Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Normas para las prácticas

Universidad de Jaén

1. Estructura de las prácticas

Las prácticas de la asignatura constan de una serie de sesiones, que se desarrollarán en el laboratorio correspondiente, más un trabajo de temática relacionada con la asignatura.

Las sesiones de prácticas constan de tres partes:

- **Introducción:** Justificación y contextualización de la práctica.
- **Actividades:** Actividades a desarrollar en el laboratorio, bajo la supervisión del profesor.
- **Cuestionario:** Actividades a desarrollar de manera autónoma, que deberán ser documentadas y entregadas para su evaluación.

El trabajo:

- Es *obligatorio* para superar las prácticas.
- Se realizará *por parejas*. Con carácter excepcional, y por causas debidamente justificadas, podrá ser realizado individualmente o en grupos de tres personas.
- Será de cualquier temática relacionada con la asignatura, previamente acordada con el profesor.
- Podrá ser tanto teórico como práctico.
- Solo se entregará una copia a través de *Ilias*, por parte de cualquiera de los autores.
- Si los miembros de la pareja que realiza el trabajo pertenecen a grupos de prácticas con diferentes profesores, el trabajo será entregado únicamente a uno de los profesores, previa comunicación a todos los implicados.
- El formato y normas de entrega será el mismo que el de los informes de las sesiones de prácticas.

2. Normas de Entrega

- Las sesiones de prácticas tienen carácter individual.

- El informe correspondiente a cada práctica debe ser entregado en un único archivo, a través de la plataforma *Ilias*. Si el mismo contiene más de un archivo o documento, deben ser todos comprimidos en un único archivo `zip`. No se admitirán otros tipos de archivos comprimidos (`rar, 7z, arj, gz, bz2,` etc.)
- El nombre del archivo debe inlcuir el número de práctica e identificar al autor de la misma (no valen nombres genéricos como `Practica.zip`). Se recomienda prescindir de tildes, eñes y otros caracteres *especiales* en el nombre del archivo.
- No es necesario incluir una copia del cuestionario de la práctica dentro de la documentación a entregar.
- Los documentos deben entregarse preferiblemente en formato `pdf`. No se admitirán documentos en formatos propietarios `(doc, docx)`.

2.1. Normas de asistencia a las prácticas

- La asistencia a las prácticas es *obligatoria*.
- Solo se aceptarán ausencias a las sesiones de prácticas por causas debidamente justificadas por escrito.
- Se permite un máximo de tres ausencias a las sesiones de prácticas.
- Una vez finalizadas las sesiones de prácticas, éstas pasarán a tener carácter de tutoría para la realización del trabajo, y dejarán de ser de asistencia obligatoria, si bien la asistencia a las mismas será tenida en cuenta en el apartado de *asistencia y participación* de la asignatura.

2.2. Evaluación de las prácticas

- Aunque de forma efectiva se realicen las prácticas en colaboración con otras personas, las sesiones de prácticas son de carácter individual. No se aceptarán informes de prácticas copiados o duplicados.
- Las sesiones de prácticas obtendrán una calificación de *apto* o *no apto*. Para aprobar las prácticas hay que superar todas las sesiones.
- Para superar una sesión, se ha de entregar el cuestionario debidamente cumplimentado, dentro del plazo establecido. El profesor revisará la documentación e informará sobre si la sesión ha sido superada o no.

- Al final del cuatrimestre se dará la oportunidad de entregar los cuestionarios de las sesiones pendientes. Los cuestionarios que se entreguen en estas circunstancias tendrán una penalización en su calificación.
- Las prácticas se evaluarán siempre que se entreguen, al menos, tres cuestionarios.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 1

Universidad de Jaén

1. Introducción

Todo administrador de un sistema basado en Tecnologías de la Información tiene la obligación de conocer sus posibles vulnerabilidades, así como de poner los medios adecuados para reducir los riesgos asociados a las mismas. Resulta pues imprescindible conocer los lugares donde informarse, hacerlo con frecuencia, y saber manejar correctamente la información recabada.

En ocasiones no basta con tener información externa para saber a ciencia cierta si nuestro sistema presenta determinada vulnerabilidad, por lo que tendremos que hacer *auditorías* para comprobar la presencia o no de ella. Esta labor, consistente básicamente en intentar comprometer el sistema mediante el *exploit* correspondiente, puede hacerse de forma manual, o de forma automática. La herramienta más conocida para llevar a cabo esta tarea es *Metasploit*.

