UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Automatizando la resolución de problemas en competencias de Seguridad Informática

AUTORES: Basso Facundo – Pretto Jeremías

DIRECTOR: Venosa Paula

DIRECTOR: Lanfranco Einar

ASESOR PROFESIONAL: -

CARRERA: Lic. En Sistemas – Lic. En Informática

Resumen

Actualmente se llevan a cabo múltiples competencias internacionales de Seguridad Informática conocidas como Capture The Flag (CTF), donde distintos equipos distribuidos a través del planeta compiten resolviendo desafíos de distintas categorías. Nuestra motivación principal al ser participantes habituales de dichas competencias, consiste en generar herramientas las cuales automaticen las tareas repetitivas que suelen requerirse para la resolución de dichos desafíos persiguiendo la idea de reutilizar las soluciones o mecanismos ya empleados y evitando de esta forma "reinventar la rueda" en cada nuevo evento. La tesina se basa en el desarrollo de dos herramientas: RSolver, que ayuda a automatizar la resolución de desafíos de criptografía RSA. Y SYPER CTF Tools, la cual provee una plataforma web para equipos de CTF que permite disparar una batería múltiples herramientas de forma intuitiva y amigable.

Palabras Clave

Capture The Flag, CTF, automatización, RSA, DRY, criptografía, esteganografía, RSolver, SYPER CTF Tools, Seguridad Informática, Jeopardy, Ataque-Defensa, explotación web, plaintext, ciphertext, algoritmos asimétricos, clave privada, algoritmos de factorización, ataque de Wiener, ataque de Hastad, ataque de texto plano común, ataque de texto plano relacionado, ataque a instancias con exponente público bajo, ataque de mensaje estereotipado, esteganografía en bit menos significativos, OWASP Top Ten.

Conclusiones

Gracias al uso de las herramientas descritas en esta tesina, hemos podido lograr una significativa optimización del uso del tiempo en competencias del tipo CTF, el cual era nuestro objetivo principal al desarrollarlas. También logramos facilitar el aprendizaje de ataques al protocolo RSA ya que la herramienta además de automatizar la resolución de los problemas hace hincapié en el aspecto formativo de los mismos. Por otra parte estas herramientas favorecen la integración de nuevos participantes ya que permiten un uso fácil e intuitivo.

Trabajos Realizados

Desarrollo de la herramienta RSolver para automatizar la resolución de problemas frecuentes en CTF de criptografía RSA.

Desarrollo de la herramienta SYPER CTF Tools para integrar y facilitar el uso mediante una interfaz web de una batería de aplicaciones CLI usualmente utilizadas en CTF.

Trabajos Futuros

RSolver
- Agregar nuevos ataques RSA
- Soporte multithreading de los ataques
- Agregar soporte AES, XOR, GPG
- Migración scripts Sage a Python

Syper CTF Tools
- Destacar ejecuciones mediante tags
- Filtrado de tags
- Incorporar nuevas herramientas
- Mejorar mecanismo de actualización

Fecha de la presentación: Diciembre 2019

Automatizando la resolución de problemas en competencias de Seguridad Informática.

Por

Facundo Basso
Jeremías Pretto

Facultad de Informática
Universidad Nacional de La Plata

Directores: Einar Lanfranco. Paula Venosa

Tesina de Licenciatura en Informática / Licenciatura en Sistemas

Diciembre 2019

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Tabla **Página**

CAPÍTULO 1

MOTIVACIÓN

En la actualidad se realizan múltiples competencias internacionales de Seguridad Informática conocidas como Capture The Flag (Captura la bandera), donde distintos equipos a lo largo y ancho del planeta compiten resolviendo desafíos. Dichas competencias están diseñadas como ejercitación educativa para dar a los participantes la experiencia de atacar y/o mejorar la seguridad de un sistema a como ocurre en el mundo real.

Estos eventos están orientados para principiantes como a expertos de todo el mundo y abordan una amplia gama de aspectos y temas: criptografía, análisis binario, ingeniería inversa, esteganografía, explotación web, seguridad de dispositivos móviles, entre otros. Desde el primer CTF (Capture The Flag[1]) realizado dutante la conferencia de seguridad más grande de EEUU, la DEFCON de 1996, las competencias de CTF han tenido un gran desarrollo y crecimiento global, debido a que suelen realizarse en forma remota a través de Internet.

La popularidad de los eventos CTF ha crecido considerablemente, más allá incluso de las instituciones educativas, por tal motivo los eventos en la actualidad son impulsados tanto por organizaciones de Seguridad Informática como por entusiastas. En 2018 se realizaron 122 competencias[1], más de dos competencias por semana.

Si bien se han realizado trabajos sobre los aspectos pedagógicos de las competencias, hay poco estudio sobre cómo se juega realmente. Es decir, qué acciones y estrategias son o serán tomadas por los participantes durante los eventos.

[1] A partir de ahora CTF

Los CTFs suelen tener una duración de entre 8 horas a 2 días, lo que significa una optimización del tiempo al máximo. Nuestra motivación principal al ser participantes habituales de dichas competencias, consiste en generar herramientas open source y colaborativas las cuales automaticen las tareas repetitivas que suelen requerirse para la resolución de retos. Además, el software está destinado a ayudar a cualquier participante, nuevo o experimentado.

Objetivo

El objetivo principal de esta tesina es el desarrollo de dos sistemas de software que permiten automatizar las tareas repetitivas que suelen realizarse al participar en competencias de seguridad informática.

Persiguiendo la idea de reutilizar las soluciones o mecanismos ya empleados y evitando de esta forma "reinventar la rueda" en cada nuevo evento.

Dichos sistemas son:

- RSolver
- SYPER CTF Tools

Los retos de criptografía son muy comunes durante las competencias de seguridad tipo CTF. Muchos de ellos consisten en explotar una vulnerabilidad en un sistema de cifrado mal implementado. Estas vulnerabilidades permiten recuperar el mensaje cifrado, que por lo general es la flag, sin necesidad de conocer de antemano la clave secreta con la que se cifró. A menudo, el flujo de trabajo para resolver estos problemas implica averiguar qué vulnerabilidad está presente, investigar cómo explotar esa vulnerabilidad y escribir un programa o script que implemente el ataque. Cada uno de estos pasos puede tomar una cantidad significativa de tiempo, además del tiempo dedicado a depurar la implementación del exploit.

Nuestro primer objetivo es el desarrollo de RSolver, una herramienta que reduce ampliamente el tiempo empleado en cada uno de estos pasos. En él se encuentran varios ataques usados para vulnerar malas implementaciones del protocolo RSA, como el ataque de Wiener, el ataque de difusión de Hastad, el uso de primos de Fermat, el ataque de módulo común, entre otros. Además, se incluyen varios métodos de factorización en números primos, cada uno con sus propios casos de uso especiales. Si nuestra herramienta ya implementa el exploit requerido para resolver el ejercicio, el tiempo necesario para investigar, codificar y depurar una solución se reduce significativamente. Además, un

problema de RSA puede ser vulnerable a más de un ataque, lo que puede ser desconocido tanto para el usuario como para el autor del problema. Por lo tanto, RSolver también se puede utilizar para probar la corrección de los desafíos de CTF, reduciendo así el tiempo requerido para implementar los scripts de solución. En resumen, RSolver realiza todas las acciones necesarias en este tipo de ejercicios: busca qué vulnerabilidad está presente, la explota, y consigue el texto plano (que suele ser la solución).

En las competencias de seguridad, además de criptografía, existen otras categorías como web y esteganografía. El segundo objetivo es el desarrollo de una aplicación para equipos de CTF la cual permite el uso fácil e intuitivo de distintas herramientas de línea de comando mediante una interfaz web. SYPER CTF Tools es multiusuario y permite ejecutar tareas concurrentemente, donde cada miembro del equipo puede ver los resultados de ejecuciones anteriores de sus compañeros, evitando así, repetir tareas realizadas previamente por otro. Al ser una aplicación web, no requiere ninguna instalación del lado cliente, puesto que las herramientas se ejecutan directamente en el servidor.

Organización del documento

La presente tesina está dividida por capítulos y cada uno aborda las diferentes etapas, el marco teórico y el desarrollo. A continuación se detalla de forma concisa la estructura y los contenidos de cada capítulo.

Capítulo 2, Marco teórico Explicaremos las competencias de seguridad estilo Capture The Flag, y luego, los conceptos y definiciones básicas de tres categorías de estas competencias: Criptografía, esteganografía y Web. Dado nuestro desarrollo de la herramienta de criptoanálisis RSolver, haremos especial énfasis en la sección Criptografía, y dentro de ésta, en el algoritmo RSA y sus vulnerabilidades.

Capítulo 3, Rsolver Este capítulo está dedicado a nuestro primer objetivo que consiste en el desarrollo de la herramienta de criptoanálisis RSolver. Visibilizaremos allí otras aplicaciones similares existentes, además de exponer el énfasis y desarrollo que le damos a la herramienta. Concluiremos con los resultados obtenidos, basándonos en la eficacia contra otras herramientas.

Capítulo 4, SYPER CTF Tools En el capítulo 4 expondremos nuestro segundo objetivo, el desarrollo de una aplicación web para equipos de CTF, que permita la ejecución de herramientas de línea de comando de manera simple e intuitiva mediante una interfaz web. Veremos las tecnologías utilizadas y las herramientas útiles para retos de criptografía, esteganografía y de explotación web a incorporar en este sistema.

Capítulo 5, Conclusiones y Trabajo futuro En este último capítulo presentaremos las conclusiones del trabajo y las proyecciones futuras.

CAPÍTULO 2

MARCO TEÓRICO

2.1 Capture The Flag

2.1.1 Definición

Los ejercicios y competencias del tipo CTF son eventos que se realizan por lo general en el contexto de conferencias de seguridad informática. Se le ofrece a los participantes la posibilidad de poner en práctica las habilidades comprendidas en las distintas facetas del campo de la Seguridad Informática con el objetivo de recuperar banderas o *Flags* . Las cuales se pueden entregar a los organizadores del concurso para demostrar que se ha resuelto un desafío, problema o se ha logrado un objetivo particular.

2.1.2 Historia

La DEFCON es la conferencia de ciberseguridad más grande de los EEUU dando sus inicios en 1993. El primer CTF fue desarrollado y hosteado en 1996 durante la DEFCON realizada en Las Vegas, Nevada. Este encuentro se transformó en una plataforma para poner a prueba habilidades de hacking y a medida que Internet fue creciendo, tanto la DEFCON como las competencias CTF lo hicieron también.

El DEFCON CTF se ha vuelto tan popular que cientos de equipos alrededor del mundo intentan clasificar y asi participar cada año. La mayoría de estos equipos dedican meses a su preparación con la expectativa de ganar un viaje a Las Vegas. Durante el

2013, dos tercios de los equipos participantes eran equipos internacionales, incluidos equipos de Japón, Corea del Sur, China, Rusia y equipos mixtos europeos.

A partir de la primera DEFCON se han desarrollado y organizado otros CTF. La mayoría de estos eventos se encuentran abiertos o disponibles a instituciones académicas mayoritariamente. De hecho la ICTF (International CTF) organizada por la Universidad de California, Santa Bárbara, es una de las más conocidas. Se ha convertido en el CTF de Ataque/Defensa más grande en la actualidad, donde han llegado a participar más de 90 equipos de todo el mundo.

2.1.3 Clasificación

Podemos decir que existen dos categorías de CTF: a) Tipo "Ataque/Defensa" y b) Tipo "Jeopardy".

En un CTF del tipo "Ataque-defensa" los organizadores proporcionan a cada equipo imágenes idénticas y pre configuradas de un servidor, con software personalizado desarrollado por ellos. El servidor de cada equipo estará conectado a una red aislada y dedicada para la competencia. Cada equipo deberá administrar y defender su propio servidor, mientras que de forma simultánea intenta penetrar las defensas creadas por los otros equipos para capturar las banderas, las cuales son entregadas a los organizadores a cambio de puntos. Intencionalmente las reglas no son muy restrictivas para no limitar la creatividad de cada equipo al momento de construir Ataques y Defensas novedosas y eficaces.

Con el fin de evitar que los equipos "apaguen" su software vulnerable como medio para defenderlo, los equipos generalmente reciben una calificación de su capacidad para continuar brindando esos "servicios" al público, lo que lleva a los organizadores a probar periódicamente si el software de cada equipo sigue siendo accesible tanto para los organizadores como para los otros equipos.

Uno de los objetivos consiste en que cada equipo pueda solucionar cualquier defecto que hallen en el software asignado en lugar de cerrarlo como alternativa. A medida que los equipos encuentran fallas en su propio *software*, van comprendiendo como son vulnerables ante cualquier ataque, además de advertir que los demás equipos participantes también son vulnerables al mismo tipo ataque. Por lo tanto cada equipo deberá aprovechar cada defecto o vulnerabilidad para penetrar en los servidores de su oponente y de este modo, tras una infiltración exitosa, recuperar/adquirir una bandera del oponente. Las banderas son colocadas por los organizadores en el servidor de cada

equipo y son reemplazadas periódicamente para proporcionar nuevas banderas que deberán ser capturadas durante el juego/competencia.

La modalidad en la rotación periódica de las banderas obliga a los equipos a demostrar que pueden mantener el acceso a los servidores de sus oponentes. Además de observar y percibir la evolución gradual de la defensa como del ataque durante la progresión del juego.

La segunda categoría de CTF tipo "Jeopardy" consiste en resolver problemas para obtener puntos. En este sentido los organizadores realizan una serie de desafíos relacionados con seguridad y los ponen a disposición de los participantes para ser resueltos. En esta modalidad los participantes no interactúan entre sí. Sin embargo, los equipos sí competirán para ser los primeros en resolver el problema y acumular la mayor cantidad de puntos.

La infraestructura para albergar una competencia CTF de este estilo se parece a un sitio web tradicional en el que se publican los objetivos. Esto permite una mayor facilidad durante la organización, además de un mayor acceso a la participación de un mayor número de equipos. En diversas ocasiones se ha contado con la participación de 500 o más equipos de manera simultánea.

A menudo las competencias de este estilo ofrecen una variedad de desafíos a través de los cuales se promueve y requiere un mayor despliegue de habilidades relacionadas con seguridad que los eventos de tipo Ataque/Defensa.

En las conferencias de seguridad informática más conocidas del mundo, se suele organizar vía on-line, con registro abierto, un CTF estilo "Jeopardy" como etapa clasificatoria, el cual invita a los que tuvieron mejor desempeño, a asistir al evento para competir en un CTF final reducido que se lleva a cabo de forma presencial durante el mismo.

Cómo experiencia personal, de 2016 a 2019, clasificamos por este medio a las finales presenciales de Cybercamp [2], organizado por el Instituto Nacional de Ciberseguridad de España. También clasificamos como equipo a las finales presenciales de Volga CTF 2017 (Rusia), Ekoparty Pre CTF (Argentina) y Affinity CTF 2019 (Polonia).

Los CTF se han vuelto tan populares que es posible encontrar un CTF de un tipo u otro durante cada semana del año. De hecho toda una comunidad ha surgido y es "*trackeada*" por sitios como ctftime.org, que ofrece un calendario completo de eventos, seguimiento de resultados y clasificación de equipos. El sistema de clasificación destaca particularmente tanto la popularidad de los CTF como la naturaleza cada vez más competitiva de los mismos.

2.1.4 Categorías

Los problemas a resolver en un Capture The Flag pertenecen a una de las siguientes categorías o una combinación de las mismas:

- Explotación web: Estos desafíos consisten en hallar y aprovechar las vulnerabilidades presentes en los sitios o aplicaciones Web. El creador del reto esconde una Flag en el servidor que hostea la Web, mientras el competidor debe aprovechar alguna vulnerabilidad para intentar acceder a la misma.

- Criptografía: Estos desafíos se focalizan en descubrir y usar vulnerabilidades presentes en un protocolo criptográfico. Los desafíos de este tipo suelen ser los más exigentes en cuanto al conocimiento teórico. Muchas veces se proporciona a los participantes una muestra de información cifrada, la cual se debe descifrar encontrando el algoritmo criptográfico utilizado y aplicando criptoanálisis para detectar la falla en la utilización del mismo.

- Esteganografía: Es el arte de ocultar un archivo de texto, imagen, video o audio dentro de otro archivo, llamado portador, de modo que no sea perceptible su existencia. Los desafíos por lo general consisten en una imagen que aparentemente no contiene nada interesante. Pero, de hecho, la imagen puede contener la Flag del desafío. Mediante filtros y algoritmos aplicados a la imagen, el jugador debe ser capaz de develar la Flag oculta.

- Explotación binaria: Consiste en generar que una aplicación actúe de manera diferente a la forma en que había sido diseñada para ser ejecutada. Al hacer que la aplicación se ejecute de manera distinta, se está obteniendo información valiosa que puede ser utilizada para alterar o controlar el sistema objetivo. Se suele manejar técnicas de corrupción de memoria, las cuales permiten a un atacante obtener privilegios no autorizados por el sistema que ejecuta la aplicación; o bien secuestrar el flujo de control de la aplicación e inyectar sus comandos directamente en el sistema. En estos desafíos el objetivo consiste en crear un Exploit para un binario ejecutable, que aproveche alguna vulnerabilidad y recupere de este modo la *flag*.

- Ingeniería inversa: Los desafíos de ingeniería inversa requieren el conocimiento de saber utilizar un *debugger* y *software* de desensamblado. El objetivo consiste en tomar un binario compilado, desensamblarlo y descubrir cómo funciona. Se debe estar familiarizado con la forma en que la aplicación utiliza el flujo de control, los bucles y las

condiciones para que pueda descubrir como ejecutar el programa a nuestra voluntad, y luego así, capturar la *flag*.

- Forense: El objetivo principal en esta categoría es "investigar" algún tipo de datos estático, como una captura de red, un archivo de volcado de memoria o una imagen de disco y encontrar la *flag* oculta.

En los siguientes capítulos nos introduciremos más detalladamente a las categorías de CTFs: Criptografía, Esteganografía y Explotación web.

2.2 Criptografía

2.2.1 Introducción

La criptografía es una herramienta indispensable utilizada para proteger la información presente en los Sistemas Informáticos. Se utiliza a todo momento por millones de usuarios a lo largo y ancho del planeta.

Los sistemas criptográficos son una parte integral de los protocolos estándar, siendo el protocolo TLS el más usado de ellos, haciendo que sea relativamente fácil incorporar cifrado sólido en una amplia gama de aplicaciones. Aunque si bien es extremadamente útil, la criptografía también es altamente quebrantable. De hecho el sistema criptográfico más seguro puede ser completamente inseguro por una sola especificación errónea u error de programación.

En las competencias de seguridad informática, el objetivo de los retos de criptografía consiste en recuperar el texto plano a partir de un mensaje cifrado.

2.2.2 Definición

La criptografía es una ciencia que utiliza métodos y técnicas con la meta principal de cifrar y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando una o más claves.

2.2.3 Conceptos básicos

Al mensaje que se va a cifrar se lo denomina *Plaintext* o Texto plano.

El cifrado es el proceso de 'disfrazar' el mensaje para ocultar su contenido.

Un Texto Plano que ha sido cifrado se denomina *Ciphertext* o Texto cifrado

El proceso de convertir un Texto Cifrado a Texto Plano se lo denomina Descifrado.

El arte y ciencia que estudia los métodos para conservar el envío y recepción de mensajes de manera segura se la conoce como Criptografía. Es practicada y desarrollada por criptógrafos. Los criptoanalistas practican el criptoanálisis que consiste entonces en vulnerar el texto cifrado.

