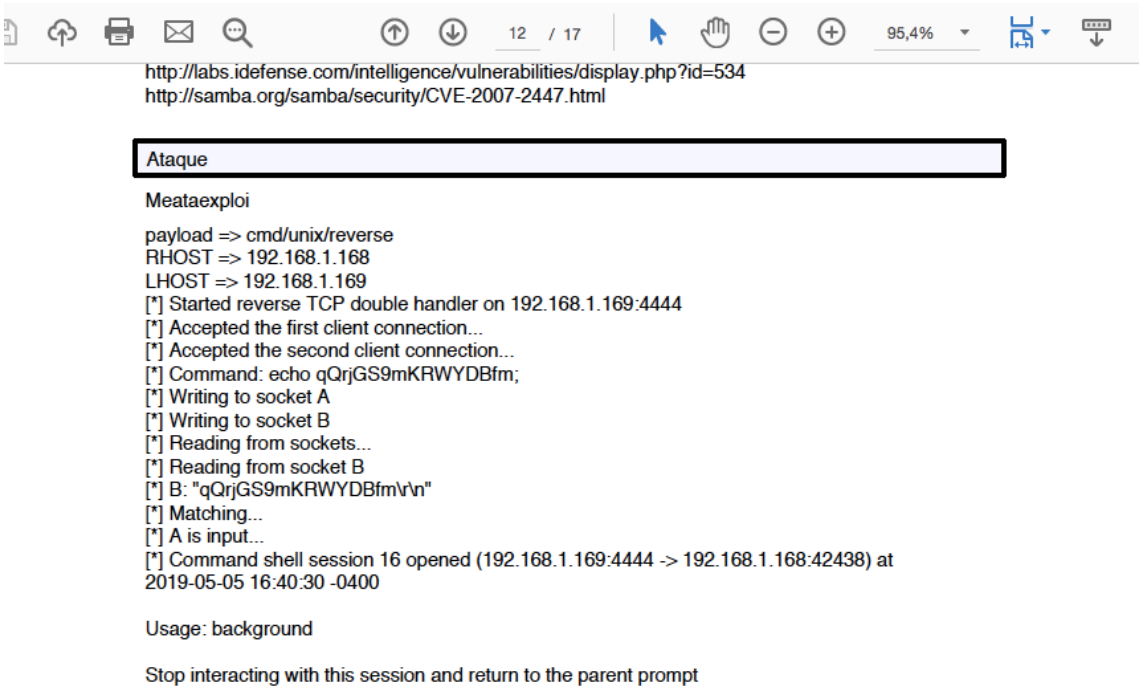


Figura 28 Detalle de la etapa 3 del Informe

La figura anterior muestra el ataque asociado a la vulnerabilidad detectada, asociada al vector de ataque, CVE-2007-2447 y sirve como evidencia real de que el exploit se puede ejecutar con éxito.

5.2.2.6.5. Manifiesto de autorización

La penúltima página del informe contiene la Autorización (permiso explicito) que supone el contrato de la empresa, que dará cobertura legal a las pruebas de penetración antes de poder iniciar cualquier actividad. En el contrato se define el ámbito y alcance de las pruebas y se especifican las acciones que están permitas y las que no, que es un requisito indispensable para toda prueba de hacking ético.