En esta práctica aprenderemos el concepto de *bugtraq*, revisaremos alguno de los más conocidos, y analizaremos la información asociada a alguna vulnerabilidad concreta. También veremos que existe todo un mercado negro de vulnerabilidades aún no publicadas, conocidas en el argot como *vulnerabilidades 0day*, dedicado a la compraventa de dicho material, con objeto de llevar a cabo acciones delictivas. Finalmente, haremos una breve introducción al *framework* de auditoría *Metasploit*.

1.1. Material

No se requiere material adicional para esta práctica.

2. Actividades

1. Las siguientes páginas *web* proporcionan información útil sobre vulnerabilidades. Un administrador debería revisar regularmente páginas de este tipo y estar suscrito a sus boletines:

   - Hispasec. Revisar las secciones *una al día*, y el servicio SANA (de pago).
   - Security Focus. Entrar en cualquiera de las vulnerabilidades y revisar las distintas pestañas con información que se presenta.
   - Base de Datos Nacional de Vulnerabilidades del NIST. Pueden hacerse consultas sobre CVE (vulnerabilidades), CCE (*common configuration enumeration*, dedicada a configuraciones inseguras), etc.

- Bases de datos CVE y CCE.
- Recopilación de listas de correo sobre seguridad. Revisar las listas *Bugtraq* y *Full Disclosure*.

2. Buscar (empleando *Google*, por ejemplo) otras páginas dedicadas a la publicación de vulnerabilidades.
3. Leer los artículos Comprando Vulnerabilidades *0day* y *The Modern History of Cyber Warfare 4: 0day Black Market and State Sponsored Attacks*, dedicados a la compraventa de vulnerabilidades en el *software*.
4. Revisar la página web de Metasploit, y buscar en la red recursos sobre dicho *software*.

3. Cuestionario

Elaborar un informe en formato `pdf`, que incluya los siguientes puntos:

1. Localice una vulnerabilidad en la página de Security Focus, preferiblemente que tenga *exploit*.
2. Haga una descripción de la misma, indicando su(s) código(s) CVE asociado(s), y valore su posible nivel de gravedad.
3. Incluya en su informe un enlace al código fuente del *exploit* asociado a la vulnerabilidad.
4. Recopile información sobre el proyecto *Metasploit* y responda a las siguientes cuestiones:
   - Explique qué es un *payload*, y qué propósito tiene.
   - ¿Para qué sirve un *payload* dinámico?
   - ¿Qué es *meterpreter?*

Todos los puntos del informe deben ser cumplimentados en castellano, con sus propias palabras, y deben ser comprensibles. No se aceptarán traducciones literales ni, por supuesto, automáticas.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 2

Universidad de Jaén

1. Introducción

Hasta la década de los 90 del siglo pasado, las técnicas de cifrado *fuertes* solo estaban al alcance de unos pocos. Fue entonces cuando Phil Zimmermann desarrolló PGP *(Pretty Good Privacy)*. Este programa permite cifrar nuestra información mediante técnicas criptográficas tanto simétricas como asimétricas, además de crear firmas digitales. También incorpora un sencillo sistema de gestión de claves, que se organizan en *anillos*, uno para las claves públicas y otro para las privadas.

Hemos de hacer hincapié en que PGP no es un algoritmo de cifrado, sino un protocolo, por lo que no está sujeto a ningún algoritmo concreto. De hecho, existen diferentes implementaciones de PGP, que no necesariamente incorporan los mismos algoritmos.

En la actualidad, PGP se ha convertido en un estándar de cifrado, cuya implementación más usada es GnuPG. Este *software* suele venir instalado por defecto en la mayoría de las distribuciones del Sistema Operativo Linux. En esta sesión práctica vamos a aprender su funcionamiento básico.

1.1. Material

Para la realización de esta práctica se requiere únicamente un computador que tenga instalada alguna versión de *GnuPG*. Cualquier ordenador con *Linux* suele llevarla instalada por defecto.

2. Actividades

1. *Verificar la instalación de GnuPG*. Escribiremos en la consola

   ```
   gpg --version
   ```

   El programa mostrará un listado de los algoritmos soportados: asimétricos, simétricos, funciones resumen y de compresión.

2. *Crear nuestro propio par de claves públicas y privadas*. Para ello abrimos la consola y ejecutamos la siguiente orden:

   ```
   gpg --gen-key
   ```

   GnuPG nos preguntará sobre el tipo de clave que queramos crear, su longitud en bits, y el período de validez que queramos darle. Escogemos la opción 1

(RSA y RSA), y el número de bits deseado, preferiblemente 4096. En cuanto al período de validez, indicaremos un valor en función del uso que queramos darle a nuestra nueva clave.