El Texto Plano se lo indica con la letra "M" y éste puede ser: un flujo de bits, un archivo de texto, una imagen, vídeo o cualquier cosa. Sin embargo en lo que respecta a una computadora, "M" puede significar o simbolizar datos binarios.

El Texto Cifrado se expresa con la letra "C". Si bien al igual que "M" son datos binarios, "C" puede tener el mismo tamaño que "M" o incluso ser mayor.

La función de Cifrado "E" transforma "M" para producir C:

$$E(M) = C$$

En el proceso inverso, la función de Descifrado "D" transforma "C" para producir de este modo a "M":

$$D(C) = M$$

Además de proporcionar confidencialidad, a menudo se pide a la criptografía que garantice:

- **Autenticación** Posibilitar al receptor de un mensaje la determinación de su origen. Es decir, un intruso no debería poder hacerse pasar por otra persona.
- **Integridad** Posibilitar al receptor de un mensaje la verificación de que éste no se haya modificado durante el tránsito. Es decir, un intruso no debería poder sustituir un mensaje falso por uno legítimo.
- **No repudio** Imposibilitar que un remitente pueda negar falsamente y tardíamente haber enviado un mensaje.

Un algoritmo criptográfico es la función matemática utilizada para realizar el cifrado y descifrado. Vale aclarar que por lo general hay dos funciones, una utilizada para cifrar y otra para descifrar.

Los algoritmos criptográficos modernos para sus operaciones de cifrado y descifrado hacen uso de una clave o Key expresada por “K”. La seguridad de un algoritmo se basa en esta clave. Es decir, un algoritmo puede ser publicado y analizado, pero al no poseer o conocer la clave el mensaje no podrá ser descifrado. El rango de valores que puede tomar la clave es llamado Keyspace o Espacio de claves. Por lo general encontramos dos tipos de algoritmos basados en clave: a) Simétricos o de clave privada y b) Asimétricos o de clave pública

2.2.4 Algoritmos simétricos

En un algoritmo de cifrado simétrico (por ejemplo DES o AES) el emisor y el receptor deben tener una clave compartida y configurada previamente. Además de conservarla en secreto. De esta manera el remitente puede usar la clave para el cifrado, y el receptor usarla para el descifrado. El principal problema con los sistemas de cifrado simétrico

no está ligado a su seguridad, sino al intercambio/distribución de las claves. Una vez que el remitente y el destinatario hayan intercambiado las claves pueden usarlas para comunicarse con seguridad. Sin embargo, ¿qué canal de comunicación seguro se ha utilizado para ser transmitidas las claves? Sería mucho más fácil para un atacante intentar la intercepción de una clave antes que probar las posibles combinaciones dentro del Espacio de Claves.

El Cifrado y Descifrado con un algoritmo simétrico se denota como:

$$Ek(M) = C$$

$$Dk(C) = M$$

2.2.5 Algoritmos asimétricos

En un algoritmo de cifrado asimétrico (por ejemplo, RSA) se utilizan un par de claves: a) La clave pública y b) la clave privada.

Las claves públicas pueden darse a conocer libremente a otras personas, mientras que las claves privadas sólo deben ser conocidas por su propietario. Lo que se cifra con la clave pública, se descifra con la clave privada correspondiente. Del mismo modo, lo que se cifra con la clave privada, se descifra con la clave pública. Si una persona que emite un mensaje a un destinatario usa la clave pública del destinatario para cifrar el mensaje, sólo el destinatario al usar la clave privada podrá descifrar dicho mensaje.

El mensaje sólo podría ser descifrado por el destinatario dado que es el único que debería conocer la clave que se necesita. Esto permite garantizar la confidencialidad del mensaje. Nadie salvo el destinatario puede descifrarlo.

En este sentido el mensaje sólo podría ser descifrado por el destinatario dado que es el único que debería conocer la clave que se necesita para ello. Esto garantiza la confidencialidad del mensaje, donde nadie salvo el destinatario puede descifrarlo.

Cualquiera usuario al utilizar la clave pública del destinatario puede cifrar mensajes que solo pueden ser descifrados por el destinatario utilizando su clave privada. Si el propietario del par de claves usa su clave privada para cifrar un mensaje, cualquier usuario podrá descifrarlo utilizando la clave pública del primero. En este caso se consigue certificar el origen del mensaje, puesto que si todos lo podemos descifrar usando una clave pública, el origen es quien tiene la clave privada correspondiente. Esta idea es el fundamento de lo que se conoce como Firma Digital.

Los sistemas de cifrado asimétricos permiten abordar el problema de la distribución de la clave dado que no requieren de un intercambio de claves como sí ocurre en los sistemas de cifrado simétricos.

La seguridad de los sistemas de Cifrado Asimétrico reside en la dificultad computacional para descubrir la clave privada a partir de la pública. La generación de un nuevo par de claves (Clave pública y la correspondiente Clave privada) utilizan funciones matemáticas que son computacionalmente viables para la generación de ambas claves, pero las propiedades matemáticas de las mismas hace que sea muy difícil o computacionalmente imposible calcular la Clave privada a partir de la Clave pública.

El cifrado al utilizar la clave pública "K" se expresa:

$$Ek(M) = C$$

Aunque la Clave pública y la Clave privada son diferentes, el descifrado con la clave privada correspondiente se expresa:

$$Dk(C) = M$$

2.2.6 RSA

2.2.6.1 Introducción

Es un sistema criptográfico asimétrico desarrollado en 1977 por Ron Rivest, Adi Shamir y Len Adleman [3]. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar mensajes digitales.

Es utilizado para asegurar el trafico web en navegadores y servidores web, para garantizar la privacidad y autenticidad del correo electrónico, para asegurar sesiones de conexión remota y es el núcleo de los sistemas de pago electrónicos con tarjeta de crédito.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros muy grandes. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores. Como en todo sistema asimétrico, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

2.2.6.2 Operación

El algoritmo RSA involucra tres pasos: generación de claves, cifrado y descifrado

Generación de claves

Las claves son generadas de la siguiente manera:

1. El usuario Alicia genera dos primos "grandes" p y q, usando alguno de los algoritmos existentes para ello, por ejemplo, véase [Menezes et al., 1997], [Pollard, 1974], [Rivest et al., 1978], [Salomaa, 1990], entre otros.

2. Alicia calcula el producto $N = pq$. N es usado como módulo para ambas claves, pública y privada.

3. Se calcula el totiente: $\phi(N) = (p-1)(q-1)$

4. Alicia elige un entero e tal que $1 < e < \phi(n)$ y e es coprimo de $\phi(N)$, es decir, no poseen factores comunes, o equivalentemente, su máximo común divisor debe ser igual a la unidad: $mcd(e, \phi(n)) = 1$. e es el exponente de la clave pública.

5. Usando el algoritmo de Euclides Extendido, Alicia calcula el inverso de e módulo $\phi(N)$; esto es, Alicia calcula el único entero d que verifica las dos siguientes propiedades:

   El entero d debe ser mayor que 1 y menor que el indicador de Euler de n, es decir: $1 < d < \phi(N)$

   El resto de efectuar la división entera del producto ed entre $\phi(N)$ ha de ser igual a 1, lo cual se escribe: $ed = 1(mod\,\phi(n))$

6. Finalmente, Alicia publica su clave pública (N, e), mientras que mantiene en secreto su clave privada que es (N, d)

Notar que si se logra factorizar el módulo N y obtener p y q, generar la clave privada resulta trivial, por lo tanto la seguridad de RSA reside en que N no sea fácilmente factorizable.

Cifrando un mensaje

Alicia le envía su clave pública (N, e) a Benito y se guarda su propia clave privada. Benito le quiere enviar el mensaje M a Alicia.

Primero transforma a M en un número m menor que N.
Luego computa el texto cifrado c:

$c = m^e \mod N$

Luego Benito le envía c a Alicia.

Descifrando un mensaje

Alicia puede recuperar m de c utilizando su clave privada d:

$m = c^d \mod N$

2.2.6.3 Ejemplo

Aquí se ejemplifica de manera sencilla el uso de RSA (los parámetros utilizados son artificialmente pequeños):

Generación de claves

1. Escoger dos números primos, en este ejemplo tomamos $p = 524287$ y $q = 131071$
2. Calcular $N = p * q = 524287 * 131071 = 68718821377$
3. Calcular $\phi(N) = (p-1) * (q-1) = 524286 * 131070 = 68718166020$
4. Elegir e tal que $1 < e < \phi(N)$ y que sea coprimo con $\phi(N)$. Elegimos $e = 65537$
5. Los números N y e serán nuestra clave pública.
6. Se obtiene el exponente de clave privada d, calculando el inverso multiplicativo modular

   $d = modinv(e, \phi(N))$

   $d = modinv(65537, 68718166020)$

   $d = 56692460753$
7. Los números $N = 68718821377$ y $d = 56692460753$ serán nuestra clave privada.

Aclaraciones

- coprimo = dos números son coprimos o primos entre sí cuando no tienen divisores en común.

- e = exponente de la clave pública.
- d = exponente de la clave privada (se debe mantener en secreto junto con p y q)
- N = módulo de ambas claves, pública y privada
- (N, e) = clave pública
- (N, d) = clave privada
- $modinv$ = inverso multiplicador modular.

Cifrado Un mensaje cifrado C se podrá obtener a partir de un mensaje M de la siguiente manera:

1. $M = hola$
2. Representación hexadecimal de $M = 686f6c61$
3. Representación decimal de $M = 1752132705$
4. $C = (M^e) \mod N$
5. $C = (1752132705^{65537}) \mod 68718821377$
6. $C = 21543768249$ = mensaje cifrado

Descifrado Un mensaje cifrado C podrá ser descifrado utilizando el siguiente procedimiento:

1. $M = (C^d) \mod N$
2. $M = (21543768249^{56692460753}) \mod 68718821377$
3. $M = 1752132705$
4. M hexadecimal = $686f6c61$ (representación hexadecimal de los bytes de M)
5. $MASCII = \backslash x68\backslash x6f\backslash x6c\backslash x61 = hola$

2.2.7 Criptoanálisis

El objetivo de la criptografía es mantener el texto plano (o la clave, o ambos) en secreto de los intrusos. Se supone que estos últimos tienen acceso a las comunicaciones del emisor con el receptor. El criptoanálisis es la ciencia de recuperar el texto plano de un mensaje cifrado sin tener acceso a la clave. Un criptoanálisis exitoso puede recuperar el mensaje o bien la clave. También puede encontrar vulnerabilidades en un criptosistema que eventualmente lleve a los resultados anteriores.

Un intento de criptoanálisis se denomina ataque.

2.2.8 RSA - Criptoanálisis

Desde su publicación inicial, el sistema RSA ha sido analizado exhaustivamente por muchos investigadores en búsqueda de vulnerabilidades. Si bien no se ha encontrado un ataque devastador, los años de criptoanálisis de RSA nos han proporcionado una visión amplia de sus propiedades. Además de valiosas pautas para un uso e implementación adecuados.

A continuación presentamos los principales métodos utilizados en los ataques contra el sistema de cifrado RSA. Describimos los principales métodos de factorización y los ataques a la función matemática subyacente, así como los ataques que explotan los detalles en las implementaciones del algoritmo. Si bien existen muchos ataques, el sistema ha demostrado ser muy seguro y la mayoría de los problemas surgen como resultado de un uso incorrecto del sistema, una mala elección de los parámetros o fallas en las implementaciones.

2.2.8.1 Algoritmos de factorización

Como se dijo anteriormente, la seguridad de RSA depende fuertemente de la dificultad de factorizar el módulo N. Por este motivo se debe prestar especial atención al estado del arte de los algoritmos de factorización para grandes enteros a la hora de usar RSA. Existen estándares publicados con recomendaciones para el tamaño de N y el tamaño de los números primos a utilizar que dependen principalmente de cuanto tiempo a futuro se desean mantener seguros los datos cifrados. Estas recomendaciones dependen de algunos de los algoritmos presentados en esta sección.

Factorización de Fermat

Un método para encontrar los factores de un entero proviene del teorema desarrollado por Fermat en 1640 [4]. Suponer que $N = pq$ es el producto de dos números primos. Si se encuentran dos enteros $x, y : N = x^2 - y^2$ y $x - y > 1$ entonces tenemos $N = (x - y)(x + y)$ por lo tanto encontramos los dos factores de N. La idea de Fermat era probar muchos pares x, y hasta que la primer igualdad se verifique. Él creó un algoritmo para elegir estos valores de prueba. A continuación describimos el algoritmo que, para una entrada $N = pq$, devuelve sus factores.

Dado un entero compuesto N:

1. Establecer $w = \lfloor \sqrt{N} \rfloor$ y $x = w$
2. Establecer $y = \lfloor \sqrt{x^2 - N} \rfloor$
3. Si $N = x^2 - y^2$, retornar $x - y$ y $x + y$
4. Si $x < N$, reemplazar x por $x + 1$ e ir al paso 2

Como sabemos que N es compuesto, el algoritmo siempre va a retornar sus factores primos. Mirando el paso 4, podemos ver que el algoritmo probará todos los enteros desde $\sqrt{N}$ a N y por esto resulta un algoritmo muy lento. Sin embargo, debe tenerse en cuenta que cuando se aplica a un módulo RSA N cuyos dos factores primos están cerca uno del otro, el método funciona bastante rápido. En el caso extremo, cuando los factores son primos consecutivos, ¡unos pocos segundos serán suficientes para encontrarlos! Por lo tanto, cuando se implementa RSA, se debe tener en cuenta que la elección de números primos consecutivos hará que el sistema sea vulnerable a uno de los métodos de factorización más antiguos.

Algoritmo ρ de Pollard

El algoritmo ρ de Pollard, descrito por Pollard en 1975 [5], tiene como objetivo encontrar un pequeño factor p de un entero dado N. Presentamos una versión simplificada del algoritmo:

Dado un número compuesto N:

1. Establecer $a = 2, b = 2$
2. Definir el polinomio modular $f(x) = (x^2 + c) \mod N$ con $c \neq 0, -2$
3. Para $i = 1, 2...$, hacer:

a) Calcular $a = f(a), b = f(f(b))$

b) Calcular $d = gcd(a - b, N)$

c) Si $1 < d < N$ entonces retornar d resultando en éxito

d) Si $d = N$ entonces terminar el algoritmo resultando en fracaso

La función f es usada para crear dos secuencias pseudo aleatorias en $\mathbb{Z}$. La razón de esto es que, escogiendo al azar dos números $x, y \in \mathbb{Z}$ existe una probabilidad de 0.5 de que luego de $1.777\sqrt{N}$ intentos, uno sea congruente módulo N. Si esto sucede y $a \neq b$, entonces $gcd(a - b, N)$ da un factor de N. El tiempo de ejecución del algoritmo es $O(\sqrt{p})$, donde p es el factor más pequeño de N. Esto significa que ante un módulo RSA N con primos balanceados, el tiempo de ejecución del algoritmo es $O(N^{1/4})$ tornándolo un método ineficiente.

Método con Curvas Elípticas

En 1987, Hendrik Lenstra tuvo una idea visionaria [6]. Pensó en usar curvas elípticas para factorizar números enteros. Se dice que el descubrimiento de Lenstra se basó en el método de Pollard ρ. El tiempo de ejecución del Método con Curvas Elípticas (ECM) depende de que se factorice el factor p más pequeño del entero N, lo que lo hace particularmente poderoso para encontrar factores de "tamaño mediano" de enteros grandes. El ECM no se usa directamente para factorizar el módulo RSA, en su lugar se usa como un paso auxiliar en el algoritmo de la Criba General del Cuerpo de Números o en inglés, General Number Field Sieve (GNFS) que se menciona en el siguiente párrafo. La base del algoritmo es la siguiente:

Dado un número compuesto N:

1. Comprobar que $gcd(6, N) = 1$ por lo que $N \neq m^r$ para cualquier $r \geq 2$. (gcd = greatest common divisor o divisor común mayor)

2. Elegir enteros A, x_1, y_1 tales que $1 < A, x_1, y_1 < N$

3. Sea E la curva eliptica $E : y^2 = x^3 + Ax + B$ donde $B = y_1^2 - x_1^2 - Ax_1$ y sea $P = gcd(x_1, y_1) \in E$

4. Comprobar que $a = gcd(4A^3 + 27B^2, N) = 1$.

   Si $a = N$, entonces volver al paso 2 y escoger un nuevo A.

   Si $1 < a < N$, entonces a es un factor de N

5. Escoger un entero k tal que $k = lcm\{1,2,3,...,K\}$ para algún $K \in \mathbb{N}$

6. Calcular $kP = gcd(a_k/d_k^2, b_k/d_k^3) = 1P + 2P + 2^2P + ... + 2^rP$ para algún $r \in \mathbb{N}$

   $= P_0 + P_1 + P_2 + ... + P_r$, para algún $r \in \mathbb{N}$

   $= \sum_{k_i=1}^{r} P_i$

7. Calcular $D = gcd(d_k, N)$

   Si $1 < D < N$, entonces D es un factor no trivial de N. Si $D = 1$, retornar al paso 2 y elegir un nuevo A. Si $D = N$, retornar al paso 5 y decrementar el valor de k.

El Método con Curvas Elípticas es el tercer método más rápido conocido de factorización y el tiempo de ejecución de este algoritmo es sub-exponencial.

Criba General del Cuerpo de Números

La Criba General del Cuerpo de Números [7] es el método más rápido conocido de factorización de grandes enteros; es un algoritmo extremadamente complejo que hace uso de muchos campos de la matemática. El tiempo de ejecución de la GNFS para factorizar un entero N de tamaño n es $E(n) = exp(1.923n^{1/3}log^{2/3}n)$.

2.2.8.2 Primeros ataques al protocolo

En RSA los parámetros $\{d, p, q, \phi(n)\}$ deben ser mantenidos en secreto. Estas cuatro piezas de información son igualmente importantes. El conocimiento de cualquiera de ellos revela los otros tres restantes y por lo tanto permite romper el cifrado por completo. Sin embargo, si RSA no es utilizado correctamente, es posible que se pueda romper el cifrado sin el uso del conocimiento de alguno de esos parámetros. Los ataques de esta sección son posibles cuando no se implementa RSA de manera adecuada.

Ataque de módulo común

Uno de tales usos indebidos es el uso del módulo N común. Supongamos que Benito envía a Alicia dos textos cifrados C_1 y C_2 de la siguiente manera:

$$C_1 \equiv M^{e1} \pmod{N}$$

$$C_2 \equiv M^{e2} \pmod{N}$$

Donde $gcd(e_1, e_2) = 1$. Entonces como demuestra el siguiente teorema, Eve puede obtener el texto plano M sin factorizar N y sin necesitar ninguno de los parámetros $\{d, p, q, \phi(N)\}$.

Teorema: Sea $n_1 = n_2$ y $M_1 = M_2$ pero $e_1 \neq e_2$ y $gcd(e_1, e_2) = 1$ tal que

$$C_1 \equiv M^{e1} \pmod{N}$$

$$C_2 \equiv M^{e2} \pmod{N}$$

Entonces M puede ser recuperado fácilmente:

$$\{[C_1, e_1, N], [C_2, e_2, N]\} \Rightarrow \{M\}$$

Demostración: Como $gcd(e_1, e_2) = 1$ entonces $e_1 x + e_2 y = 1$ con $x, y \in \mathbb{Z}$, lo cual se puede calcular con el algoritmo extendido de Euclides en tiempo polinomial. Así,

$$\begin{aligned} C_1^x C_2^y &\equiv (M_1^{e_1})^x (M_2^{e_2})^y \\ &\equiv M^{e_1 x + e_2 y} \\ &\equiv M(mod N) \end{aligned}$$

Este ataque sugiere que nunca debe usarse un módulo común en el cifrado RSA.