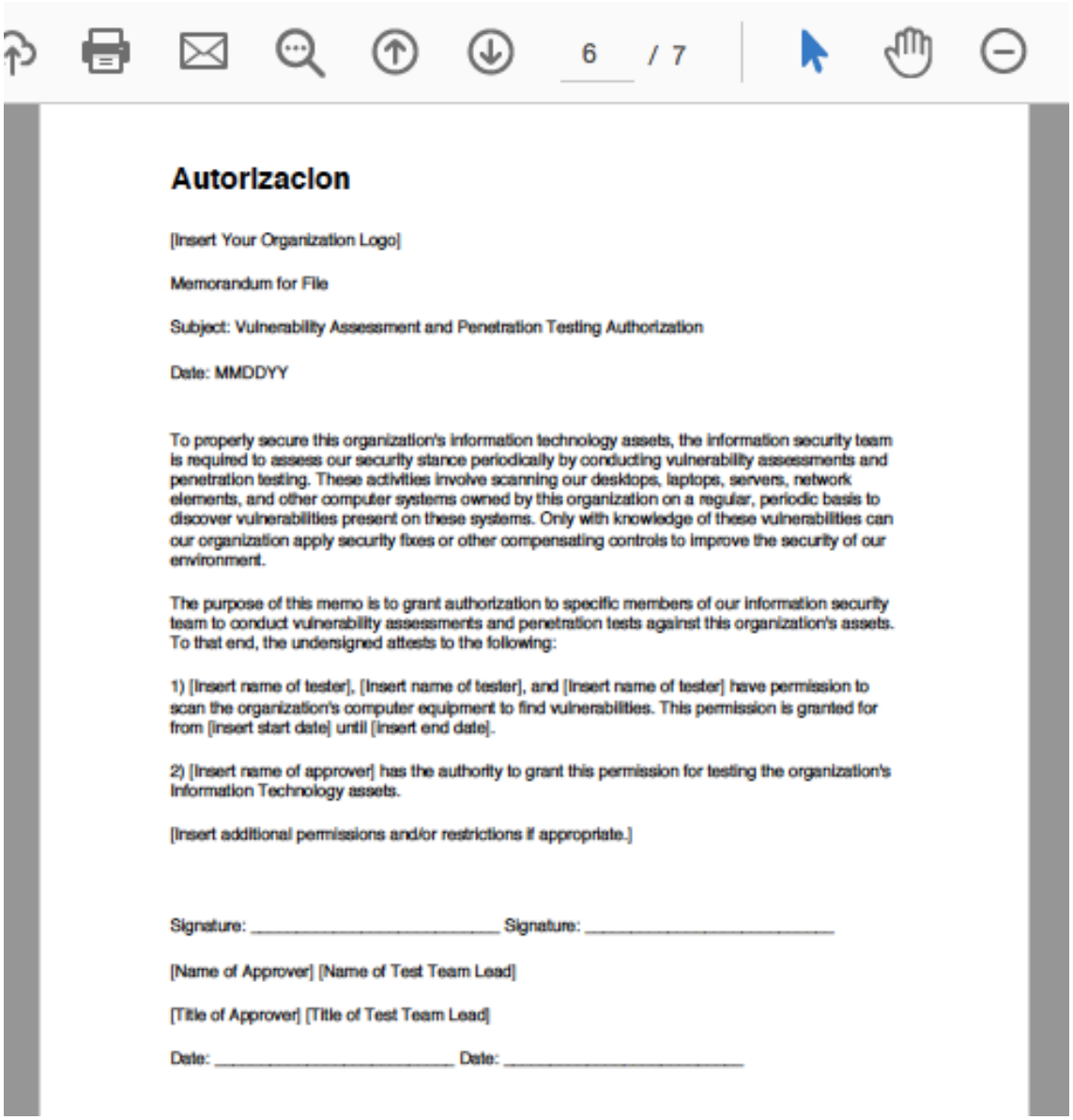


Figura 29 Detalle de la Autorización en el informe

5.2.2.6.6. Manifiesto de confidencialidad

Los resultados de las pruebas de penetración, así como cualquier tipo de información asociada a estas pruebas, pertenecen a la empresa. Por lo tanto, no podrá ser divulgada fuera de la organización que encargo las pruebas. En la siguiente figura se muestra el manifiesto de confidencialidad que se adjunta como última página en el informe generado, que es un requisito indispensable para todas pruebas de penetración.