Seguidamente, habrá que construir el identificador de usuario para la clave. Para ello se nos pedirá el nombre del dueño de la clave, un comentario opcional, y una dirección de e-mail válida. Es **muy importante** proporcionar datos correctos en esta fase, ya que serán utilizados posteriormente para poder realizar el cuestionario.

Finalmente, se nos pedirá una contraseña para desbloquear la clave generada en el futuro. En este punto, el sistema estará preparado para generar nuestra clave de forma aleatoria, y para ello requerirá que llevemos a cabo algún tipo de actividad en el ordenador mientras se hacen los cálculos necesarios.

3. *Verificar que ya tenemos nuestra clave*. Para ello ejecutamos la orden

   ```
   gpg --list-keys
   ```

   el programa mostrará la clave generada, junto con cualquier otra clave pública que haya instalada en el anillo de claves. Si escribimos

   ```
   gpg --list-secret-keys
   ```

   aparecerá información relativa a las claves privadas que tengamos. Cada clave lleva asociado un código hexadecimal de 8 dígitos, que sirve de identificador. Cuando hagamos búsquedas, o queramos referirnos a una clave determinada, podemos usar tanto este valor, como el identificador de usuario (o una subcadena del mismo).

   Como medida de seguridad adicional, cada clave dispone de una *huella dactilar*, que permite identificarla de manera totalmente unívoca, y que se suele emplear para validar la autenticidad de una clave descargada de internet o recibida por e-mail. Para ver la huella dactilar de nuestra clave escribiremos lo siguiente:

   ```
   gpg --fingerprint <id>
   ```

   donde `<id>` es alguna cadena que identifique la clave, como se ha mencionado más arriba.

4. *Exportar nuestra clave pública en formato ASCII*. La siguiente orden genera un bloque de texto que podemos volcar a un archivo, o enviar vía e-mail, para proporcionar a otras personas nuestra clave pública.

   ```
   gpg -a --export <id>
   ```

El destinatario utilizará luego la siguiente orden para importar nuestra clave pública:

```
gpg --import <nombre_de_archivo>
```

5. *Subir nuestra clave a un servidor de claves, y descargar las de otras personas*. A veces es más sencillo emplear un servidor externo para poder intercambiar claves públicas. Para subir nuestra clave pública recién generada al servidor, escribiremos en la consola

```
gpg --keyserver hkp://keyserver.ubuntu.com:80
        --send-keys <id>
```

y el programa subirá automáticamente al servidor la clave cuyo identificador hexadecimal sea `<id>`.

El siguiente paso es descargar las claves de otras personas. Si conocemos el identificador hexadecimal de la clave que buscamos, basta con ejecutar

```
gpg --keyserver hkp://keyserver.ubuntu.com:80
        --recv-keys <id>
```

En caso de que desconozcamos el identificador hexadecimal, podemos buscar en el servidor la clave o claves a partir del identificador de usuario:

```
gpg --keyserver hkp://keyserver.ubuntu.com:80
        --search-keys <patron>
```

El programa listará todas las claves públicas encontradas que contengan `<patron>` en el identificador de usuario, para luego importar cualquiera de ellas.

6. *Cifrado y descifrado de mensajes*. GnuPG soporta diferentes tipos de cifrado, adecuados tanto para cifrar archivos binarios como texto ASCII estándar. En el primer caso, emplearemos los siguientes comandos:

   - Cifrar un archivo binario:

     ```
     gpg -e -r <destinatario> <fichero>
     ```

   - Descifrar el archivo binario:

     ```
     gpg -d <fichero_cifrado>
     ```

   Para cifrar mensajes de texto, tenemos dos opciones. Una de ellas consiste en cambiar `-e` por `-ea` en el caso anterior y cifrar normalmente, con lo que GnuPG creará un nuevo fichero con la extensión `.asc`, en formato ASCII e imprimible,

que podemos copiar directamente en un e-mail. Si queremos obtener el fichero cifrado directamente por la salida estándar, podemos usar el siguiente comando:

```
cat <fichero>| gpg -ea -r <destinatario>
```

En todos los comandos anteriores podemos incluir más de un destinatario, usando varias veces la opción `-r`.

Como puede observarse, GnuPG únicamente pide la contraseña para descifrar, que es la única de las operaciones de este apartado que requiere emplear la clave privada.