Ataques de Hastad

Otro error en RSA se produce al cifrar varios mensajes de texto plano relacionados entre si con exponentes públicos pequeños y módulos diferentes. Colectivamente estos ataques se denominan Ataques de Hastad, por ser Johan Hastad quien los describió [8]. Consideraremos dos tipos de ataques: ataque de texto plano común y ataques de texto plano relacionado.

Ataque de texto plano común

Existe un problema cuando el mismo texto plano M es cifrado con varias claves públicas (e, N_i), cada una usando el mismo exponente e y un módulo diferente N_i.

Teorema: Suponga un texto plano m es cifrado k veces con las claves públicas $(e, N_1), (e, N_2), ..., (e, N_l)$ donde $k \geq e$ y $N_1, N_2, ..., N_k$ son coprimos entre si. Sea $N_0 = min\{N_1, N_2, ..., N_k\}$ y $N = \prod_{i=1}^{k} N_i$. Si el texto plano m satisface que $m < N_0$ entonces

Marvin, sabiendo que $c_i \cong m^e \pmod{N_i}$ y (e,N_i) para $i = 1,2,...,k$, puede computar el texto plano M en tiempo polinomial en $log(N)$

Demostración: Dado que los módulos N_i son coprimos entre si, se puede utilizar el teorema chino del resto para calcular $C \cong m^e \pmod{N}$. Como $m < N_0$ tenemos que $m^e < N_1 N_2 ... N_k = N$ entonces $C = m^e$. Por lo tanto lo que queda hacer es calcular la e-ésima raíz de C para encontrar el texto plano m.

Ataque de texto plano relacionado

Este ataque [9] permite que los textos planos sean diferentes pero deben estar relacionados por polinomios conocidos, y requiere que un número más grande de mensajes sean cifrados.

Teorema: Dadas las claves públicas $(e_1,N_1),(e_2,N_2),...,(e_k,N_k)$ donde los módulos son coprimos entre si, y $f_1(x) \in \mathbb{Z}_{N1}[x]$,..., $f_k(x) \in \mathbb{Z}_{Nk}[x]$, con $N_0 = minN_1,N_2,...,N_k$ y $N = \prod_{i=1}^{k} N_i$. Para un texto plano $m < N_0$, si $k \geq max_i e_i deg(f_i(x))$ entonces dado $c_i = f_(m)(modN_i)$ y (e_i,N_i) para $i = 1,2,...,k$, el texto plano m puede ser calculado en tiempo polinomial en $log(N)$

Demostración: Se asume que todos los polinomios $f_i(x)$ son mónicos. Si $f_j(x)$ no lo es se lo debe multiplicar por el inverso del coeficiente principal. Si el inverso no existe, se revela la factorización de N_j la cual rompe esa instancia de RSA y permite descifrar c_i para recuperar m.
Sea $\delta = max_i\{e_i deg(f_i(x))\}$ y $h_i = \delta - deg(f_i(x)^{e_i})$ para $i = 1,..,k$. Luego se definen k polinomios mónicos de grado δ:

$$g_i(x) = x^{h_i}(f_i(x)^{e_i} - c_i) \in \mathbb{Z}_{Ni} \text{ para } i = 1,...,k$$

Notar que $g_i(m) \cong 0 \pmod{N_i}$ para $i = 1,...,k$, entonces se puede utilizar el teorema chino del resto tomando a $g_i(x)$ y N_i como entradas y calcular un nuevo polinomio mónico de grado δ $G(x) \in \mathbb{Z}_N[x]$ que satisface:

$$G(m) \cong 0 \pmod{N}$$

donde $m < N_0 < N^{1/l} < N^{1/D}$. Entonces se está en condiciones de utilizar el método Coppersmith [10] y calcular m en tiempo polinomial en $log(N)$ y δ.

Ataques cíclicos En 1977, Simmons y Norris [11] observaron que un texto plano siempre se puede calcular volviendo a cifrar repetidamente su texto cifrado hasta que vuelva a sí mismo (es decir, vuelva al texto cifrado original). Dado un texto cifrado

$c = m^e mod N$ y la clave pública (e,N), si después de $l+1$ cifrados se recupera el texto cifrado, es decir

$$c^{e^{l+1}} \equiv c \pmod{N}$$

entonces

$$c^{e^{l}} \equiv m \pmod{N}$$

Por lo tanto, el texto en claro es revelado luego de l recifrados. El ataque cíclico original consiste simplemente en encontrar el l más pequeño con el cual el texto plano se puede recuperar. Este valor para l se conoce como el exponente de recuperación (del mensaje m).

Tenga en cuenta que hay algunos mensajes de texto plano que siempre tienen un exponente de recuperación muy pequeño. Por ejemplo los textos plano $m = \pm 1$ tienen un exponente de recuperación $l = 1$.

2.2.8.3 Ataques a instancias con exponente público bajo

Esta sección se refiere a la seguridad del sistema de RSA cuando se utiliza un exponente público bajo, como $e = 3$. Esto puede ocurrir cuando el dispositivo de cifrado tiene poco poder computacional. Estos ataques están lejos de romper el sistema, ya que no pretenden factorizar el módulo, sino recuperar textos planos específicos o parte de ellos.

Ataque de mensaje estereotipado Cuando una parte del texto plano es conocida, es posible recuperar el texto plano entero si el exponente público y el tamaño de la parte del mensaje desconocido son lo suficientemente pequeños. En el caso extremo suponemos que se conoce que el texto plano va a ser pequeño en relación al tamaño del módulo. Esto es, sabemos que los bits más significativos del texto plano son todos ceros. Si un texto plano $m < N^{1/e}$ es cifrado con el exponente público e, entonces claramente el texto plano puede ser recuperado ya que:

$$c = m^e \bmod N = m^e$$

Simplemente calculando la raíz e-esima del texto cifrado c se revela el mensaje m. Cuando el exponente público es muy pequeño, como $e = 3$ o $e = 5$, es posible que el texto plano sea más chico que $N^{1/e}$. Por ejemplo, con un módulo de 1024 bits y un exponente público $e = 3$, un mensaje de tamaño menor a 342 bits puede ser fácilmente descifrado. Una manera de contrarrestar este ataque sería agregar bits aleatorios al texto plano

para que se cumpla $m > N^{1/e}$. Ahora suponga que el texto plano es grande, por lo tanto el ataque anterior no funcionaría, pero conocemos la mayor parte de el. Por ejemplo, suponga un mensaje de la forma: "El secreto para el 29 de febrero de 2008 es ?????", donde el secreto es desconocido pero pequeño. En esta situación Coppersmith [10] demostró que si la parte desconocida del mensaje es lo suficientemente pequeña entonces es posible recuperarla a partir del texto cifrado.

Teorema: Sea (e,N) una clave pública RSA válida y sea m cualquier texto plano. Dada la clave pública y $c = m^e \bmod N$, si todos menos como mucho una fracción de $1/e$ bits consecutivos del texto plano son conocidos, entonces todo el mensaje puede ser descifrado en tiempo polinomial en $log(N)$ y en e.

Prueba: Como al menos una fracción de $1/e$ bits consecutivos de m son conocidos, entonces podemos describir el mensaje como $m = m_2 2^{k_2} + m_1 2^{k_1} + m_0$ donde todo es conocido menos m_1. Además se sabe que m_1 satisface $|m_1| < N^{1/e}$. Esto sugiere buscar solución a polinomio de grado e $f_N(x) \in \mathbb{Z}_N[x]$ dado por:

$$f_N(x) = 2^{-k_1 e}((m_2 2^{k_2} + x 2^{k_1} + m_0)^e - c) \bmod N$$

dado que $f_N(m_1) = 2^{-k_1 e}(m^e - c) = 0 \bmod N$. Esto es, m_1 es una raíz de $f_N(x) \bmod N$. Dado que $|m_1| < N^{1/e}$, el método de Coppersmith puede ser usado para computar m_1 y revelar el mensaje completo.

Ataque de mensaje relacionado

Cuando dos mensajes $M_1 y M_2$ están relacionados por $M_2 = \alpha M_1 + \beta$ y se cifran con la misma clave pública, es posible recuperar los textos planos si el exponente público es pequeño y se conoce la relación. El ataque original para el exponente público e = 3, por Franklin y Reiter [12], tomado de Coppersmith et al. [13], se da en el siguiente teorema:

Teorema: Sea (e,N) una clave pública de RSA válida con $e = 3$. Sean m_1 y m_2 dos mensajes que satisfacen $m_2 = a m_1 + b$. Dado $a, b, c_1 = (m_1)^3 \bmod N$, $c_2 = (m_2)^3 \bmod N$ y la clave pública, ambas m_1 y m_2 pueden ser computadas en tiempo polinomial en $log(N)$.

Prueba: Dado c_1, c_2, a, b y N, el texto plano m_1 puede ser computado ya que:

$$\frac{b(c_2 + 2a^3 c_1 - b^3)}{a(c_2 - a^3 c_1 + 2b^3)} \bmod N = \frac{m_1(3a^3 b m_1^2 + 3a^2 b^2 m_1 + 3ab^3)}{3a^3 b m_1^2 + 3a^2 b^2 m_1 + 3ab^3} \bmod N = m_1$$

Una vez que se conoce m_1, puede calcularse $m_2 = a m_1 + b$. Si el cálculo falla (por ejemplo el denominador no existe módulo N) entonces se encontró un factor de N y se rompe el sistema.

2.2.8.4 Ataques a instancias con exponente privado bajo

Esta sección hace referencia a los peligros de utilizar RSA con un exponente privado demasiado bajo. Mientras que en algunas situaciones puede ser deseable reducir los costos de descifrado lo máximo posible, existen ataques para vulnerar completamente el sistema RSA si el exponente es lo suficientemente bajo.

Ataque de Wiener

Un ataque significativo es el ataque de fracción continua de Wiener [14]. El mismo demuestra que elegir un valor pequeño para d dará como resultado un sistema inseguro en el que un atacante puede recuperar toda la información que se desea mantener secreta, es decir, romper el sistema RSA. Wiener ha probado que el atacante puede recuperar eficientemente d cuando $d < \frac{1}{3}N^{\frac{1}{4}}$. Wiener además comenta algunas contramedidas para defenderse de este ataque que permiten también un descifrado rápido. Las técnicas son las siguientes:

- **Elegir una clave pública grande**: Reemplazar e por e', donde $e' = e + k\varphi(N)$ para algún k grande. Cuando e' es lo suficientemente grande, por ejemplo $e' > N^{\frac{3}{2}}$, entonces el ataque de Wiener no puede ser aplicado sin importar que tan bajo es d.

- **Usar el Teorema Chino del Resto**: Suponga que se escoge d tal que ambos $d_p = d \mod (p-1)$ y $d_q = d \mod (q-1)$ son pequeños pero d no lo es, entonces se puede descifrar C rápidamente de la siguiente manera:

  1. Primero computar $M_p \equiv C^{d_p} \mod p$ y $M_q \equiv C^{d_q} \mod q$
  2. Luego utilizar el Teorema Chino del Resto para calcular el único valor de $M \in \mathbb{Z}_{\mathbb{N}}$ el cual satisface $M \equiv M_p \mod p$ y $M \equiv M_q \mod q$. El resultado de M satisface $M \equiv C^d \mod N$ como se necesitaba. El punto está en que el ataque de Wiener no puede aplicarse ya que el valor de $d \mod \varphi(N)$ puede ser muy grande.

Teorema: Sea $N = pq$ con $q < p < 2q$ y sea $d < \frac{1}{3}N^{\frac{1}{4}}$.

Dados (N, e) con $ed = 1 \pmod{\varphi(N)}$ entonces el atacante puede recuperar d eficientemente.

Prueba: La comprobación está basada en aproximaciones usando fracciones continuas. Como $ed = 1 \mod \varphi(N)$, existe k tal que $ed - k\varphi(N) = 1$.

Por lo tanto:

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)}$$

.

Por consiguiente, $\frac{k}{d}$ es una aproximación de $\frac{e}{\varphi(N)}$.

A pesar de que el atacante no conoce $\varphi(N)$, puede usar N para aproximarlo.

En efecto, dado que $\varphi(N) = N - p - q + 1$ y $p + q - 1 < 3\sqrt{N}$, tenemos $|N - \varphi(N)| < 3\sqrt{N}$.

Usando N en lugar de $\varphi(N)$ obtenemos:

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed - k\varphi(N) - kN + k\varphi(N)}{Nd}\right|$$

$$= \left|\frac{1 - k(N - \varphi(N))}{Nd}\right| \leq \left|\frac{3k\sqrt{N}}{Nd}\right| = \frac{3k}{d\sqrt{N}}$$

Ahora, $k\varphi(N) = ed - 1 < ed$, como $e < \varphi(N)$, podemos ver que $k < d < \frac{1}{3}N^{\frac{1}{4}}$, entonces se obtiene:

$$\left|\frac{e}{N} - \frac{k}{d}\right| \leq \frac{1}{dN^{\frac{1}{4}}} < \frac{1}{2d^2}$$

Esta es una relación de aproximación clásica. El número de fracciones $\frac{k}{d}$ con $d < N$ aproximándose tan cerca a $\frac{e}{N}$ está limitado por $log_2 N$. De hecho, todas las fracciones son obtenidas como convergentes de la expansión continua de la fracción de $\frac{e}{N}$. Todo lo que queda hacer es calcular las $\log N$ convergentes de la fracción continua para $\frac{e}{N}$. Una de ellas será igual a $\frac{k}{d}$. Como $ed - k\varphi(N) = 1$, tenemos $\gcd(k, d) = 1$, por lo tanto $\frac{k}{d}$ es una fracción reducida. Este es un algoritmo de tiempo lineal para recuperar la clave privada d.

Ataque de Boneh y Durfee

Boneh y Durfee en [15] mejoran el resultado de Wiener y demuestran, basándose en el Algoritmo de Lenstra-Lenstra-Lovász [16] , que si el exponente privado d utilizado en un sistema RSA es menor que $N^{0.292}$ entonces el sistema es inseguro.

2.2.8.5 Ataque por exposición parcial de clave privada

Un conocido principio en Criptografía indica que la seguridad de cualquier sistema criptográfico debería basarse en mantener secreta determinada clave. Un ataque al criptosistema RSA propuesta por Boneh, Durfee y Frankel [17] muestra la importancia de proteger íntegramente el exponente privado d. Ellos demostraron que, si el módulo n es de k bits, conociendo los $k/4$ bits menos significativos de d, el atacante puede

reconstruir todo d en tiempo lineal a $e(log(e))$, donde e es el exponente público. Esto significa que si e es pequeño, y se encuentran expuestos aunque sea 1/4 bits de d es posible recuperar toda la clave privada d

Teorema de BDF: Sea (N,d) una clave privada RSA en donde N tiene longitud de n bits. Dados al menos los $n/4$ bits menos significativos de d, un atacante puede reconstruir todo d en tiempo lineal en $e(log(e))$.

Teorema de Coppersmith: Sea $N = pq$ un módulo RSA de n bits. Dados al menos los $n/4$ bits menos significativos de p o los $n/4$ bits más significativos de p, uno puede factorizar eficientemente N.

Desarrollo: Por defección de e y d, existe un entero k tal que:

$$ed - k(N - p - q + 1) = 1$$

Como $d < \phi(N)$ debemos tener $0 < k \leq e$. Reduciendo la ecuación módulo $2^{n/4}$ y seteando $q = N/p$, obtenemos:

$$(ed)p - kp(N - p + 1) + kN = p(mod 2^{n/4})$$

Ya que el atacante conoce los $n/4$ bits menos significativos de d, conoce el valor de $edmod2^{n/4}$. Por consiguiente, obtiene una ecuacion en k y p. Por cada uno de los e posibles valores de k, el atacante resuelve la ecuación cuadrática en p y obtiene un número de candidatos para $pmod2^{n/4}$. Para cada uno de esos candidatos, ejecuta el algoritmo del teorema de Coppersmith para intentar factorizar N. Se puede demostrar que el número total de candidatos para $pmod2^{n/4}$ es como mucho $e(log(e))$. Por lo tanto, después de a lo sumo $e(log(e))$ intentos, N será factorizado.

2.2.8.6 Ataques de implementación o de canal lateral

La mayoría de los ataques contra RSA vistos hasta el momento se basan en debilidades del algoritmo y sus parámetros. Por otro lado existe una categoría muy grande de ataques, que no intentan atacar directamente al algoritmo RSA o al sistema en sí, sino que intentan atacar vulnerabilidades específicas de la implementación en un dispositivo en particular, como por ejemplo tarjetas inteligentes, tokens, o una computadora, lo que permitiría al atacante filtrar información secreta, como por ejemplo claves privadas, mensajes, etc.

Ataque de timing

Paul Kocher introdujo los ataques de timing contra RSA en 1995 [18]. Los ataques

de timing aprovechan la correlación entre la clave privada d y el tiempo de ejecución de la operación criptográfica de descifrado. Se sabe que la operación de descifrado de RSA consiste en una exponenciación modular utilizando la clave privada d como exponente. Las exponenciaciones modulares generalmente son implementadas utilizando un algoritmo del tipo "Square-and-Multiply". Si la clave privada tiene k bits de longitud, la operación consiste en un bucle que se ejecuta a través de los bits de d, con un máximo de $2k$ multiplicaciones modulares. En cada paso, los datos son elevados al cuadrado con la ejecución de una multiplicación modular si el actual bit del exponente es uno. Al medir el tiempo de ejecución de la operación de descifrado en un gran número de mensajes aleatorios, un atacante puede conocer bits de d uno por uno, comenzando con el bit menos significativo y así recuperar el exponente d completo almacenado en el dispositivo. Como contramedida se pueden agregar delays aleatorios al algoritmo de exponenciación para confundir al atacante.

Análisis de potencia

En 1998, Paul Kocher [19] en conjunto con investigadores de su compañía descubrieron otro ataque apuntado a tarjetas inteligentes y tokens criptográficos. El ataque se basa en monitorizar el consumo de energía del dispositivo. El consumo de energía varía significativamente durante los distintos pasos de las operaciones criptográficas, atacante puede sacar provecho de esto y recuperar información secreta. Se definieron dos tipos de ataques: los ataques de potencia simples, que consisten en observar directamente el consumo de energía de un dispositivo, y los ataques de análisis de potencia diferencial, que son más poderosos, ya que utilizan el análisis estadístico y las técnicas de corrección de errores para extraer información correlacionada con claves privadas. A pesar de que estos ataques son bastante complejos y requieren un alto nivel de habilidad técnica para implementar, Kocher dice que "se pueden realizar con unos pocos miles de dólares de equipo estándar y, a menudo, pueden romper un dispositivo en unas pocas horas o menos"

Ataque de análisis de fallos en RSA Funcionan explotando errores en las operaciones criptográficas realizadas por el dispositivo. Estos errores pueden ser aleatorios, latentes, o inducidos. Existen varios ataques de análisis de fallas contra dispositivos criptográficos de clave pública y clave privada. Tal como se sabe, la operación de descifrado en RSA es muy intensiva en cómputo, consiste en una exponenciación modular que utiliza números típicamente en el rango de los 300 dígitos decimales.