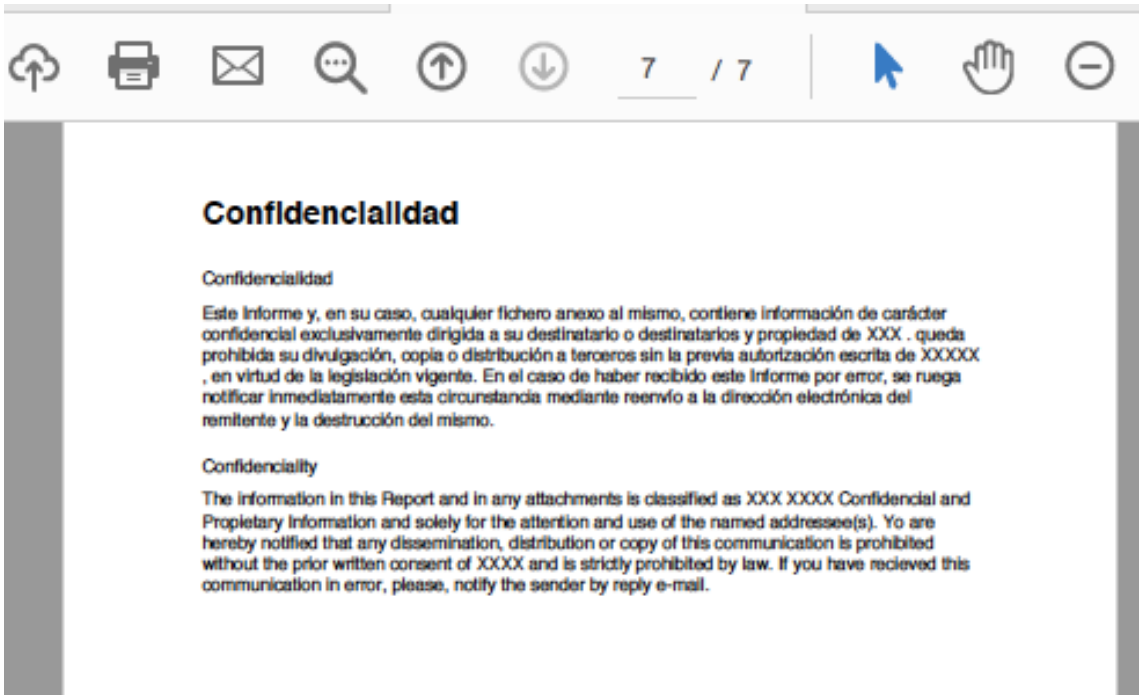


Figura 30 Detalle Confidencialidad en el Informe

6. Experimentación y casos de uso

Una vez descrita la aplicación vamos a incluir las pruebas realizadas para la aplicación desarrollada AutoPenExploit descrita en los apartados anteriores. El objetivo de este apartado es validar como se resuelve el proceso de automatización descrito.

Para la experimentación se han buscado diferentes entornos (casos de uso) tanto reales como de laboratorio, para poder realizar las validaciones de las diferentes etapas como se detalla la siguiente tabla de entornos:

Entorno		Ambiente	Descripción
	Test	Laboratorio	Test de Laboratorio Local
	Test	Real	Test Penetración
	Bank	Real	Site Bancario
	Edu	Real	Site Educacional
	Com	Real	Site Comercial
	Ocio	Real	Site de Ocio
	Trans	Real	Site de transporté

Tabla 16. Resumen de Entornos de explotación

La tabla siguiente describe el nivel de ejecución, es decir como se encadenan las etapas, el objetivo principal del proyecto como ya se ha reiterado es la ejecución automática (ligadas a la infraestructura a partir de un dominio de entrada), pero en todos los entornos no ha sido posible realizarla para todas las etapas por limitaciones éticas al no tener autorización expresa para poder realizar las pruebas de penetración, por lo que no avanzamos a las siguientes etapas en algunos entornos. Otras técnicas como la fuerza bruta para la búsqueda de contraseñas o la ingeniería social quedan fuera del alcance inicial del proyecto, esta tendría una continuidad manual con herramienta especificas, pero quedan expuestas como vulnerabilidades descubiertas en la tabla de resultados, los iconos facilitan la compresión de los mecanismos de ejecución empleados.

	Ejecución Manual
	Ejecución Automática (AutoPenExploit)
	Sin ejecución (No existe contrato)

Tabla 17. Niveles de ejecución de las etapas

Para cada etapa se han agrupado los elementos detectados, en función de su naturaleza como veremos en la tabla siguiente.

Elementos	
	email
	Nombres de usuarios (metadados en DOC, PDF)
	Organigrama CEO, CIO, empleados, centros)
	Infraestructuras (IP, Host, DNS)
	Vulnerabilidades
	Exploit (asociado a vulnerabilidad)

Tabla 18. Tabla de elementos detectados

La siguiente tabla muestra las amenazas detectadas, y el grupo de posibles ataques que podrían desencadenar un ataque.