7. *Generación de firmas digitales*. Otra de las aplicaciones de GnuPG consiste en firmar digitalmente tanto ficheros como mensajes de texto. Tenemos las siguientes opciones:

   - Crear un nuevo fichero que contiene el original más la firma digital:
     ```
     gpg -s <fichero>
     ```
   - Crear una firma digital separada del fichero original, que queda intacto:
     ```
     gpg -b <fichero>
     ```
   - Crear una firma digital en modo texto:
     ```
     gpg --clearsign <fichero>
     ```
   - Crear una firma digital en modo texto, por la salida estándar:
     ```
     cat <fichero>| gpg --clearsign
     ```
   - Verificar cualquier tipo de firma:
     ```
     gpg --verify <fichero>
     ```

8. *Firmar la clave pública de otra persona y actualizarla en el servidor.* Una vez descargada una clave de otra persona en nuestro anillo, podemos firmarla digitalmente. Eso significa que tenemos total certeza de que la clave es auténtica (nada nos hubiera impedido crear nuestra clave en la actividad 2 a nombre de cualquier persona), y que nosotros *certificamos* su autenticidad. Emplearemos el siguiente comando:

   ```
   gpg --sign-key <id>
   ```

   Una vez firmada la clave, podemos actualizarla en el servidor, para que otros usuarios puedan *ver* nuestra firma, y aumentar su nivel de confianza sobre la autenticidad de la clave. El siguiente comando actualiza las claves públicas de nuestro anillo con todas las nuevas firmas que haya en el servidor, y sube las firmas que nosotros hayamos añadido:

```
gpg --keyserver hkp://keyserver.ubuntu.com:80
        --refresh-keys
```

Se recomienda que cada uno de los alumnos firme las claves públicas de varios de sus compañeros, y que consiga también unas cuantas firmas para su propia clave.

Si se pretende utilizar en el futuro la clave que se ha creado en esta sesión de prácticas, el profesor le agradecería que firmara su clave pública, cuyo identificador es `6627F766`, y cuya huella dactilar es:

```
FEE8 C959 6399 2668 9E16 38E7 9740 4C7A 6627 F766
```

9. *Exportar la clave privada generada para poder usarla en otras computadoras*. Como es lógico, si queremos emplear en el futuro la clave que acabamos de crear, tendremos que exportarla en un archivo, lo que conseguiremos con los siguientes comandos:

   ```
   gpg -a -o <fichero_pub> --export <id>
   gpg -a -o <fichero_priv> --export-secret-keys <id>
   ```

   Con el comando `gpg <nombre_fichero>` comprobaremos que las claves han sido exportadas correctamente. Finalmente, escribiendo

   ```
   gpg --import <fichero>
   ```

   podremos importar nuestras claves en otra computadora.

10. *Borrar la clave generada del ordenador del laboratorio (opcional)*.

    ```
    gpg --delete-secret-keys <id>
    gpg --delete-keys <id>
    ```

3. Cuestionario

1. Genere un par de claves pública/privada, con un plazo máximo de validez de 10 meses.
2. Exporte la clave pública en un fichero de texto en formato ASCII, con el nombre `XXXXXXXX_pub.asc`, donde `XXXXXXXX` es el identificador de la clave generada.
3. Escriba un fichero de texto ASCII (`mensaje.txt`), que contenga la siguiente información:
   - El nombre, DNI del alumno/a, y los grupos de teoría y prácticas a los que pertenece.
   - El curso académico.
   - La huella dactilar de la clave pública generada.
   - Todo lo anterior debe ir firmado en modo texto (`--clearsign`) con la clave privada generada.
4. **Opcional:** Suba su clave pública a un servidor de claves, y firme digitalmente los correos que envíe a cualquiera de los profesores de la asignatura.

Envíe por *Ilias* los dos archivos resultantes de estas actividades al profesor, siguiendo las normas generales.

Notas:

- Asegúrese de que incluye en el informe de prácticas su clave pública y no la privada.
- La clave pública exportada debe estar en formato ASCII, lo cual implica que debe poder visualizarse en cualquier editor de texto.
- La firma en modo texto debe abarcar toda la información contenida en el fichero `mensaje.txt`.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 3

Universidad de Jaén

1. Introducción

La Criptografía es una de las herramientas esenciales para garantizar determinadas propiedades de los sistemas TIC, como la confidencialidad, la integridad y el no repudio.

Cryptool es una aplicación de aprendizaje electrónico gratuita para Windows. Puede utilizarse para aplicar y analizar algoritmos criptográficos. La versión actual ofrece, entre otras cosas, lo siguiente:

- Numerosos algoritmos criptográficos, clásicos y modernos (cifrado y descifrado, generación de clave, contraseñas seguras, autentificación, protocolos seguros, ...)
- Visualización de varios métodos (p.ej. César, Enigma, RSA, Diffie-Hellman, firmas digitales, AES)
- Criptoanálisis de ciertos algoritmos (p.ej. Vigenère, RSA, AES)
- Métodos de medida criptoanalítica (p.ej. entropía, n-grams, autocorrelación)
- Métodos auxiliares (p.ej. tests de primalidad, factorización, codificación en base64)
- Tutorial sobre teoría de números.
- Ayuda detallada on-line.
- Script con más información sobre criptografía.