Muchas implementaciones de descifrado y firma RSA utilizan una técnica conocida como el Teorema del Resto Chino (CRT), que al trabajar con módulo p y q (en lugar de módulo $n = pq$) puede proporcionar una mejora considerable en el rendimiento. Boneh, Demillo y Lipron [20] describen una técnica que, al explotar un error que ocurre durante el descifrado o al firmar, y analizar la salida, un atacante podría factorizar el módulo n y, por lo tanto, recuperar la clave privada del dispositivo. Tanto la salida como la entrada de la operación son necesarios para que el ataque tenga éxito. Para realizar este tipo de ataque, solo se necesita inducir un error en el dispositivo durante la operación de descifrado (por ejemplo, por voltaje o variación de velocidad de reloj). También debemos tener en cuenta que, a diferencia de muchos de los ataques descriptos anteriormente, la dificultad de este es independiente de la longitud de la clave.

2.3 Esteganografía

2.3.1 Introducción

En este capítulo detallaremos el concepto de Esteganografía. La misma, es una técnica para ocultar un mensaje secreto dentro de un mensaje ordinario y extraerlo en el destino para mantener la confidencialidad de los datos. La palabra esteganografía deriva de las palabras griegas "stegos" (cobertura) y "grafía" (escritura) definiéndose como "escritura cubierta". En resumen, es el arte de ocultar datos de manera discreta dentro de otros datos.

El objetivo principal de la esteganografía es ocultar la información lo suficientemente bien como para que los destinatarios involuntarios no sospechen que el medio esteganográfico contiene datos ocultos. La mayoría de los trabajos de esteganografía se han llevado a cabo en diferentes medios de cobertura de almacenamiento como texto, imagen, audio o vídeo.

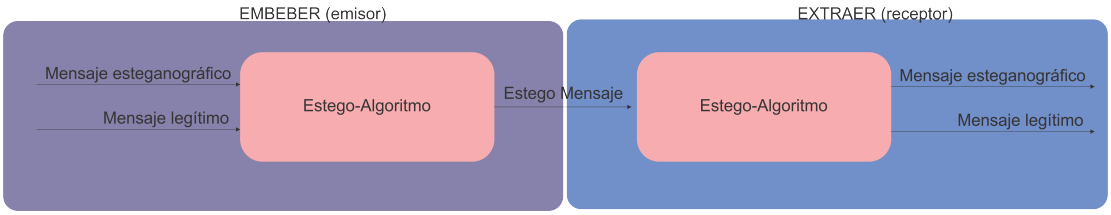
Se suele confundir la esteganografía y la criptografía dado a que ambos se utilizan para garantizar la confidencialidad de los datos. Sin embargo, la principal diferencia entre ellos es que, con el cifrado, cualquiera puede ver que ambas partes se están comunicando en secreto. La esteganografía oculta la existencia de un mensaje secreto y, en el mejor de los casos, nadie puede ver que ambas partes se están comunicando en secreto. El cifrado permite una comunicación segura que requiere una clave para leer la información. Un atacante no puede eliminar el cifrado, pero es relativamente fácil modificar el archivo, por lo que es ilegible para el destinatario deseado.

El estegoanálisis es el proceso para detectar la presencia de esteganografía. En este capítulo veremos diferentes enfoques hacia la implementación de la esteganografía archivos "multimedia" (texto, imagen estática, audio y vídeo). También se discutirán algunos métodos de estegoanálisis.

En la categoría Esteganografía de CTF's, se debe aplicar estegoanálisis para encontrar cuál es el mensaje esteganográfico embebido, el cuál suele ser la *Flag*.

2.3.2 Funcionamiento y Terminología

En el siguiente gráfico se ilustra las acciones de embeber y extraer un mensaje. Luego describimos la terminología básica de éste proceso.



Se define como **esquema esteganográfico** al conjunto de componentes que permite llevar a cabo la comunicación esteganográfica.

El **portador** es el conjunto de datos susceptible de ser alterado para incorporar el mensaje que queremos mantener en secreto. Puede ser de muchos tipos o formatos. Ejemplos: imagen (en sus distintos formatos), audio (en sus distintos formatos), texto plano, archivos binarios, mensaje de protocolo de comunicación, entre otros.

Se habla de **mensaje-legítimo** para referirse al mensaje transportado por el portador.

Se llama **mensaje esteganográfico** al mensaje que queremos mantener en secreto y queremos esconder dentro del portador. Pueden ser de distintos tipos o formatos. Ejemplos: imagen (en sus distintos formatos), audio (en sus distintos formatos), texto plano, archivos binarios, entre otros.

Estego-algoritmo es el algoritmo esteganográfico que indica cómo realizar el procedimiento de incorporación del mensaje que queremos mantener en secreto en el portador.

La acción de ocultar el mensaje dentro del portador se denomina **empotrar** (del inglés *to embed*).

Se llama **estego-mensaje** al resultado de empotrar el mensaje esteganográfico dentro del portador.

La acción de recuperar, a partir del estego-mensaje, del mensaje oculto esteganográfico se denomina **extraer**.

Por el rol desempeñado dentro del proceso esteganográfico, el emisor también se llama **embebedor** y el receptor, **extractor**. Al igual que en todo acto de comunicación convencional, es común que los roles de emisor y receptor se intercambien sucesivamente entre las partes que se comunican.

2.3.3 Clasificación

El estego-algoritmo es el algoritmo esteganográfico que indica cómo realizar el procedimiento de incorporación del mensaje esteganográfico en el portador para obtener

el estego-mensaje. Según el tipo de estego-algoritmo podemos distinguir entre dos tipos de esteganografía: Esteganografía pura y esteganografía de clave secreta.

2.3.3.1 Esteganografía pura

En este tipo de esteganografía el estego-algoritmo establece un método fijo para incorporar el mensaje esteganográfico en el portador para obtener el estego-mensaje. En esta estrategia se está suponiendo que la víctima (o el guardián del mensaje legítimo) no conoce nada sobre el estego-algoritmo. Por lo tanto estamos basando nuestra seguridad en la oscuridad. Este enfoque para conseguir seguridad raramente funciona y es especialmente malo en el caso de la criptografía.

2.3.3.2 Esteganografía de clave secreta

En este tipo de esteganografía el estego-algoritmo está parametrizado por una clave esteganográfica a la que se le llama estego-clave que define como aplicar el algoritmo. Por ejemplo, la estego-clave podría indicar el lugar dentro del portador a partir del cual se comienza a realizar la incorporación del mensaje secreto. Tanto el estego-algoritmo como la estego-clave deben estar previamente acordadas entre el emisor y el receptor. El proceso de extracción consiste en aplicar el estego-algoritmo y la estego-clave necesarios al estego-mensaje recibido para obtener el mensaje esteganográfico.

2.3.4 Medios de esteganografía

La esteganografía puede estar presente en muchos medios y con muchas técnicas. En esta sección haremos foco en las técnicas esteganográficas más comunes en CTF's, ordenadas por el medio digital del portador.

2.3.4.1 Esteganografía en Texto

Se puede lograr alterando el formateo del texto o alterando ciertas características de los elementos textuales (por ejemplo, los caracteres). El objetivo en el diseño de los métodos de codificación es desarrollar alteraciones que sean decodificables de manera confiable (incluso en presencia de ruido), pero que en gran medida no sean perceptibles para el lector. Estos criterios, decodificación confiable y cambio mínimo visible, son algo conflictivos; aquí radica el desafío en el diseño de técnicas de marcado en los documentos. El archivo de formato de documento es un archivo de computadora que describe el contenido del documento y el diseño de la página (o el formato), utilizando lenguajes de

descripción de formato estándar como PostScript2, TeX, @off, etc. Es de este archivo de formato que la imagen (lo que ve el lector) es generado. Las tres técnicas de codificación que proponemos ilustran diferentes enfoques los cuales se pueden usar por separado o en conjunto. Cada técnica goza de ciertas ventajas y pueden diferir en aplicabilidad como veremos a continuación.

Codificación de cambio de línea

En este método se altera la posición vertical de las líneas de texto para ocultar un secreto dentro del documento. Esta codificación puede aplicarse al archivo de formato o a la imagen del documento. El secreto embebido puede extraerse tanto del archivo de formato como de la imagen. En algunos casos esta decodificación puede lograrse sin necesidad del documento original, ya que se conoce de antemano que el mismo tiene un espaciado de linea uniforme.

Codificación de cambio de palabra

En este método se altera la posición horizontal de las palabras dentro de las líneas de texto para ocultar un secreto dentro del documento. Esta codificación se puede aplicar al archivo de formato o la imagen del documento. El secreto embebido puede extraerse tanto del archivo de formato como de la imagen. El método es aplicable solo a documentos con espaciado variable entre palabras adyacentes. El espaciado variable en los documentos de texto se usa comúnmente para distribuir espacios en blanco al justificar texto. Debido a este espaciado variable, la decodificación requiere la imagen original, o más específicamente, el espaciado entre palabras del documento no codificado.

Codificación de características

Este es un método de codificación que se aplica al archivo de formato o a la imagen del documento. La imagen se examina para determinadas cualidades del texto, y esas características se modifican, o no, según la palabra en clave deseada. La decodificación requiere la imagen original, o más específicamente, una especificación del cambio en píxeles en una característica. Hay muchas opciones posibles de características de texto; una por ejemplo, sería alterar las líneas verticales finales de determinadas letras, es decir, la parte superior de las letras, b, d, h, etc. Estas letras se alteran extendiendo o acortando sus longitudes en uno (o más) píxeles.

Hay otra forma de esteganografía de texto la cual es definida por Chapman et al. [21] en la cual se hace uso del lenguaje natural escrito para ocultar un mensaje secreto.

2.3.4.2 Esteganografía en Imágenes

Ocultar información dentro de imágenes es una técnica popular hoy en día. Una imagen con un mensaje secreto en el interior se puede distribuir fácilmente en la Web o en grupos de noticias. Para ocultar un mensaje dentro de una imagen sin cambiar sus propiedades visibles, la imagen portador se puede alterar en áreas "ruidosas", por ejemplo con muchas variaciones de color, por lo que las modificaciones en esas áreas atraerán menos la atención. Los métodos más comunes para hacer estas alteraciones involucran el uso del bit menos significativo o LSB, el enmascaramiento, el filtrado y las transformaciones en la imagen portador. Estas técnicas se pueden utilizar con distintos grados de éxito en diferentes tipos de archivos de imagen.

Bits menos significativos

Un enfoque simple para incrustar información en la imagen portador es usar los bits menos significativos (LSB). Las técnicas de esteganografía más simples incrustan los bits del mensaje directamente en los LSB de la imagen de portada, la modificación de éstos no produce una diferencia perceptible por el ser humano porque la magnitud del cambio es muy pequeña. Para ocultar un mensaje secreto dentro de una imagen, se necesita una imagen de portada adecuada. Debido a que este método utiliza bits de cada píxel en la imagen, es necesario usar un formato de compresión sin pérdida, de lo contrario la información oculta se perderá en las transformaciones. Cuando se utiliza una imagen en color de 24 bits, se puede usar un poco de cada uno de los componentes de color rojo, verde y azul, por lo que se puede almacenar un total de 3 bits en cada píxel.

Por ejemplo, la siguiente cuadrícula puede considerarse como 3 píxeles de una imagen en color de 24 bits, usando 9 bytes de memoria:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

Cuando el carácter A (cuyo valor binario es igual a 10000001) es insertado, los bytes de los píxeles quedan de esta forma:

(00100111 11101000 11001000)

(00100110 11001000 11101000)

(11001000 00100111 11101001)

En este caso, solo se necesitan cambiar tres bits para insertar el carácter correctamente. En promedio, solo será necesario modificar la mitad de los bits en una imagen para ocultar un mensaje secreto. Los cambios en los resultados que se realizan en

los bits menos significativos son demasiado pequeños para ser reconocidos por el sistema visual humano (HVS), por lo que el mensaje está oculto de manera efectiva.

Enmascaramiento y filtrado

Las técnicas de enmascaramiento y filtrado, generalmente restringidas a imágenes de 24 bits o escala de grises, adoptan un enfoque diferente para ocultar un mensaje. Estos métodos son efectivamente similares a las marcas de agua de papel, creando marcas en una imagen. Esto se puede lograr, por ejemplo, modificando la luminancia de partes de la imagen. Mientras que el enmascaramiento cambia las propiedades visibles de una imagen, se puede hacer de tal manera que el ojo humano no note las anomalías. Dado que el enmascaramiento utiliza aspectos visibles de la imagen, es más robusto que la modificación LSB con respecto a la compresión, el recorte y los diferentes tipos de procesamiento de imágenes. La información no está oculta en el nivel de "ruido", pero está dentro de la parte visible de la imagen, lo que la hace más adecuada que las modificaciones de LSB en caso de que se esté utilizando un algoritmo de compresión con pérdida como el JPEG.

Transformaciones

Una forma más compleja de ocultar un mensaje secreto dentro de una imagen surge a partir del uso y las modificaciones de las transformaciones discretas del coseno (DCT). El algoritmo de compresión JPEG utiliza DCT para transformar sucesivos bloques de 8 x 8 píxeles de la imagen, en 64 coeficientes DCT cada uno. Cada coeficiente DCT *F(u, v)* de un bloque de 8 x 8 píxeles de imagen $f(x,y)$ viene dado por:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)\right], C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & C.C. \end{cases}$$

Después de calcular los coeficientes, se realiza la siguiente operación:

$$F^Q(u,v) = \frac{F(u,v)}{Q(u,v)}$$

Donde $Q(u,v)$ es una tabla de cuantificación de 64 elementos. Un simple algoritmo en pseudo código para ocultar un mensaje dentro de una imagen JPEG podría ser:

Entrada: estego-mensaje, imagen portadora
Salida: Imagen esteganográfica contenedora de un mensaje

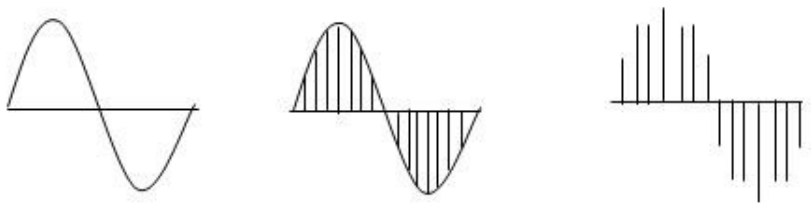
while haya información que embeber
  Obtener el siguiente coheficiente DCT de la imagen portadora
  if DCT 6=0 and DCT 6=1 then
    Obtener el siguiente bit menos significativo (LSB) del mensaje
    reemplazar bit LSB del DCT con el bit del mensaje
  end if
  insertar DCT en la imagen esteganográfica
terminar while

2.3.4.3 Esteganografía en Audio

En la esteganografía en audio, el mensaje secreto está incrustado en la señal de audio digitalizada que produce una ligera alteración de la secuencia binaria del archivo de audio correspondiente. Hay varios métodos disponibles, detallaremos brevemente cada uno.

LSB Coding

Es similar a la técnica LSB que se emplea en las imágenes. El proceso consiste luego de digitalizar una señal de audio y aplicar el proceso de cuantificación:



En esta técnica, el bit menos significativo de la secuencia binaria de cada muestra de archivo de audio digitalizado se reemplaza con el equivalente binario del mensaje secreto.

Por ejemplo, si queremos esconder la letra "*A*" en un archivo de audio de 16 bits, entonces cada bit menos significativo de 8 muestras consecutivas (de 16 bits de tamaño cada una) se reemplaza con cada bit de la secuencia binaria de la letra "*A*":

Codificación de fase

El sistema auditivo humano no puede reconocer el cambio de fase en una señal de audio. El método de codificación de fase explota este hecho. Esta técnica codifica los bits

Table 2.1: Ejemplo LSB Audio

Flujo de audio	"A" en binario	Flujo de audio con el mensaje oculto
1001 1000 0011 1100	**0**	1001 1000 0011 110**0**
1101 1011 0011 1000	**1**	1101 1011 0011 100**1**
1011 1100 0011 1101	**1**	1011 1100 0011 110**1**
1011 1111 0011 1100	**0**	1011 1111 0011 110**0**
1011 1010 0111 1111	**0**	1011 1010 0111 111**0**
1111 1000 0011 1100	**1**	1111 1000 0011 110**1**
1101 1100 0111 1000	**0**	1101 1100 0111 100**0**
1000 1000 0001 1111	**1**	1000 1000 0001 111**1**

del mensaje secreto como cambios de fase en el espectro de fase de una señal digital, logrando una codificación inaudible en términos de relación señal/ruido.

Mensaje de Eco

En este método, el mensaje secreto se incrusta en la señal de audio como un eco. Los tres parámetros de la señal son: amplitud, velocidad de decaimiento y desplazamiento de la señal original, que varían para representar un mensaje secreto codificado. Se establecen debajo del umbral del sistema auditivo humano para que el eco no se pueda percibir fácilmente.

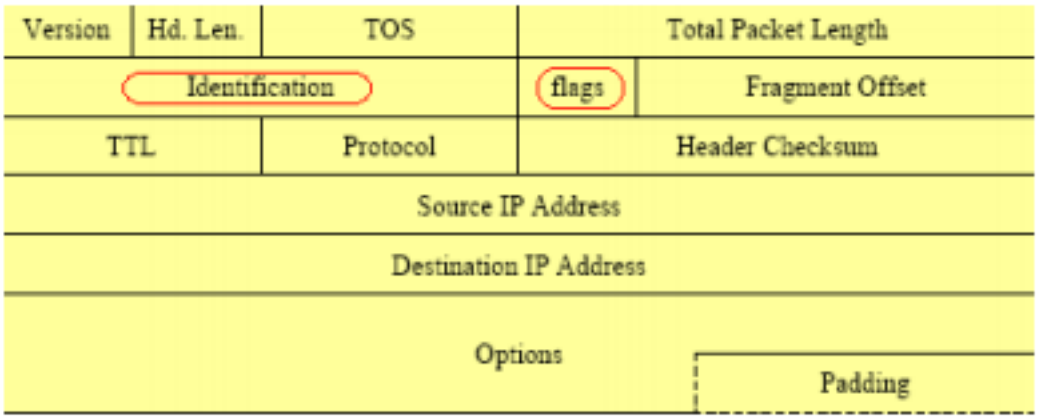
2.3.4.4 Esteganografía en Video

Los archivos de video generalmente se componen de imágenes y sonidos, por lo que la mayoría de las técnicas relevantes para ocultar datos en imágenes y audio también son aplicables a los medios de video. En el caso de esteganografía en video, el remitente envía el mensaje secreto al destinatario utilizando una secuencia de video como medio de cobertura. La clave secreta opcional 'K' también se puede usar durante la incrustación del mensaje secreto en los medios portadores para producir el "estego-video". Después de eso, el estego-video se comunica a través del canal público al receptor. En el extremo receptor, el receptor utiliza la clave secreta junto con el algoritmo de extracción para extraer el mensaje secreto del objeto stego.

2.3.4.5 Esteganografía en Protocolos de red

Este es otro enfoque de la esteganografía, que emplea datos de ocultación en el nivel de datagramas de la red en una red basada en TCP/IP como Internet. Network Covert Channel es el sinónimo de esteganografía de red. El objetivo general de este enfoque es

hacer que el datagrama stego no sea detectable por observadores de la red como sniffers, ni los Sistemas de detección de intrusiones (IDS), etc. En este enfoque, la información que se debe ocultar se coloca en el encabezado IP de un datagrama TCP/IP. Algunos de los campos de encabezado IP y encabezado TCP en una red IPv4 se eligen para ocultar datos. Primero, demostraremos cómo se pueden aprovechar los campos "Identification" y "flags" del encabezado Ipv4 con esta metodología.