	Ingeniería email
	Fuerza bruta
	Ingeniería social
	Búsqueda exploit
	Penetración Windows
	Penetración Linux

Tabla 19. Tabla de grupos de amenazas

Para cada entorno y cada etapa podrían considerarse una serie de riesgos de intrusión, en función de los elementos descubiertos en cada etapa, representado por colores como se muestra a continuación los riesgos y la etapa en la que aparecen.

Riesgos de intrusión		Etapa
	No detectado	1/2
	Detectado	1
	Alto	2

	Comprometido	3
	No comprometido	3

Tabla 20. Nivel de riesgo de intrusión

Como resumen de la experimentación realizada la ultima tabla muestra, para cada entorno y cada etapa, los elementos que podrían ser una amenaza y si se consiguió la explotación. Los iconos descritos en las anteriores tablas combinados con los colores de riesgo de intrusión nos ayudan a comprender la tabla de resultados y alcance de la experimentación.

Entorno y Etapa		Elementos Detectados		Amenaza		Siguiente Etapa	
Test 1	1		Email		N/A		N/A
			Nombre Usuarios		N/A		N/A
			Organigrama		N/A		N/A
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		Manual
			Vulnerabilidades		Exploit		Etapa 3
	3		Exploit Shellcode		Penetración		Persistencia
			Exploit Shellcode		Penetración		Persistencia
Test 2	1		Email		Ing. Email		Manual
			Nombre Usuarios		N/A		N/A
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		N/A
			Vulnerabilidades		Exploit		N/A
	3		Exploit Shellcode		Penetración		Manual
Bank 1	1		Email		Ing. Email		N/A
			Nombre Usuarios		Fuerza Bruta		N/A
			Organigrama		Ing. Social		N/A
			Infraestructura		Exploit		N/A

Bank 2 	1		Email		Ing. Email		N/A
			Nombre Usuarios		N/A		N/A
			Organigrama		Ing. Social		N/A
			Infraestructura		Exploit		N/A
Com 1 	1		Email		Ing. Email		Manual
			Nombre Usuarios		Fuerza Bruta		Manual
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		N/A
			Vulnerabilidades		Exploit		N/A
Com 1 	1		Email		Ing. Email		Manual
			Nombre Usuarios		N/A		N/A
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		N/A		N/A
			Vulnerabilidades		N/A		N/A
Ocio 1 	1		Email		Ing. Email		Manual
			Nombre Usuarios		N/A		N/A
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		N/A
			Vulnerabilidades		Exploit		N/A
Trans 1 	1		Email		Ing. Email		Manual
			Nombre Usuarios		Fuerza Bruta		Manual
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		N/A

			Vulnerabilidades		Exploit		N/A
Edu 1	1		Email		Ing. Email		Manual
			Nombre Usuarios		Fuerza Bruta		Manual
			Organigrama		Ing. Social		Manual
			Infraestructura		Exploit		Etapa 2
	2		Infraestructura		Exploit		N/A
			Vulnerabilidades		Exploit		N/A
Edu 2	1		Email		Ing. Email		N/A
			Nombre Usuarios		Fuerza Bruta		N/A
			Organigrama		Ing. Social		N/A
			Infraestructura		Exploit		N/A

Tabla 21. Resumen de Experimentaciones

7. Conclusiones y trabajo futuro

Como se indico en la motivación, la realización de las pruebas de penetración es un tema de máxima necesidad y actualidad, el objetivo ha sido crear un sistema automatizado que cubra las etapas de los pentesting hasta la explotación, mediante el uso de potentes herramientas Open Source existentes (las mismas que usan los atacantes maliciosos). Suponen una auditoria real de la exposición de vulnerabilidades, este objetivo ha sido cubierto. Del mismo modo también se ha cubierto la generación de un informe que conforme una auditoria que evidencie las vulnerabilidades descubiertas, así como los indicios reales de la etapa de explotación si las hubiese.

Como resumen de las conclusiones podemos destacar el cumplimiento de las siguientes características:

- **Validación**, del modelo automático desarrollado posibilita el encadenamiento entre las diferentes etapas de las pruebas de penetración.

- **Escalabilidad**, se pueden incrementar las funcionalidades a través de la introducción de nuevas herramientas en cada etapa

- **Independencia**, las herramientas utilizadas mantienen su ciclo de desarrollo y actualización independientemente de la aplicación de automatización.