Canal cubierto utilizando header "flags"

El tamaño del campo Flag es de 3 bits. Hay 3 banderas indicadas por cada bit. El primer bit está reservado. La segunda y la tercera, denotadas por DF (No fragmentar) y MF (Más Fragmentos) respectivamente. Un datagrama no fragmentado tiene toda la información de fragmentación cero (es decir, MF = 0 y Fragment Offset de 13 bits = 0) lo que da lugar a una condición de redundancia, es decir, DF (No fragmentar) puede llevar el asunto "0" o "1" Al conocimiento del tamaño máximo del datagrama.

Ahora, si el remitente y el destinatario tienen un conocimiento previo de la Unidad de Transferencia Máxima (MTU) de su red, entonces pueden comunicarse de forma encubierta entre sí utilizando el bit de indicador de DF del encabezado IP. La longitud del datagrama debe ser menor que la ruta MTU, de lo contrario el paquete se fragmentará y este método no funcionará.

La siguiente tabla muestra cómo se comunica el remitente 1 y 0 al destinatario mediante el uso del bit de indicador DF.

Este es un ejemplo de comunicación encubierta, ya que no hay forma de que los dispositivos de monitoreo de red como IDS o sniffers detecten la comunicación porque el datagrama de cobertura es un datagrama normal. Como la carga útil no se ha modificado, no hay forma de que un IDS o cualquier otro dispositivo de filtrado de contenido pueda

Table 2.2: Ejemplo Esteganografía de Red vía header Flags

Datagrama	**Campo "flag"**	**Fragmento de 13 bits**	**Observaciones**
1	0**1**0	00..00	Datagrama 1 ocultando "1"
2	0**0**0	00.00	Datagrama 2 ocultando "0"

reconocer esta actividad. La principal limitación de este enfoque es que ambas partes deben tener un conocimiento previo de la ruta MTU y el datagrama del remitente no debe estar más fragmentado en el camino.

Canal cubierto utilizando header "Identification"

El campo "Identification" de 16 bits en el encabezado Ipv4 se utiliza para identificar el paquete fragmentado de un datagrama IP. Si no hay una fragmentación del datagrama, entonces este campo de Identificación se puede usar para incrustar información específica del remitente.

Canal de comunicación cubierto utilizando header ISN

El encabezado TCP llamado ISN (Número de secuencia inicial) es otro candidato para para utilizar en esteganografía de la red. El número de secuencia inicial es un dígito de 32 bits generado durante el protocolo de enlace TCP / IP de tres vías entre el cliente y el servidor, que es el siguiente:

- El cliente envía un paquete TCP / IP con el indicador SYN. Este es un segmento donde el cliente especifica el número de puerto del servidor al que quiere conectarse y el ISN del cliente.
- El servidor responde con un paquete TCP/IP con el flag SYN y el flag ACK activado, con el número de confirmación que es el ISN + 1 del cliente. Este es el segmento 2.
- El cliente debe reconocer el paquete SYN de este servidor al enviar el paquete con el indicador ACK activado con el número de confirmación, que es ISN + 1 del servidor. Este es el segmento 3.

El gran espacio de direcciones de 32 bits del campo "Número de secuencia" se puede utilizar para el canal secreto. El remitente creará un paquete TCP / IP en el que el mensaje secreto se puede incrustar en el campo "Número de secuencia" y la parte receptora que escucha pasivamente extraerá los datos.

Otros candidatos para Esteganografía de red pueden ser el puerto origen y el checksum en los headers UDP, y el header Code de ICMP.

2.4 Web

2.4.1 Introducción

Los sitios web de todo el mundo se programan utilizando distintos lenguajes de programación que, si bien tienen vulnerabilidades conocidas y el desarrollador debe tener en cuenta, hay problemas fundamentales en Internet que pueden aparecer independientemente del lenguaje o framework elegido.

Estos problemas comunes a menudo aparecen en los CTF como desafíos de seguridad web donde el usuario necesita explotar un error para obtener algún tipo de privilegio o control sobre el sistema y así encontrar la bandera.

Así como en los capítulos anteriores hemos visibilizado los problemas más comunes de criptografía y esteganografía en CTF's, en éste haremos lo propio con los problemas WEB.

2.4.2 OWASP Top Ten

El Proyecto Abierto de Seguridad en Aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a permitir que las organizaciones desarrollen, adquieran y mantengan aplicaciones y APIs confiables. OWASP difunde gratuitamente herramientas y estándares de seguridad, libros de desarrollo seguro y testing de aplicaciones, presentaciones, hojas de trucos (cheatsheets), controles de seguridad estándar, bibliotecas, investigaciones, conferencias, etc.

Son los desarrolladores del *OWASP TOP Ten* [22], documento que menciona los diez riesgos de seguridad más importantes en aplicaciones web. Esta lista se publica y actualiza aproximadamente cada tres años.

Detallaremos estas diez vulnerabilidades del TOP Ten 2017, debido a que son las que se dan con mas frecuenta en los desafíos para la categoría Web en CTF's.

2.4.2.1 A1 - Inyección

Las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.

Casi cualquier fuente de datos puede ser un vector de inyección: variables de entorno, parámetros, servicios web externos e internos, y todo tipo de usuarios. Los defectos de inyección ocurren cuando un atacante puede enviar información dañina a un intérprete.

Estos defectos son muy comunes, particularmente en código heredado. Las vulnerabilidades de inyección se encuentran a menudo en consultas SQL, NoSQL, LDAP, XPath, comandos del SO, analizadores XML, encabezados SMTP, lenguajes de expresión, parámetros y consultas ORM. Los errores de inyección son fáciles de descubrir al examinar el código y los escáneres y fuzzers ayudan a encontrarlos.

Una inyección puede causar divulgación, pérdida o corrupción de información, pérdida de auditabilidad, o denegación de acceso. El impacto al negocio depende de las necesidades de la aplicación y de los datos.

2.4.2.2 A2 - Pérdida de Autenticación

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente).

Los atacantes tienen acceso a millones de combinaciones de pares de usuario y contraseña conocidas (debido a fugas de información), además de cuentas administrativas por defecto. Pueden realizar ataques mediante herramientas de fuerza bruta o diccionarios para romper los hashes de las contraseñas.

Los errores de pérdida de autenticación son comunes debido al diseño y la implementación de la mayoría de los controles de acceso. La gestión de sesiones es la piedra angular de los controles de autenticación y está presente en las aplicaciones. Los atacantes pueden detectar la autenticación defectuosa utilizando medios manuales y explotarlos utilizando herramientas automatizadas con listas de contraseñas y ataques de diccionario.

Los atacantes solo tienen que obtener el acceso a unas pocas cuentas o a una cuenta de administrador para comprometer el sistema. Dependiendo del dominio de la aplicación, esto puedo permitir robo de identidad y divulgación de información sensible.

2.4.2.3 A3 - Exposición de Datos sensibles

Muchas aplicaciones web y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o información personalmente identificable. Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo

fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.

En lugar de atacar la criptografía, los atacantes roban claves, ejecutan ataques Man in the Middle [23] o roban datos en texto plano del servidor, en tránsito, o desde el cliente. Se requiere un ataque manual pero pueden utilizase bases de datos con hashes que han sido hechas públicas para obtener las contraseñas originales utilizando GPUs.

En los últimos años, este ha sido el ataque de mayor impacto. El error más común es simplemente no cifrar los datos sensibles. Cuando se emplea criptografía, es común la generación y gestión de claves, algoritmos, cifradores y protocolos débiles. En particular algoritmos débiles de hashing para el almacenamiento de contraseñas. Para los datos en tránsito las debilidades son fáciles de detectar, mientras que para los datos almacenados es muy difícil. Ambos tienen una posibilidad de explotación muy variable.

Los fallos con frecuencia comprometen datos que deberían estar protegidos. Típicamente, esta información incluye información personal sensible como registros de salud, datos personales, credenciales y tarjetas de crédito, que a menudo requieren mayor protección, según lo definido por las leyes o reglamentos como el PIBR de la UE o las leyes locales de privacidad.

2.4.2.4 Entidades Externas XML

Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).

Los atacantes pueden explotar procesadores XML vulnerables si cargan o incluyen contenido hostil en un documento XML, explotando código vulnerable, dependencias o integraciones.

De forma predeterminada, muchos procesadores XML antiguos permiten la especificación de una entidad externa, una URI que se referencia y evalúa durante el procesamiento XML. Las herramientas SAST (Static Application Security Testing o de Testeo Estático de Seguridad en Aplicaciones) pueden descubrir estos problemas inspeccionando las dependencias y la configuración. Las herramientas DAST (Dynamic Application Security Testing o de Testeo Dinámico de Seguridad en Aplicaciones) requieren pasos manuales adicionales para detectar y explotar estos problemas. Los

testers necesitan ser entrenados para hacer estas pruebas, ya que no eran realizadas antes de 2017.

Estos defectos se pueden utilizar para extraer datos, ejecutar una solicitud remota desde el servidor, escanear sistemas internos, realizar un ataque de denegación de servicio y ejecutar otro tipo de ataques.

2.4.2.5 A5 - Pérdida de Control de Acceso

Las restricciones sobre lo que los usuarios autenticados pueden hacer no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, por ejemplo, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos o cambiar derechos de acceso y permisos.

La explotación del control de acceso es una habilidad esencial de los atacantes. Las herramientas SAST y DAST pueden detectar la ausencia de controles de acceso pero, en el caso de estar presentes, no pueden verificar si son correctos. Es detectable utilizando medios manuales o de forma automática en algunos frameworks que carecen de controles de acceso.

Las debilidades del control de acceso son comunes debido a la falta de detección automática y a la falta de pruebas funcionales efectivas por parte de los desarrolladores de aplicaciones. La detección de fallas en el control de acceso no suele ser cubierto por pruebas automatizadas, tanto estáticas como dinámicas.

El impacto técnico incluye atacantes anónimos actuando como usuarios o administradores; usuarios que utilizan funciones privilegiadas o crean, acceden, actualizan o eliminan cualquier registro.

2.4.2.6 A6 - Configuración de Seguridad Incorrecta

La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, ad hoc o por omisión (o directamente por la falta de configuración). Son ejemplos: S3 buckets abiertos, cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, frameworks, dependencias y componentes desactualizados.

Los atacantes a menudo intentarán explotar vulnerabilidades sin parchear o acceder a cuentas por defecto, páginas no utilizadas, archivos y directorios desprotegidos, etc. para obtener acceso o conocimiento del sistema o del negocio.

Configuraciones incorrectas de seguridad pueden ocurrir en cualquier nivel del stack tecnológico, incluidos los servicios de red, la plataforma, el servidor web, el servidor de

aplicaciones, la base de datos, frameworks, el código personalizado y máquinas virtuales preinstaladas, contenedores, entre otros. Los escáneres automatizados son útiles para detectar configuraciones erróneas, el uso de cuentas o configuraciones predeterminadas, servicios innecesarios y opciones heredadas.

Los defectos frecuentemente dan a los atacantes acceso no autorizado a algunos datos o funciones del sistema. Ocasionalmente, estos errores resultan en un completo compromiso del sistema. El impacto al negocio depende de las necesidades de la aplicación y de los datos.

2.4.2.7 A7 - Cross-Site Scripting (XSS)

Los XSS ocurren cuando una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta JavaScript en el navegador. Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (defacement) los sitios web, o redireccionar al usuario hacia un sitio malicioso.

Es un tipo de vulnerabilidad informática, que puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar. Existen tres formas usuales de XSS para atacar a los navegadores de los usuarios

- XSS Reflejado: la aplicación o API utiliza datos sin validar, suministrados por un usuario y codificados como parte del HTML o Javascript de salida. No existe una cabecera que establezca la Política de Seguridad de Contenido (CSP). Un ataque exitoso permite al atacante ejecutar comandos arbitrarios (HTML y Javascript) en el navegador de la víctima. Típicamente el usuario deberá interactuar con un enlace, o alguna otra página controlada por el atacante, como un ataque del tipo pozo de agua, publicidad maliciosa, o similar.

- XSS Almacenado: la aplicación o API almacena datos proporcionados por el usuario sin validar ni sanear, los que posteriormente son visualizados o utilizados por otro usuario o un administrador. Usualmente es considerado como de riesgo de nivel alto o crítico.

- XSS Basados en DOM: frameworks en JavaScript, aplicaciones de página única o APIs incluyen datos dinámicamente, controlables por un atacante. Idealmente, se debe evitar procesar datos controlables por el atacante en APIs no seguras. Los ataques

XSS incluyen el robo de la sesión, apropiación de la cuenta, evasión de autentificación de múltiples pasos, reemplazo de nodos DOM, inclusión de troyanos de autentificación, ataques contra el navegador, descarga de software malicioso, keyloggers, y otros tipos de ataques al lado cliente.

XSS es la segunda vulnerabilidad más frecuente en OWASP Top 10, y se encuentra en alrededor de dos tercios de todas las aplicaciones. Las herramientas automatizadas pueden detectar algunos problemas XSS en forma automática, particularmente en tecnologías maduras como PHP, J2EE / JSP, y ASP.NET.

2.4.2.8 A8 - Deserialización Insegura

Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.

Aplicaciones y APIs serán vulnerables si deserializan objetos hostiles o manipulados por un atacante. Esto da como resultado dos tipos primarios de ataques:

- Ataques relacionados con la estructura de datos y objetos; donde el atacante modifica la lógica de la aplicación o logra una ejecución remota de código que puede cambiar el comportamiento de la aplicación durante o después de la deserialización.
- Ataques típicos de manipulación de datos; como ataques relacionados con el control de acceso, en los que se utilizan estructuras de datos existentes pero se modifica su contenido. La serialización puede ser utilizada en aplicaciones para:
  - Comunicación remota e Inter-Procesos (RPC/IPC).
  - Protocolo de comunicaciones, Web Services y Brokers de mensajes.
  - Caching y Persistencia.
  - Bases de datos, servidores de caché y sistemas de archivos.

2.4.2.9 A9 - Componentes con vulnerabilidades conocidas

Los componentes como bibliotecas, frameworks y otros módulos se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor. Las aplicaciones

y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.

2.4.2.10 A10 - Registro y Monitoreo Insuficientes

El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos

CAPÍTULO 3

RSOLVER

3.1 Introducción

Esta sección está dedicada a nuestro primer objetivo, la herramienta de criptoanálisis desarrollada, RSolver. Para organizarla la hemos dividida en tres secciones.

La primer parte está destinada al trabajo de investigación realizado. Allí nos referiremos a las vulnerabilidades de RSA, que fueron la base de nuestro desarrollo, y a las aplicaciones actuales que realizan tareas similares.

En la segunda parte veremos el desarrollo de la herramienta en sí, desde las dependencias utilizadas, el modo de instalación, funcionamiento, ataques disponibles y el modo de uso con ejemplos.

Por último, compararemos nuestro software con los demás existentes y demostraremos por qué consideramos que el nuestro es el más completo y eficiente.

3.2 Trabajos existentes

3.2.1 Trabajos de investigación existentes

En esta sección veremos distintas bibliografías sobre implementación y criptoanálisis de RSA que utilizamos como base para desarrollar RSolver.

3.2.1.1 Twenty Years of Attacks on the RSA Cryptosystem

Twenty Years of Attacks on the RSA Cryptosystem [24] es el resultado de la investigación realizada por Dan Boneh de la Universidad de Stanford. La cual nos proporciona un recorrido de muchas potenciales vulnerabilidades en el sistema de cifrado RSA y como explotarlas. Entre los ataques que allí describe se encuentran el ataque de módulo común, el ataque de Wiener, el ataque de Boneh-Durfee, el ataque de Hastad y el ataque de Franklin-Reiter (vistos en el marco teórico).

3.2.1.2 Cryptanalysis of RSA and Its Variants

En este libro [25] Jason Hinek hace un estudio exhaustivo de los ataques algebraicos más conocidos en RSA y sus principales variantes, incluidos el RSA rebalanceado, RSA multi-primos y RSA multi-factores. Para cada ataque el autor presenta pruebas matemáticas si es posible o una justificación matemática para aquellos ataques que se basan en suposiciones. Además proporciona en algunos casos evidencia experimental.

3.2.1.3 Cryptanalytic Attacks on RSA

Song Yan en este estudio [26] cubre casi todos los principales ataques y defensas criptoanalíticas conocidas de RSA y sus variantes. Abarca ataques de factorización de enteros, ataques de logaritmo discreto, menciona ataques de cómputo cuántico, ataques de clave pública y privada y ataques de canal lateral.

3.2.2 Herramientas de software similares

En esta sección se detallan otras herramientas para el algoritmo RSA, algunas orientadas a CTF, que tienen similitudes a Rsolver.

3.2.2.1 RsaCTFTool

De las herramientas aquí nombradas, RsaCTFTool [27] es la mas similar a Rsolver y la más completa para ejercicios de CTF. Descifra datos de una clave pública débil e intenta recuperar la clave privada a partir de una serie de ataques de clave pública.

Posee tres funcionalidades básicas:

1. Atacar RSA a partir de una clave pública, intenta encontrar la clave privada mediante diversos modos de ataques.

2. Crear una clave pública en formato PEM a partir de n y e.

3. Extraer datos decimales a partir de una clave DER o PEM.

3.2.2.2 FeatherDuster

FearherDuster [28] es una herramienta automática de criptoanálisis modular. Intenta que el proceso de identificación y explotación de sistemas criptográficos débiles sea lo más simple posible. Usa internamente la librería Cryptanalib. Es una herramienta más general, ya que no sólo ataca RSA, sino que también permite encontrar vulnerabilidades en criptografía simétrica, ya sea cifrado por bloques o stream, con modos ECB, CBC, etc.

3.2.2.3 AttackRSA

AttackRSA [29] está dirigida a ejercicios de CTF. Tiene cinco tipos de ataques a RSA disponibles: Wiener, factorización de Fermat, Hastad, texto plano elegido, y módulo común. El ingreso de los datos de entrada es vía argumento.

3.2.2.4 Rsatool

Rsatool [30] es una herramienta para la generación de claves privadas RSA en formato PEM a partir de ingresar los datos decimales. Tiene dos formas de crear claves: A partir p y q, o bien a partir de de n y d.

No tiene la posibilidad de encontrar vulnerabilidades en la implementación, es simplemente una herramienta para la creación de claves RSA con valores customizados por el usuario.

3.2.2.5 Ctf Crypto

Ctf Crypto [31] es una herramienta que proporciona ataques para varios métodos criptográficos utilizados en CTFs. Utiliza Sage para la explotación de los mismos y tiene ataques de factorización, ataques de exponente público bajo y ataques de clave privada parcial, entre otros.

3.2.2.6 X RSA

X RSA [32] está destinada a ejercicios de RSA de CTFs y es desarrollada en Python. Presenta una lista de posibles ataques como también las variables de entrada de RSA

necesarias para explotarlo. Tiene una interfaz de selección de las opciones vía terminal que facilita su uso.

3.2.2.7 Rshack

Rshack [33] es una aplicación escrita en python que permite llevar a cabo algunos ataques al protocolo RSA además de ofrecer herramientas para manipular claves RSA. Implementa los ataques de Wiener, Hastad, Fermat, Bleichenbacher, de módulo común y de texto plano elegido. Provee herramientas para manipular archivos de claves PEM y realizar cifrado/descifrado RSA.