- **Utilidad**, la generación del informe desde el descubrimiento hasta la explotación, audita la exposición real de los sistemas analizados y ayuda a su mitigación.

De la misma manera que se han cubierto los objetivos, como trabajo futuro quedaría la extensibilidad de nuevas funcionalidades, mediante la incorporación de nuevas herramientas (existen cientos de ellas disponibles y en nuevas en continuo desarrollo) que podríamos añadir en las diferentes etapas.

Se podrían incorporar funciones de planificación, alertas y distribución de informes, de forma que periódicamente se ejecute enviado los resultados a una o varias listas de distribución con la información obtenida.

La incorporación de funcionalidades de IA permitiría proponer mitigaciones a las vulnerabilidades encontradas, o que se propongan explotaciones en función de vulnerabilidades encontradas y aprendidas en otro sistema previamente, y que no han podido ser descubiertas en las etapas previas a la explotación en el sistema objetivo.

El perímetro de la Ciberseguridad es muy extenso, con una evolución vertiginosa por lo que queda mucho trabajo futuro por hacer.

Referencias

1. **Thomas Chen, Jean-Marc Robert.** The Evolution of Viruses and Worms. *Statistical Methods in Computer Security.* s.l. : CRC Press;, 2004.

2. **Infojobs.** guia-trabajar-ciberseguridad . 2018. https://orientacion-laboral.infojobs.net/guia-trabajar-ciberseguridad .

3. **Shodan.** www.shodan.io. *https://www.shodan.io/.* 2019.

4. **Maltego.** www.paterva.com. *https://www.paterva.com/web7/.* 2019.

5. **Pablo González, Germán Sánchez y Jose Miguel Soriano.** *Pentesting con Kali Linux.* s.l. : 0xWord, 2017.

6. **Sagar Rahalkar, Nipun Jaswal.** *Metasploit Revealed: Secrets of the Expert Pentester.* s.l. : Packt Publishing, 2017.

7. **Engebretson, Patrick.** *Patrick Engebretson The Basics Of Hacking And Penetration Testing, Second Edition.* ELSEVIER : s.n., 2013.

8. **David Kennedy, Jim O'gorman, Devon Kearns.** *Metasploit: The Penetration Tester's Guide* . s.l. : No Starch Press, 2011.

9. **de Raphaël Hertzog, Mati Aharoni, Jim O'Gorman.** *Kali Linux Revealed: Mastering the Penetration Testing Distribution.* s.l. : Offsec Press, 2017.

10. **Offensive Security.** https://www.kali.org/. 2019.

11. **Wikipedia/Kali.** https://es.wikipedia.org/wiki/Kali_Linux.

12. **Python Software Fundation.** 2019. https://www.python.org/.

13. **Washington, Sam.** *Learning Python Networking - Second Edition.* s.l. : Packt Publishing, 2019.

14. **Brandon Rhodes, John Goerzen.** *Foundations of Python Network Programming.* s.l. : Apress, 2014.

15. **Wikipedia / Python.** https://es.wikipedia.org/wiki/Python.

16. **TIOBE Index.** https://www.tiobe.com/tiobe-index/. 06 2019.

17. **SQLIte.** https://www.sqlite.org.

18. **Wikepiedia/SQLite.** https://es.wikipedia.org/wiki/SQLite.

19. **MsgPack.** https://msgpack.org/. https://msgpack.org/.

20. **https://en.wikipedia.org/wiki/MessagePack.** https://en.wikipedia.org/wiki/MessagePack. 2019.

21. **ReportLab.** https://www.reportlab.com/.
2019. https://www.reportlab.com/.
22. **tools.kali.org.** tools.kali.org. 2019.
23. **wingware.** 2019. https://wingware.com/.
24. **SQLite Studio.** 2019. https://sqlitestudio.pl/index.rvt.
25. **Lyon, Gordon "Fyodor".** Nmap network scanning. https://insecure.org/.
26. **Sagar Rahalkar, Nipun Jaswal.** *Metasploit Revealed: Secrets of the Expert Pentester.* s.l. : Packt Publishing, 2017.
27. **RAPID7.** Metasploit PRO - RPC API Guide. https://Metasploit.help.rapid7.com/docs/pro-api-methods-reference.
28. **January, Furqan Khan.** *Hands-On Penetration Testing with Python.* s.l. : Pack Publishing, 2019.
29. **Comun Vulnerabilities asn Exposures.** http://cve.mitre.org/.
30. **Ortega, Jose Manuel.** *Mastering Python for Networking an Security.* s.l. : Packt Publishing, 2018.