3.3 RSolver

Como mencionamos, los retos de criptografía son muy comunes en las competencias de seguridad CTF y muchos implican explotar una vulnerabilidad en un sistema de cifrado RSA mal implementado. Estas vulnerabilidades hacen posible factorizar un módulo RSA en un tiempo razonable o directamente calcular el texto plano (la bandera del reto).

A menudo, el flujo de trabajo para resolver uno de estos problemas de RSA implica:

- Averiguar qué vulnerabilidad está presente.
- Investigar cómo explotar esa vulnerabilidad.
- Escribir un programa/script que lo haga

Suponiendo que la vulnerabilidad no es revelada en el detalle del reto, cada uno de estos pasos puede tomar una cantidad significativa de tiempo. Además de eso, también está el tiempo dedicado a depurar la implementación del script de explotación, habitualmente referido como el exploit.

Para ayudar a reducir la cantidad de tiempo empleado en cada uno de estos pasos, hemos desarrollado RSolver, una herramienta de exploits RSA para los desafíos CTF relacionados con los criptosistemas RSA. Muchas de las vulnerabilidades conocidas, como el ataque de Wiener, el ataque de difusión de Hastad, el uso de primos de Fermat, el ataque de módulo común, ya están implementadas. Además, se incluyen varios métodos de factorización en números primos cada uno con sus propios casos de uso especiales. Si nuestra herramienta ya implementa el exploit requerido, el tiempo necesario para investigar, codificar y depurar una solución se reduce significativamente. Si no se conoce

de inmediato la vulnerabilidad de RSA o el exploit requerido, un usuario podría ejecutar una batería de exploits contra los textos cifrados y los datos clave de RSA en un intento por descubrir más información. Además, un problema de RSA puede ser vulnerable a más de un ataque, lo que puede ser desconocido tanto para el usuario como para el autor del problema.

No sólo la utilizamos para resolver ejercicios en algún evento, sino que también la usamos cuando tenemos que escribir retos para incluir en alguna competencia, por ejemplo en el ámbito de la Facultad de Informática de Universidad Nacional de La Plata, participamos en las cátedras Desarrollo Seguro de Aplicaciones [34] e Introducción a la Ciberseguridad [35], y es durante el trancurso de las cursadas que utilizamos RSolver para verificar los desafíos que se incluyen en los CTFs que organizamos para los alumnos .

3.4 Formas de uso

La herramienta se presenta en varios formatos: se la puede utilizar cómo una librería (módulo) de Python 3 e incluir dentro de otro software por ejemplo, o bien se la puede ejecutar por línea de comando en sistemas Linux. Además, está presente en formato web en "Syper CTF Tools", herramienta que describimos en el siguiente capítulo.

3.5 Licencia

Como expusimos en los objetivos, el software desarrollado es libre para ser usado, copiado o modificado. Es por ello que se elige la licencia GNU GPL versión 3 para que aplique a este proyecto. Para llevar a cabo esto, se utilizó como plataforma de distribución del código fuente a GitHub, utilizando git como sistema de control de versiones.

3.6 Python 3

Se utilizó Python 3 para el desarrollo de RSolver. Es un lenguaje de programación open source, multiplataforma (Linux, Windows, Mac, Android), y simple de aprender. Además de ser libre y gratuito, es mundialmente usado, contando con una gran comunidad de usuarios que mantienen innumerables fuentes de información que facilitan su utilización. La elección no se basa únicamente por el core del lenguaje, sino que se apoya en la

innumerable cantidad de librerías que existe, debido a que cualquiera puede aportar una de ellas, e incluso RSolver se facilita a la comunidad como una de ellas.

Además, Python 3 es uno de los preferidos en la industria de la seguridad informática y seguridad de la información, por ejemplo para:

- Desarrollo de exploits
- Redes
- Debugging
- Criptografía
- Ingeniería Inversa
- Fuzzing
- Web
- Análisis Forense
- Análisis de Malware

3.7 Dependencias e Instalación

3.7.1 Dependencias del sistema

Si bien cada exploit podría utilizar sus propias dependencias, el core de RSolver requiere las siguientes librerías instaladas en el sistema base:

- MPFR: Librería MPFR desarrollada en C, para cálculos de punto flotante de precisión múltiple con redondeo correcto.
- MPC: Librería para aritmética de números complejos con precisión arbitrariamente alta.
- Sagemath: SageMath (anteriormente Sage o SAGE, "System for Algebra and Geometry Experimentation") es un sistema de álgebra computacional con características que cubren muchos aspectos de las matemáticas, incluyendo álgebra, combinatoria, teoría de grafos, análisis numérico, teoría de números, cálculo y estadística. Esta dependencia es opcional, pero altamente recomendado.

3.7.2 Dependencias de Python

Además, se utilizan las siguientes librerías de Python:

- *pyCrypto*: Módulo base de criptografía para Python.
- *sympy*: SymPy es una biblioteca de Python para matemáticas simbólicas. Su propósito es llegar a ser un sistema de álgebra por computadora (CAS) completo manteniendo el código tan simple como sea posible para poder ser legible y extensible de manera fácil.
- *gmpy*: GMPY (General MultiPrecision arithmetic for Python) gmpy es un módulo de Python con código C que proporciona acceso a la biblioteca aritmética de precisión múltiple GMP (o MPIR).
- *termcolor*: Escribir en colores en la terminal.
- *argparse*: El módulo nativo de Python argparse facilita la escritura de interfaces de línea de comandos. El programa define qué argumentos requiere, y argparse descubrirá cómo analizarlos desde sys.argv. El módulo argparse también genera automáticamente mensajes de ayuda y uso y emite errores cuando los usuarios le dan argumentos inválidos al programa.
- *glob*: Manejo de archivos del sistema.
- *os*: Manejo del sistema operativo.
- *signal*: Establece controladores para eventos asincrónicos.
- *time*: Este módulo proporciona varias funciones relacionadas con el manejo de tiempo.
- *subprocess*: Gestión de subprocesos.
- *binascii*: Convertir entre binario y ASCII.
- *fractions*: Manejo de números racionales.
- *pprint*: Imprimir datos con formato.
- *traceback*: Manejo de excepciones.
- *logging*: Este módulo define funciones y clases que implementan un sistema flexible de registro de eventos para aplicaciones y bibliotecas.
- *base64*: Codificación y decodificación de base64.

3.7.3 Instalación

Una vez instaladas las dependencias, instalamos con el gestor de módulos Pip:

pip3 install rsolver

Esto instala tanto el módulo de Python cómo el programa por línea de comando.

3.8 Funcionamiento

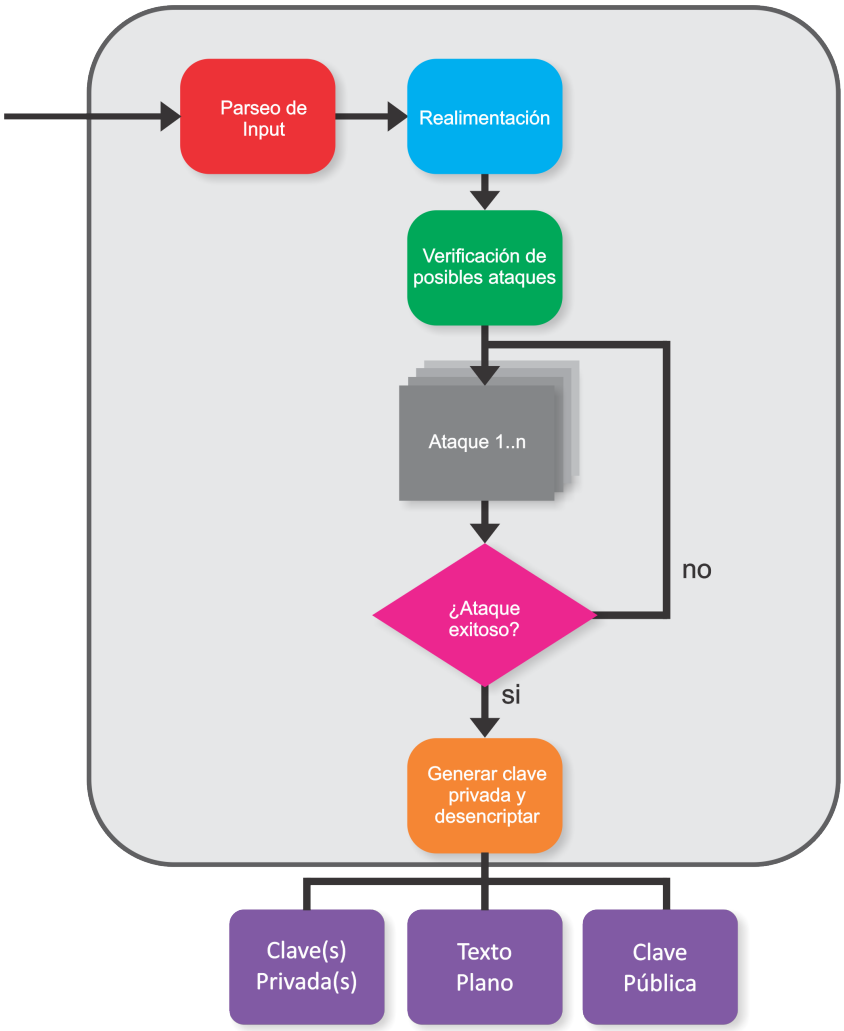
RSolver es una herramienta para la resolución de retos de RSA, que además provee funciones auxiliares para la implementación del protocolo. Las funciones principales de Rsolver son:

- Evaluar si existen debilidades de implementación en un criptosistema RSA dado.
- En caso de que se verifique una mala implementación, Rsolver disparará una batería de ataques e intentará recuperar la clave privada utilizada.
- Descifrar del texto cifrado (por lo general la bandera del reto) una vez obtenida la clave privada.
- Extra: Generar claves públicas y privadas con valores establecidos por el usuario en distintos formatos.

La principal ventaja de RSolver es que nos brinda mucha versatilidad, ya que permite como entrada cualquier parámetro presente en la implementación RSA (p, q, n, e, d, $\phi(n)$, dp, dq, qinv) y en múltiples formatos (archivo, decimal, hexadecimal, base 64), además de claves públicas (en formato PEM o DER), o partes de claves privadas (los primeros bytes o los últimos), permitiendo poder encarar cualquier tipo de reto sin importar el formato en el que se presenten los datos, sin preocuparse por la transformación a los valores estándar.

Lo primero que hace la herramienta es el **parseo los datos ingresados**: cualquier dato participante en el proceso de RSA se ingresa al sistema. Se verifica si estos datos son suficientes para generar una clave privada o pública. En caso de que no sea suficiente, la herramienta intenta **retroalimentarse**, es decir, intentar conseguir los datos necesarios restantes en función de los ingresados. Por ejemplo, como vimos en la explicación de RSA, $n = p * q$, si se ejecutó la herramienta brindándole los datos p y n, se puede conseguir q dividiendo n/p.

Figura 3.1: Flujo de ejecución de Rsolver



Si esto aún no es suficiente para generar la clave privada, se entra en la etapa de **criptoanálisis**, dónde se verifica para cada ataque disponible si se cumplen las condiciones necesarias para utilizarlo, por ejemplo, para el ataque de factores comunes, se deben tener factores primos en común en las claves públicas ingresadas, o para el ataque de Fermat, los factores primos p y q deben estar cerca.

Luego, se ejecutan los ataques que sí cumplen las condiciones, y si alguno logra explotar la vulnerabilidad, se genera la clave privada, y, en caso de que haya sido ingresado el texto cifrado, se obtiene el texto plano. Además, de manera informativa, y para fomentar el aprendizaje, se muestra en pantalla referencias con una explicación sobre los ataques que tuvieron éxito.

3.8.1 Ataques implementados

Alguno de los ataques que realiza RSolver son:

Wiener: Implementación del ataque de Wiener, que es factible cuando d es pequeño.

Boneh and Durfee: Es una mejora del algoritmo de Wiener, que funciona cuando el exponente privado d es chico comparado con el módulo N, más precisamente cuando $d < N^0.292$.

Módulo común: Si se tienen dos textos cifrados del mismo mensaje M que fueron cifrados con diferente exponente pero con el mismo módulo n, es fácil recuperar el mensaje M si el $mcd(e1, e2) = 1$ y $mcd(ct2, n) = 1$

e es 1: Si el exponente público es 1, $M = c$ por lo que no se cifra.

Primo común: Si se tienen dos claves públicas, que comparten un módulo en común, es decir $n1 = p * q$ y $n2 = p * r$, entonces es posible factorizar ambas claves en un tiempo razonable.

FactorDB: Busca si el módulo N se encuentra factorizado en la web Factordb [36].

Get p,q and e from dp, dq y qinv: Si se tiene dp, dq y qinv, el script consigue p, q y e.

Hastad: Implementación del algoritmo Teorema Chino del resto, si se tienen k claves públicas diferentes y k textos cifrados con esas claves, pero todas con el exponente público $e = k$, es posible recuperar el mensaje en texto plano.

Low q: Si uno de los factores es chico, es posible factorizar haciendo fuerza bruta sobre los primeros primos.

Multi primos: Ataque en caso de que n sea factorizable por más de dos factores.

N es primo: Esta es una mala implementación de RSA, sucede en caso de que usen el módulo n primo, en vez de la multiplicación de dos primos.

Se tiene dp: Si se tiene dp ($dp = pmod(q-1)$), n y e, es posible recuperar p y q en tiempo razonable.

Yafu: Utilización del software de factorización Yafu para tratar de factorizar el módulo.

Fermat: Ataque posible cuando se utilizan p y q con valores cercanos.

Multi primos Yafu: Otra técnica es usar el software de factorización Yafu para encontrar múltiples factores.

Primos de Mersenne: Cuando se utilizan los primos de Mersenne más grandes conocidos.

Cómo ejemplo, utilizaremos el reto llamado *RSA Returns* del CTF *easyCTF* del año 2018. En él, la descripción del reto nos dice lo siguiente:

It's the return of everyone's favorite cryptosystem! Crack it for another flag. Help me decipher file.

n = 963782884382350040791768766444132778471460595279483101846709450816614079025851585568165378868719236326249917881267528484629398894856832307302995971433129

e = 65537

c = 744372384092422293657230440929981477879274068697788092531354927993093987582083018137886004402518974542009040817374858469786477527000745005045012289929736

En el detalle de este ejercicio, podemos ver que n y e conforman la clave pública, mientras que c es el mensaje encriptado, generado con dicha clave. El objetivo del reto es desencriptar c, para lo cuál necesitamos conseguir la clave privada a partir de la pública. El problema, consiste en la factorización del módulo n para conseguir los factores primos p y q, necesarios para generar la clave privada.

3.8.2 Rsolver - Línea de comando

3.8.2.1 Línea de comandos

En la versión de línea de comandos de Rsolver, los datos de entrada pueden ser ingresados de dos maneras distintas:

Vía argumentos: la primer manera, es enviar las variables de entrada cómo argumento en la llamada al programa:

Figura 3.2: Ejecución de Rsolver en línea de comandos con argumentos

```
$ rsolver -n 9637828843823500407917687664441327784714605
952794831018467094508166140790258515855681653788687192 36
326249917881267528484629398894856832230730295971433129
-e 65537 -c 7443723840924222936572304409299814778792740686977880925313549279930939875820830181378860044025189745
420090408173748584697864775270007450050450122899297 36
```

Vía archivo: muchas veces, los valores que nos dan son muy largos para ser enviados vía argumento, al intentar recibimos el error "La lista de argumentos es demasiado larga", por ello, otra forma de utilizarlo, es poner las variables en un archivo de texto, por ejemplo ejercicio.txt:

Figura 3.3: Archivo de texto con los datos de entrada

Y luego llamamos al programa con el argumento *–inputfile*:

Figura 3.4: Ejecución de Rsolver en línea de comando con archivo de entrada

```
$ rsolver --inputfile ejercicio.txt
```

3.8.2.2 Módulo de Python 3

Para resolver el ejemplo utilizando el módulo de Python incluímos la librería rsolver y creamos el objeto Rsolver pasándole como argumento el tiempo máximo de espera para cada script (0 es sin límite).

Luego, le agregamos al objeto las variables conocidas n, e y c con los métodos *addn*, *adde* y *addc* respectivamente.

Una vez que se le dió al objeto todos los datos conocidos, utilizamos la función "crack"

Figura 3.5: Rsolver como librería de Python 3

```
>>> import rsolver
>>> rs = rsolver.Rsolver(0)
OUTPUT FOLDER: output_06
>>> rs.addn(963782884382350040791768766444132778471460595279483101846709450816614079025851585568165378868719236326249917881267528484629398894856832230730295971433129)
>>> rs.adde(65537)
>>> rs.addc(744372384092422293657230440929981477879274068697788092531354927993093987582083018137886004402518974542009040817374858469786477527000745005045012289929736)
>>> rs.crack()
```

3.8.3 Enriquecimiento de los datos de entrada

Lo primero que intenta hacer Rsolver, es agregar más información con los datos de entrada. Lo único útil que puede hacer en este caso, al tener n y e, es generar la clave pública en formato PEM.

3.8.4 Verificación de posibles ataques

Luego, la herramienta evalúa los datos de entrada que se poseen y selecciona los scripts de ataques que serían factibles disparar.

Uno de los ataques que logra resolver el reto del ejemplo es factorizar n utilizando la página "FactorDB". Dicha web contiene factores primos para un número compuesto dado. El script entonces, encuentra los factores primos p y q, necesarios para la generación de la clave privada.

3.9 Herramientas

Esta sección está destinada a la comparación de Rsolver con herramientas similares.

Primero, compararemos mediante tablas las características inherentes al software: sus funciones principales, los valores de entrada que aceptan y los ataques que poseen disponibles para el crackeo. Luego, basándonos en métodos de investigación empíricos,

compararemos la eficacia de nuestro sistema con los demás. Utilizaremos una forma sistemática, disciplinada, cuantificable y controlada de evaluar nuestra propuesta frente a otras que ya existen para conocer bajo que criterios se logra una mejoría.

Los métodos de investigación empíricos se han convertido en los últimos años en una manera efectiva para evaluar la eficacia de las herramientas y técnicas propuestas en los campos de la Ingeniería del Software y los Sistemas de Información.

3.9.1 Comparación con otras herramientas

Ahora veremos con distintas tablas, las comparaciones de Rsolver con los otros softwares investigados, para entender sus características y las diferencias entre cada uno.

Figura 3.6: Comparación según los datos de entrada

Parámetro / Formato	**Tipo**	**RSolver**	**RsaCTFTool**	**AttackRSA**	**X-RSA**	**FeatherDuster**
p decimal	argumento	X	X	X	X	X
p hex	argumento	X				
q decimal	argumento	X	X	X	X	X
q hex	argumento	X				
n decimal	argumento	X	X	X	X	X
n hex	argumento	X				
e decimal	argumento	X	X	X	X	X
e hex	argumento	X				
phi decimal	argumento	X			X	
phi hex	argumento	X				
dp decimal	argumento	X			X	
dp hex	argumento	X				
dq decimal	argumento	X			X	
dq hex	argumento	X				
qinv decimal	argumento	X				
qinv hex	argumento	X				
PEM	archivo	X	X			X
input text	archivo	X				X
partial private key	archivo	X				
partial private key	argumento	X				
c decimal	argumento	X				
c hex	argumento	X				
c base64	archivo	X				
c base64	argumento	X				

Figura 3.7: Comparación según su funcionalidad

	Rsolver	RsaTool	RsaCTFTool	FeatherDuster	AttackRSA	Ctf-Crypto	X-RSA
Generar claves	X	X	X				
Ataques RSA	X		X	X	X	X	X
Extraer de clave PEM	X		X				

3.9.2 Resultado de pruebas

De un lote de 50 ejercicios extraídos, de distintos CTF's reconocidos a nivel mundial, nos encontramos con que:

- RSolver resolvió 48 de los 50, con un sólo comando, sin transformaciones de formato.
- RsaCTFTool resolvió 41, de los cuáles hubo que crear la clave pública, por lo que llevó dos ejecuciones en la mayoría de los casos.
- AttackRSA resolvió 19. también teniendo que transformar las variables a decimal.
- CTF-Crypto resolvió 9.

CAPÍTULO 4

SYPER CTF TOOLS

Este capítulo está destinado al desarrollo de nuestro segundo objetivo, SYPER CTF Tools; una plataforma web para equipos de CTFs que integra múltiples herramientas útiles para estas competencias.

Con el objetivo de no reinventar la rueda (DRY, Dont Repeat Yourself) utilizamos *Wooey* [37] como base de nuestro trabajo, la cuál permite el acceso rápido e intuitivo a las herramientas vía web.

4.1 SYPER CTF Tools

SYPER CTF Tools es una personalización de Wooey, viniendo instalado en el mismo, más de 25 herramientas ampliamente utilizadas.

Entre las ventajas más destacables, tenemos que es colaborativo, cuenta con gran facilidad para agregar nuevos scripts, es multi-usuario, multi-hilos y permite ver los resultados de ejecuciones anteriores, permitiendo el acceso a un abanico de programas útiles para las competencias de manera online y sin tener que recordar los parámetros obligatorios puesto que la interfaz web nos brinda de manera simple esta información.

Resumiendo, para SYPER CTF Tools desarrollamos scripts estilo wrappers para lanzar diferentes herramientas de línea de comando con sus respectivos argumentos válidos y las integramos en una versión personalizada de Wooey.

4.1.1 Wooey

Wooey es una aplicación realizada en y para el framework de Python 3 *Django* [38]. Provee una interfaz web simple para ejecutar desde la misma scripts de Python que utilizan interfaces de línea de comandos. Transforma las CLI (interfaces de línea de comando) en GUI (interfaces gráficas de usuario), es decir, convierte los argumentos que reciben estos scripts en inputs de formularios HTML para facilitar su configuración y ejecución, permitiendo que una persona sin experiencia pueda utilizar dichos scripts vía un navegador web.

4.2 Herramientas incluidas

En esta sección veremos las diferentes herramientas de línea de comandos que desarrollamos su adaptador en script de Python e integramos en SYPER CTF Tools.

4.2.1 Herramientas de Esteganografía

4.2.1.1 Steghide [39]

Steghide es un programa para esteganografía que permite esconder datos en archivos. Acepta varios formatos de archivos de imágenes y sonido. Este proceso resulta imperceptible para análisis estadísticos de primer orden.

4.2.1.2 zsteg [40]

Zsteg detecta datos escondidos usando esteganografía en archivos de imágenes PNG y BMP.

4.2.1.3 stegoVeritas [41]

StegoVeritas es una herramienta automática para realizar ataques de fuerza bruta sobre bytes menos significativos, transformar, y extraer metadatos en imágenes.

4.2.1.4 stegdetect [42]

Stegdetect analiza archivos de imágenes en búsqueda de contenido esteganográfico. Corre tests estadísticos para determinar si hay contenido esteganográfico e intenta encontrar que sistema ha sido utilizado para esconder la información.

4.2.1.5 stegbreak [43]

Stegbreak lanza un ataque de fuerza bruta por diccionario sobre archivos JPG asumiendo que contienen información oculta embebida con el programa jphide.

4.2.1.6 jphide/jpseek [44]

Jphide permite esconder un archivo en un archivo JPEG. El objetivo no es simplemente ocultar un archivo, sino hacerlo de tal manera que sea imposible probar que el archivo host contiene un archivo oculto. Dada una imagen visual típica, una tasa de inserción baja (menos del 5%) y la ausencia del archivo original, no es posible concluir con una certeza valiosa de que el archivo host contenga datos insertados. A medida que aumenta el porcentaje de inserción, la naturaleza estadística de los coeficientes jpeg difiere de "normal" en la medida en que aumenta la sospecha. Luego con Jpseek y proporcionando la password utilizada, se puede recuperar el archivo escondido.

4.2.1.7 jsteg [45]

Jsteg permite esconder datos dentro de archivos JPEG. Esto es logrado copiando cada bit de los datos que queremos esconder dentro de los bits menos significativos de la imagen. La cantidad de datos que pueden ocultarse depende del tamaño del archivo JPEG; toma cerca de 10-14 bytes del JPEG esconder 1 byte de información.

4.2.1.8 openstego [46]

Openstego es una aplicación para esteganografía que provee dos funcionalidades:

- Ocultamiento de datos: Permite esconder cualquier tipo de dato dentro de otro archivo.
- Marca de agua: Puede agregar una marca de agua a archivos con una firma invisible. Puede ser usado para detectar copias de archivo no autorizadas.

4.2.1.9 outguess [47]

Outguess es una herramienta esteganográfica que permite al usuario embeber información en una imagen. Almacena los datos a ocultar modificando bits redundantes para que no cause cambios perceptibles en la imagen portadora.

4.2.1.10 spectrology [48]

Spectrology es una herramienta que permite esconder imágenes en archivos de sonido, encodeandolas en su espectrograma.

4.2.1.11 stegano [49]

Stegano es un módulo de python que permite realizar esteganografía utilizando técnicas de LSB.

4.2.1.12 cloacked-pixel [50]

Cloacked-pixel es una herramienta hecha en python que permite realizar y detectar esteganografía en imágenes utilizando técnicas de LSB. Permite también cifrar los datos antes de insertarlos.

4.2.1.13 MP3Stegz [51]

Mp3Stegz es un programa que aplica algoritmos esteganográficos en archivos mp3. Esta herramienta mantendrá el tamaño y la calidad de audio del archivo mp3 original. El mensaje oculto está comprimido (zlib) y cifrado.

4.2.1.14 gifshuffle [52]

El programa gifshuffle se usa para ocultar mensajes en imágenes GIF al mezclar el mapa de colores, lo que deja la imagen visiblemente sin cambios. gifshuffle funciona con todas las imágenes GIF, incluidas aquellas con transparencia y animación, y además proporciona compresión y cifrado del mensaje oculto.

4.2.1.15 StegCracker [53]

Con StegCracker se pueden realizar ataques de fuerza bruta a imágenes para recuperar archivos ocultos mediante esteganografía.

4.2.2 Herramientas Vulnerabilidades Web

Para llevar a la práctica la búsqueda de vulnerabilidades especificadas en el apartado de marco teórico web, incluímos conectores para las siguientes herramientas:

4.2.2.1 nmap [54]

Nmap ("Network Mapper") es una utilidad gratuita y de código abierto para el descubrimiento de redes y auditoría de seguridad. Nmap utiliza paquetes IP de distintas formas para determinar qué hosts están disponibles en la red, que servicios ofrecen (detallando nombre de aplicación y versión) qué sistemas operativos (y versiones de SO) están ejecutando, que tipo de filtro de paquetes o firewall están en uso y docenas de otras características. Fue diseñado para escanear rápidamente redes grandes, pero funciona bien contra hosts individuales.

4.2.2.2 sublist3r [55]

Sublist3r es una herramienta hecha en Python diseñada para enumerar subdominios de sitios web usando OSINT. Ayuda a pentesters y bug hunters a recopilar y reunir subdominios para el dominio que se está analizando. Sublist3r enumera subdominios utilizando muchos motores de búsqueda como Google, Yahoo, Bing, Baidu y Ask. Sublist3r tambien enumera subdominios utilizando Netcraft, Virustotal, ThreatCrowd, DNSdumpster y ReverseDNS.

4.2.2.3 nikto [56]

Nikto es un escáner de servidor web de código abierto (GPL) que realiza pruebas exhaustivas contra servidores web para detectar múltiples elementos, incluidos más de 6700 archivos/programas potencialmente peligrosos, verifica versiones desactualizadas de más de 1250 servidores, y problemas específicos de versiones en mas de 270 servidores. También verifica elementos de configuración del servidor, como la presencia de múltiples archivos index, malas configuraciones en el servidor HTTP e intentará identificar los servidores web y el software instalados. Los elementos de escaneo y los complementos se actualizan con frecuencia y se pueden actualizar automáticamente.

Nikto no está diseñado como una herramienta sigilosa. Escaneará un servidor web en el menor tiempo posible, y dejará rastros en archivos de registro o en un IPS / IDS. Sin embargo, posee soporte para algunos métodos anti-IDS.

No todos los chequeos son un problema de seguridad, aunque la mayoría lo son, hay algunos que son verificaciones de tipo "solo información" que buscan cosas que pueden no tener una falla de seguridad, pero el webmaster o el ingeniero de seguridad pueden no saber que están accesibles en el servidor. Estos elementos suelen estar marcados adecuadamente en la información devuelta por el programa.

4.2.2.4 WPScan [57]

WPScan es un escáner de vulnerabilidades para WordPress de caja negra y gratuito, escrito por profesionales de seguridad y administradores de blogs para probar la seguridad de sus sitios.

Al usar WPScan, se puede escanear un sitio web con una instalación de WordPress en búsqueda de vulnerabilidades tanto en el núcleo de WordPress como en sus complementos y temas. También se puede detectar si existe alguna contraseña débil o si hay problemas en las configuraciones de seguridad. La base de datos de wpvulndb.com es usada para chequear versiones de software vulnerable, el equipo de WPScan se encarga de mantenerla actualizada.

4.2.2.5 DIRB [58]

DIRB es un escáner de contenido web. Busca objetos web existentes (y/o ocultos). Básicamente funciona lanzando un ataque de diccionario contra un servidor web y analizando la respuesta. DIRB viene con un conjunto de listas de palabras de ataque preconfiguradas para un uso fácil, pero también se pueden usar listas de palabras personalizadas. Además, a veces se puede usar DIRB como un escáner CGI clásico. El propósito principal de DIRB es ayudar en la auditoría profesional de aplicaciones web. Especialmente en pruebas relacionadas con la seguridad. Cubre algunos agujeros no cubiertos por los escáneres de vulnerabilidad web clásicos además, DIRB busca objetos web específicos que otros escáneres CGI genéricos no pueden buscar.

4.2.2.6 Gobuster [59]

Gobuster es una herramienta desarrollada en GO usada para realizar ataques de fuerza bruta contra URIs (directorios y archivos) en sitios web, subdominios DNS y nombres de virtual host en servidores web.

4.2.2.7 sqlmap [60]

Sqlmap es una herramienta para pruebas de penetración de código abierto que automatiza el proceso de detección y explotación de fallas de inyección SQL y toma de control de servidores de bases de datos. Viene con un potente motor de detección, provee muchas funciones y una amplia gama de configuraciones que abarcan desde el fingerprinting hasta la obtención de datos de la base de datos, el acceso al sistema de

archivos subyacente y la ejecución de comandos en el sistema operativo a través de conexiones fuera de banda.

4.2.2.8 fimap [61]

fimap es una herramienta de python que puede encontrar, preparar, auditar, explotar e incluso "googlear" automáticamente para detectar errores de inclusión de archivos locales y remotos en aplicaciones web. Fimap se asemeja a algo como Sqlmap pero solo para errores LFI / RFI en lugar de inyección SQL.

4.2.2.9 wfuzz [62]

Wfuzz se creó para facilitar la tarea en las evaluaciones seguridad de aplicaciones web y se basa en un concepto simple: reemplaza cualquier referencia a la palabra clave FUZZ por el valor de un payload dado.

Un payload en Wfuzz es una fuente de datos.

Este simple concepto permite inyectar cualquier entrada en cualquier campo de una solicitud HTTP, lo que permite realizar ataques de seguridad web complejos en diferentes componentes de aplicaciones web como: parámetros, autenticación, formularios, directorios / archivos, encabezados, etc.

4.2.2.10 wapiti [63]

Wapiti es una aplicación libre y de código abierto de línea de comandos que permite auditar la seguridad de sitios web o aplicaciones web. Realiza escaneos de "caja negra" (no estudia el código fuente) de la aplicación web buscando scripts y formularios donde pueda inyectar datos. Una vez que obtiene la lista de URL, formularios y sus entradas, Wapiti actúa como un fuzzer, inyectando payloads para detectar si es vulnerable.

4.2.2.11 Commix [64]

Commix, es una herramienta automatizada que puede ser utilizada por desarrolladores web, pentesters o incluso investigadores de seguridad para realizar pruebas en aplicaciones web en búsqueda de bugs, errores o vulnerabilidades relacionadas con ataques de inyección de comandos.

Al utilizar esta herramienta, resulta muy fácil encontrar y explotar una vulnerabilidad de inyección de comandos en un determinado parámetro vulnerable o encabezado HTTP.

4.2.2.12 hydra [65]

Cuando se necesita usar la fuerza bruta para romper un servicio de autenticación, Hydra suele ser la herramienta elegida. Puede realizar ataques rápidos de diccionario contra más de 50 protocolos, incluidos telnet, ftp, http, https, smb, varias bases de datos y mucho más.

4.3 Trabajando con los scripts

4.3.1 Agregando scripts

Los scripts se pueden agregar a través de la interfaz web de administrador de Django.

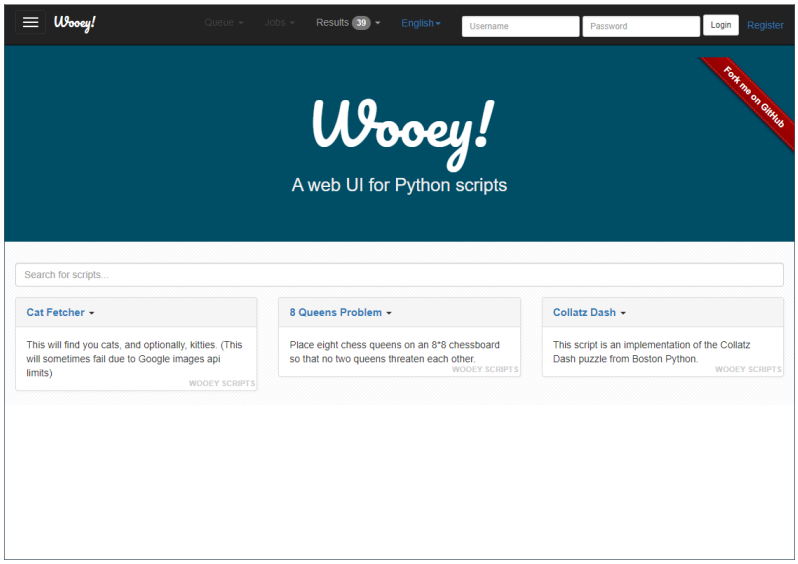
Se debe definir la clase argparse con todos los parámetros posibles que recibe el script para su ejecución declarando si son opcionales u obligatorios. Por ejemplo:

```
#!/usr/bin/env python

__author__ = 'Chris Mitchell'

import argparse
import os
import socket
import sys
from urllib import FancyURLopener
from apiclient import discovery

description = """
This will find you cats, and optionally, kitties.
"""

socket.setdefaulttimeout(10)

parser = argparse.ArgumentParser(description = description)
parser.add_argument('--count', help='The number of cats to find (max:
    10)', type=int, default=1)
parser.add_argument('--kittens', help='Search for kittens.', action='
    store_true')
parser.add_argument('--breed', help='The breed of cat to find', type=
    str, choices=('lol', 'tabby', 'bengal', 'scottish', 'grumpy'))

# Start FancyURLopener with defined version
```

```
24 class MyOpener(FancyURLopener):
25     version = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv
       :1.8.1.11)Gecko/20071127     Firefox/2.0.0.11'
26
27 myopener = MyOpener()
28
29 def main():
30     args = parser.parse_args()
31     searchTerm = 'kittens' if args.kittens else 'cats'
32     cat_count = args.count if args.count < 10 else 10
33     if args.breed:
34         searchTerm += ' {0}'.format(args.breed)
35
36     # Notice that the start changes for each iteration in order to
       request a new set of images for each loop
37     service = discovery.build('customsearch', 'v1', developerKey=os.
       environ.get('GOOGLE_DEV_KEY'))
38     cse = service.cse()
39     search_kwrds = {
40         'q': searchTerm,
41         'cx': os.environ.get('GOOGLE_CX'),
42         'fileType': 'jpg',
43         'imgType': 'photo',
44         'num': cat_count,
45         'searchType': 'image'
46     }
47     request = cse.list(**search_kwrds)
48     response = request.execute()
49     for item in response.get('items', []):
50         url = item.get('link')
51         filename = url.split('/')[-1]
52         try:
53             myopener.retrieve(url, filename)
54         except IOError:
55             continue
56
57
58 if __name__ == "__main__":
59     sys.exit(main())
```

4.3.2 Listando scripts

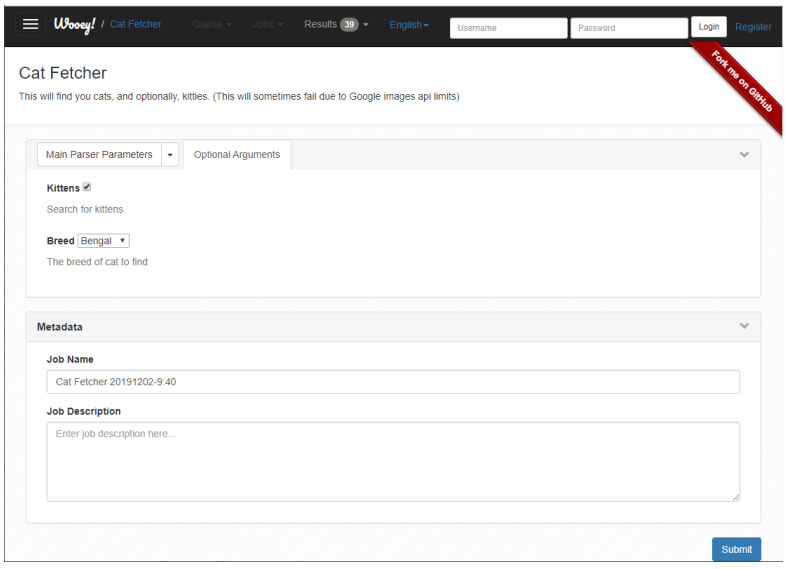
Figura 4.1: Listando scripts



Los scripts pueden ser accedidos vía la página de inicio o por el cuadro de búsqueda de scripts.