Glosario

ARM	Arquitectura de Microprocesadores (RISC)
BSD	Distribución de Unix.
Erlang	Lenguaje de programación orientado a la concurrencia.
GNOME	Entorno de Modelo de Objeto de Red GNU.
GNU	Proyecto de software libre (GNU is Not Unix).
Go	lenguaje concurrente, compilado inspirado en C.
GPL	Licencia Pública General de GNU.
JSON	Notación de Objetos de JavaScript.
KDE	Entorno de escritorio Linux (Kool DEsktop)
LGPL	Licencia Pública General Reducida de GNU.
Lisp	Lenguajes multiparadigma con notación polaca.
LXDE	Entorno de escritorio X11 ligero.
Modula-3	Lenguaje de programación imperativo, estructurado y modular.
PDP-11	Computadora fabricado Digital Equipment Corp
Perl	Lenguaje interpretado con características de C inferior.
Ruby	lenguaje interpretado, reflexivo y orientado a objetos.
Smalltalk	Lenguaje reflexivo, orientado a objetos y con tipado dinámico.
Swift	Lenguaje de programación multiparadigma para iOS y macOS.
X11	Sistema de gestor de ventanas X para Unix.
Xfce	Entono de escritorio Linux (XForms Common Enviroment).

Apéndice:

Detalle del menú de AutoPenExploit en Python

```
while not salir:

    mr = Ini.getMenuRuducido()

    print "        ********************************************************************************"
    print "        *                                                                              *"
    print "        *      /\                      /  _ \           / __|        _       _  _     *"
    print "        *     /  \  __ __| |_ ___  | ( ) |___  _ __  | |_    ___ | | ___ (_)| |_     *"
    print "        *    / /\ \| | | ||__/ _ \ |   _// _ \| '_ \ \__ \ / _  \| |/ _ \| || __|    *"
    print "        *   / /  \ \ (_| |  | (_| ||  | \  __/| | | | _| || ( ) || | (_) | || |_     *"
    print "        *   \/    \/\___ \__ \___/ |__|  \___||_| |_||___/|   __/|_|\___/|_| \__|    *"
    print "        *                                                 |  |                         *"
    print "        *                                                 |__|                         *"
    print "        *                                                                              *"
    print "        *  AutoPensSploit  Ver 1.0.7.                                                  *"
    print "        *  Coded by J.Gaspar Cano Esquibel                                             *"
    print "        *  jgce@alu.ua.es                                                              *"
    print "        *                                                                              *"
    print "        ********************************************************************************"
    print ""
    print " Dominio :" + Ini.getDominio()
    print " Report  :" + Ini.getInforme() + "   Logg :" + Ini.getLog()
    print " Path    :" + Ini.directorio
    print ""
    print " O P C I O N E S :"
    print ""
    print "0. Asignar Dominio"
    print "10.RUN"
    printfm (mr,"1. Etapa1  ")
    printfm (mr,"121. Etapa 1 * Harvester - All")
    printfm (mr,"122. Etapa 1 * Harvester - Solo Parser")
    printfm (mr,"131. Etapa 1 * Metagool - All")
    printfm (mr,"132. Etapa 1 * Metagool - Solo Parser")
    printfm (mr,"2. Etapa 2")
    printfm (mr,"211. Etapa 2 * Nmap - All")
    printfm (mr,"212. Etapa 2 * Nmap - Solo Parser")
    printfm (mr,"3. Etapa 3 * Explotacion")
    printfm (mr,"311. Etapa3 * MSexploit")
    print ("15. Report")
    printfm (mr,"151. Report Etapa 1")
    printfm (mr,"152. Report Etapa 2")
    printfm (mr,"153. Report Etapa 3")

    print ("30. Menu Extendido")

    printfm (mr,"51. Borrar Etapa 1 ")
```