4.3.3 Ejecutando scripts

Figura 4.2: Ejecutando scripts

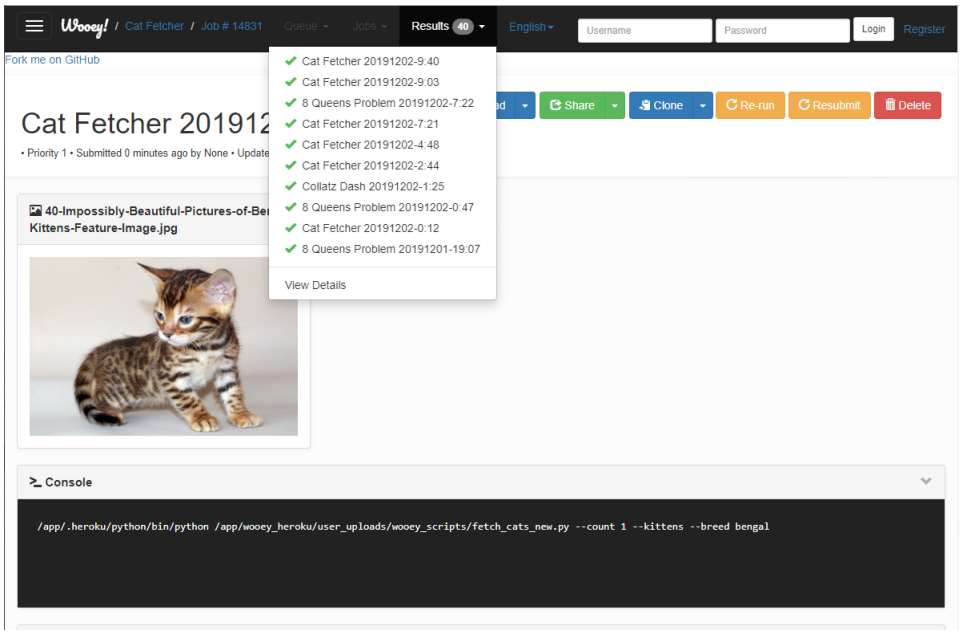


Una vez seleccionado el script que se quiere ejecutar, se deben ingresar los parámetros obligatorios y los opcionales deseados.

Cuando se envía el formulario, se crea un nuevo job en lista de espera para ser ejecutado.

4.3.4 Observando resultado

Figura 4.3: Observando resultado



Ejecutado el job se puede acceder al resultado del mismo desde el menú desplegable "Results" del panel superior.

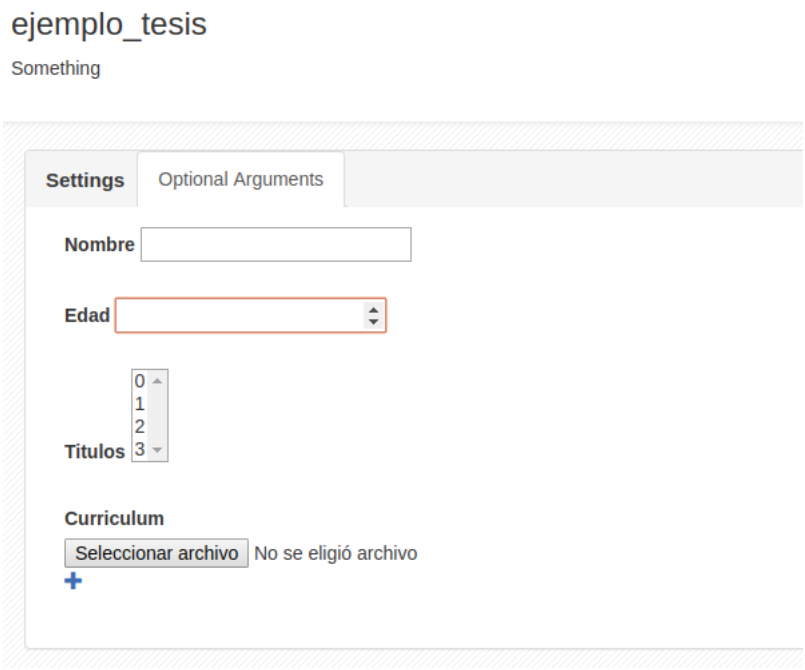
4.4 Ejemplo de uso

Supongamos el siguiente script de Python, el cuál recibe como argumentos las variables *nombre* (tipo string), *talle-preferidos* (lista de enteros), *edad* (entero) y *curriculum* (archivo), y veremos cómo se ve en Wooey dicho script transformado.

```
1 import argparse
2 import sys
3
4 parser = argparse.ArgumentParser(description="Something")
5 parser.add_argument('--nombre',type=str)
```

```
6  parser.add_argument('--edad',type=int)
7  parser.add_argument('--talle-preferidos',type=int, choices=[0,1,2,3],
       nargs=2)
8  parser.add_argument('--curriculum',type=argparse.FileType('r'),nargs='*
       ')
9
10 if __name__ = '__main__':
11     args = parser.parse_args()
12     sys.stdout.write('{}'.format(args))
```

Figura 4.4: Ejemplo del script ejemplo_tesis.py pasado a Wooey



Como vemos en la figura 4.3, cada posible argumento se convirtió en un objeto Input de HTML: La variable *nombre* de tipo string, se convirtió en un Input de tipo texto. La variable *edad*, en un input de tipo número, *Títulos* se convirtió en el tag select y *curriculum* en un file upload.

Entonces veamos como es la configuración de argumentos del script anterior vía línea de comando y Wooey. Por ejemplo, queremos configurar los siguientes parámetros en la ejecución del script:

- Nombre: Jeremías
- Edad: 27

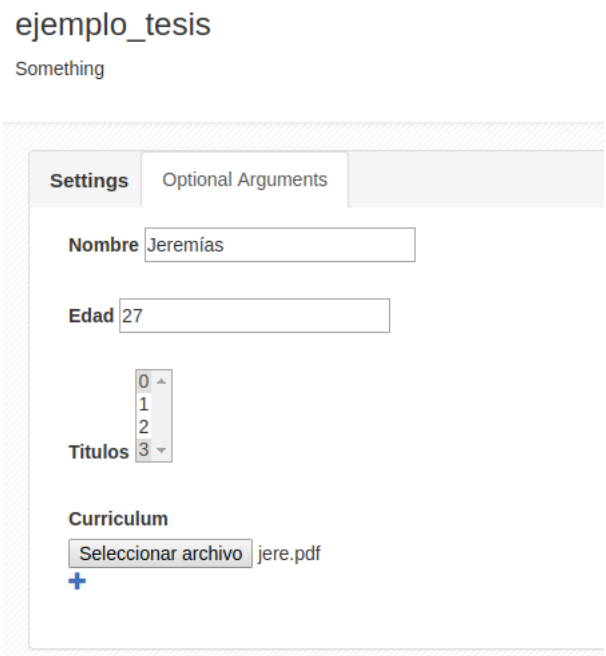
- talle-preferidos: 0 y 3
- curriculum: cv.pdf

Para ejecutarlo en línea de comando, deberíamos hacer:

Figura 4.5: Ejemplo de pasajes de parámetros del ejemplo en línea de comando

```
$ python3 ejemplo_tesis.py --nombre Jeremias --edad 27 --talle-preferidos 0,3 --curriculum jere.pdf
```

Figura 4.6: Ejemplo de pasaje de parámetros del ejemplo en Wooey



Esta transformación de la línea de comando a interfaz web, nos facilita muchísimo la configuración de los distintos argumentos gracias a lo intuitivo que es Wooey.

CAPÍTULO 5

CONCLUSIONES Y TRABAJO FUTURO

5.1 Conclusiones generales

Como concursantes frecuentes de competencias de seguridad informática, nos encontramos con el problema del poco tiempo que disponemos para resolver la totalidad de los desafíos que nos proponen. Por lo que surgió la necesidad de aprovechar al máximo cada minuto disponible. Gracias al uso de las herramientas descriptas en esta tesina, hemos podido lograr una significativa optimización del tiempo, el cual era nuestro objetivo principal al desarrollarlas.

RSolver además facilita el aprendizaje de ataques al protocolo RSA ya que tras la ejecución exitosa de un ataque proporciona una descripción del mismo para no solo recuperar la flag y resolver el problema sino también hacer hincapié en el aspecto formativo del mismo.

Hemos percibido también que los CTFs se han vuelto cada vez más competitivos, por lo que la necesidad de utilizar herramientas automatizadas se vuelve obligatoria si se quiere lograr un buen resultado.

Por otra parte, como formadores de equipo, estas herramientas favorecen la integración de nuevos participantes, ya que puede ser frustrante para un jugador novato encarar un problema de los que no tiene experiencia previa y no poder resolverlo. Con RSolver puede realizar fácilmente desafíos de criptografía RSA y luego estudiar sobre el método utilizado provisto por la herramienta.

Con SYPER CTF Tools se facilita una batería de tools que son utilizadas

frecuentemente al competir en CTFs, listas para ser ejecutadas de manera intuitiva, con una interfaz web amigable y en español. De no poseer esta herramienta, el participante debería instalar por cuenta propia uno por uno estos programas, y utilizarlos desde la terminal, con muchas opciones complejas y documentación en lenguaje extranjero.

5.2 Trabajo futuro

En cuanto al desarrollo de RSolver, podrían agregarse nuevos ataques a RSA, aún no implementados o que vayan surgiendo.

Por otro lado, podríamos migrar ataques implementados en otros lenguajes a Python, como el ataque de Boneh-Durfee, el cual necesita una dependencia (SageMath) que ocupa alrededor de 4Gb de espacio. Si logramos la migración ahorraríamos ese espacio.

También sería conveniente estudiar la posibilidad de ampliar la capacidad de problemas abordables incluyendo por ejemplo los referido a AES, XOR y GPG que suelen presentarse en los distintos CTFs.

Otra mejora posible puede ser la paralelización de los ataques para hacerlos más eficientes.

Para SYPER CTF Tools, una mejora podría ser agregar funcionalidad para poder categorizar mediante tags los resultados de las ejecuciones de las herramientas y poder filtrarlas luego por el patrón que más nos convenga. Por ejemplo, si resolver un ejercicio nos llevó cinco ejecuciones de una tool, con diferente configuración de parámetros en cada una de ellas, sería útil poder destacar la ejecución que nos dió la solución, para poder encontrarla rápidamente en caso de presentarse un problema parecido.

Incorporar nuevas herramientas al listado actual, sería bueno tener la política de incorporar las que se vayan requiriendo en los distintos CTFs de los que participamos para resolver algún reto en particular.

Si bien, al estar desarrolladas en Python, y este ser uno de los lenguajes de programación más difundidos en el mundo, realizar aportes a nuestros desarrollos resultaría sencillo para cualquier programador. La mejora de la documentación es uno de los pendientes a realizar para favorecer aún más el desarrollo por parte de terceros.

BIBLIOGRAFÍA

[1] CTFTime, "Ctfs durante 2018," 2018.

[2] Cybercamp, "Cybercamp web," 2019.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.

[4] W. L. Hosch, "Fermat's theorem." `https://www.britannica.com/science/Fermats-theorem`, 2009.

[5] J. M. Pollard, "A monte carlo method for factorization," *BIT Numerical Mathematics*, vol. 15, pp. 331–334, Sep 1975.

[6] H. W. Lenstra, "Factoring integers with elliptic curves.," *Annals Math.*, vol. 126, pp. 649–673, 1987.

[7] M. E. Briggs, "An introduction to the general number field sieve," 1998.

[8] J. Håstad, "On using rsa with low exponent in a public key network," in *Advances in Cryptology*, CRYPTO '85, (Berlin, Heidelberg), pp. 403–408, Springer-Verlag, 1986.

[9] D. Bleichenbacher, "On the security of the kmov public key cryptosystem," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 235–248, Springer, 1997.

[10] D. Coppersmith, "Finding a small root of a univariate modular equation," in *Advances in Cryptology — EUROCRYPT '96* (U. Maurer, ed.), (Berlin, Heidelberg), pp. 155–165, Springer Berlin Heidelberg, 1996.

[11] G. J. Simmons and M. J. Norris, "Preliminary comments on the m.i.t. public-key cryptosystem," *Cryptologia*, vol. 1, no. 4, pp. 406–414, 1977.

[12] M. K. Franklin and M. K. Reiter, "A linear protocol failure for rsa with exponent three," 1995.

[13] D. Coppersmith, M. K. Franklin, J. Patarin, and M. K. Reiter, "Low-exponent rsa with related messages," in *EUROCRYPT*, 1996.

[14] M. J. Wiener, "Cryptanalysis of short rsa secret exponents," *IEEE Transactions on Information Theory*, vol. 36, pp. 553–558, May 1990.

[15] D. Boneh and G. Durfee, "New results on the cryptanalysis of low exponent rsa," *IEEE Transactions on Information Theory*, 2000.

[16] A. K. Lenstra, H. W. Lenstra, and L. Lovasz, "Factoring polynomials with rational coefficients," *MATH. ANN*, vol. 261, pp. 515–534, 1982.

[17] D. Boneh, G. Durfee, and Y. Frankel, "An attack on rsa given a small fraction of the private key bits," in *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '98, (Berlin, Heidelberg), pp. 25–34, Springer-Verlag, 1998.

[18] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, (London, UK, UK), pp. 104–113, Springer-Verlag, 1996.

[19] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), (Berlin, Heidelberg), pp. 388–397, Springer Berlin Heidelberg, 1999.

[20] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology — EUROCRYPT '97* (W. Fumy, ed.), (Berlin, Heidelberg), pp. 37–51, Springer Berlin Heidelberg, 1997.

[21] M. Chapman, G. I. Davida, and M. Rennhard, "A practical and effective approach to large-scale automated linguistic steganography," in *Information Security* (G. I. Davida and Y. Frankel, eds.), (Berlin, Heidelberg), pp. 156–165, Springer Berlin Heidelberg, 2001.

[22] OWASP, *OWASP Top 10 - 2017, Los diez riesgos más críticos en Aplicaciones Web*. The OWASP Foundation, 2017.

[23] Symantec, "Norton web." https://us.norton.com/internetsecurity-wifi-what-is-a-man-in-the-middle-attack.html, 2019.

[24] D. Boneh, "Twenty years of attacks on the RSA cryptosystem," *Notices of the American Mathematical Society*, vol. 46, no. 2, 1999.

[25] M. J. Hinek, *Cryptanalysis of RSA and Its Variants*. Chapman & Hall/CRC, 1st ed., 2009.

[26] S. Y. Yan, *Cryptanalytic Attacks on RSA*. Berlin, Heidelberg: Springer-Verlag, 2007.

[27] Ganapati, "Rsa attack tool (mainly for ctf) - retreive private key from weak public key and/or uncipher data." https://github.com/Ganapati/RsaCtfTool, 2019.

[28] nccgroup, "An automated, modular cryptanalysis tool; i.e., a weapon of math destruction." https://github.com/nccgroup/featherduster, 2019.

[29] rk700, "An all-in-one tool including many common attacks against rsa problems in ctf.." https://github.com/rk700/attackrsa, 2019.

[30] ius, "rsatool can be used to calculate rsa and rsa-crt parameters." https://github.com/ius/rsatool, 2019.

[31] ValarDragon, "Tools for solving rsa and other crypto problems in ctfs.." https://github.com/ValarDragon/CTF-Crypto, 2019.

[32] X-Vector, "Cryptography tool for rsa attacks." https://github.com/X-Vector/X-RSA, 2019.

[33] zweisamkeit, "Rshack - tool for rsa ctf's challenges." https://github.com/zweisamkeit/RSHack, 2019.

[34] E. Lanfranco, "Cátedra desarrollo seguro de aplicaciones - unlp." https://www.info.unlp.edu.ar/wp-content/uploads/2018/03/Desarrollo-seguro-de-aplicaciones.pdf, 2019.

[35] E. L. Nicolás Macia, "Cátedra introducción a la ciberseguridad - unlp." https://www.info.unlp.edu.ar/wp-content/uploads/2018/05/Introduccion-a-la-ciberseguridad-2018.pdf, 2019.

[36] factordb, "Factor db web." http://www.factordb.com, 2019.

[37] wooey, "Wooey.." https://github.com/wooey/Wooey, 2019.

[38] django, "Django web." https://www.djangoproject.com/, 2019.

[39] StefanoDeVuono, "Steghide is a steganography program that is able to hide data in various kinds of image- and audio-files.." https://github.com/StefanoDeVuono/steghide, 2013.

[40] zed 0xff, "zsteg: detect stegano-hidden data in png and bmp." https://github.com/zed-0xff/zsteg, 2014.

[41] bannsec, "Yet another stego tool." https://github.com/bannsec/stegoVeritas, 2019.

[42] N. Provos, "Stegdetect is an automated tool for detecting steganographic content in images." https://linux.die.net/man/1/stegdetect, 2012.

[43] N. Provos, "Stegbreak launches brute-force dictionary attacks on jpg image." https://linux.die.net/man/1/stegbreak, 2012.

[44] h3xx, "Hide and seek for linux - these two programs let you hide a file in a jpeg file and then let you recover it later." https://github.com/h3xx/jphs, 2017.

[45] lukechampine, "jsteg is a package for hiding data inside jpeg files." https://github.com/lukechampine/jsteg, 2019.

[46] syvaidya, "Openstego is a steganography application." https://github.com/syvaidya/openstego, 2019.

[47] N. Provos, "Outguess is a universal steganographic tool that allows the insertion of hidden information into the redundant bits of data sources." https://github.com/resurrecting-open-source-projects/outguess, 2001.

[48] solusipse, "Images to audio files with corresponding spectrograms encoder." https://github.com/solusipse/spectrology, 2017.

[49] cedricbonhomme, "Stegano is a pure python steganography module. different methods of steganography and steganalysis are provided." https://github.com/cedricbonhomme/Stegano, 2019.

[50] livz, "Platform independent python tool to implement lsb image steganography and a basic detection technique." https://github.com/livz/cloacked-pixel, 2017.

[51] A. Zaenuri, "mp3stegz is an application that apply steganographic algorithm in mp3 file.." https://sourceforge.net/p/mp3stegz/wiki/Home/, 2008.

[52] M. Kwan, "Gif colourmap steganography." http://www.darkside.com.au/gifshuffle/, 2003.

[53] Paradoxis, "Steganography brute-force utility to uncover hidden data inside files." https://github.com/Paradoxis/StegCracker, 2008.

[54] nmap, "Nmap - the network mapper." https://github.com/nmap/nmap, 2019.

[55] aboul3la, "Sublist3r - fast subdomains enumeration tool for penetration testers." https://github.com/aboul3la/Sublist3r, 2019.

[56] sullo, "Nikto web server scanner." https://github.com/sullo/nikto, 2019.

[57] wpscanteam, "Wpscan is a free, for non-commercial use, black box wordpress vulnerability scanner written for security professionals and blog maintainers to test the security of their wordpress websites." https://github.com/wpscanteam/wpscan, 2019.

[58] D. Team, "Dirb is a web content scanner. it looks for existing web objects. it basically works by launching a dictionary based attack against a web server and analizing the response." https://sourceforge.net/projects/dirb/files/, 2019.

[59] OJ, "Directory/file, dns and vhost busting tool written in go." https://sourceforge.net/projects/dirb/files/, 2019.

[60] sqlmapproject, "Automatic sql injection and database takeover tool." https://github.com/sqlmapproject/sqlmap, 2019.

[61] kurobeats, "fimap is a little python tool which can find, prepare, audit, exploit and even google automatically for local and remote file inclusion bugs in webapps." https://github.com/kurobeats/fimap, 2019.

[62] xmendez, "wfuzz - web application fuzzer." https://github.com/xmendez/wfuzz, 2019.

[63] devloop, "Wapiti - a web-application vulnerability scanner." https://sourceforge.net/projects/wapiti/, 2019.

[64] commixproject, "Automated all-in-one os command injection and exploitation tool.." https://github.com/commixproject/commix, 2019.

[65] vanhauser thc, "Hydra password cracker.." https://github.com/vanhauser-thc/thc-hydra, 2019.