1.1. Material

Para la realización de esta práctica emplearemos una máquina virtual con todo el software y los ficheros auxiliares preinstalados.

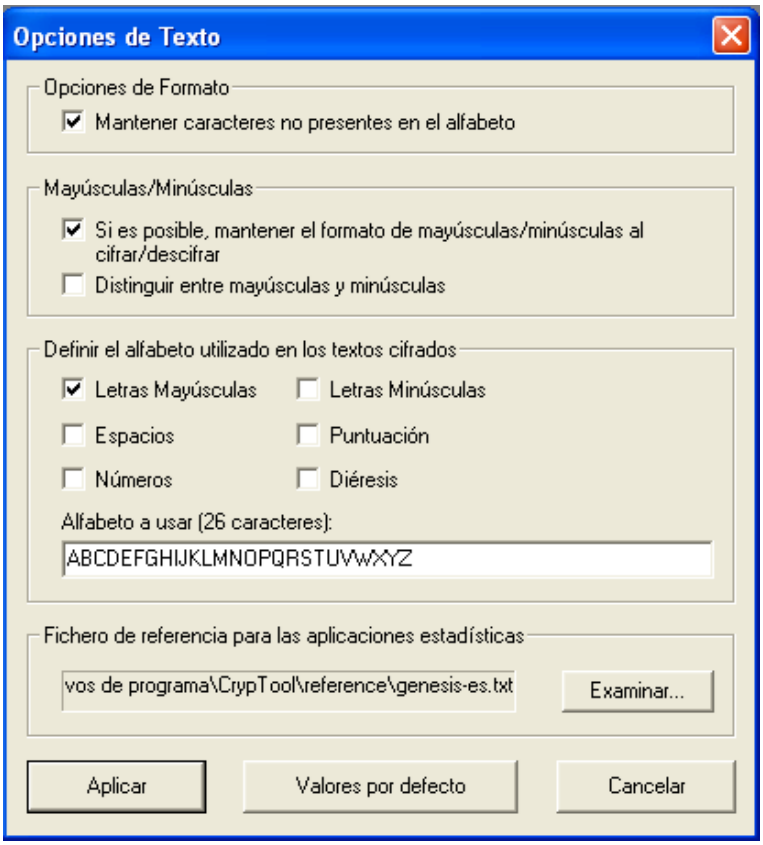


Figura 1: Opciones de texto para las operaciones de cifrado clásico en *Cryptool*.

2. Actividades

1. *Configuración inicial*. Después de arrancar *Cryptool* y cerrar la ventana de bienvenida, seleccionar *Opciones* → *Texto...* y ajustar los parámetros a los valores que se muestran en la figura 1.

2. *Cifrado monoalfabético (César)*. Ver la demostración interactiva *(Procedimientos Indiv.*→ *Visualización de Algoritmos* → *César...)*. Pulsar *play* (▷) en el recuadro *Navigation* para ir viendo la demostración.

3. *Cifrado y Criptoanálisis monoalfabético*.

   - En primer lugar, cargaremos el archivo `Neuromante.txt` en el entorno de trabajo.
   - Calcularemos su histograma *(Análisis* → *Herramientas para el Análisis* → *Histograma)*. Reducimos el tamaño de la ventana y la dejamos en el fondo.
   - Cifraremos el fichero *(Cifrar/Descifrar* → *Simétrico (clásico)* → *César/Rot-13...)*.
   - Calculamos el histograma del fichero cifrado, y lo comparamos con el histograma del fichero sin cifrar.
   - Calcular el histograma del fichero `genesis-es.txt` (en la carpeta `reference` dentro del directorio de *Cryptool)*. Comparar con los anteriores.

4. *Cifrado polialfabético (Vigenère)*.

   - Cargar el fichero `Vigenere.txt`.
   - Calcular su gráfica de autocorrelación *(Análisis* → *Herramientas para el Análisis* → *Autocorrelación)*.
   - Como puede observarse, existen picos en dicha gráfica a distancias regulares (9). Eso nos da una idea de la longitud de la contraseña empleada.
   - Emplear el método automático de descifrado *(Análisis* → *Cifrado Simétrico (clásico)* → *Sólo Texto Cifrado* → *Vigenère)*. *Cryptool* estimará entonces la contraseña y posteriormente descifrará el mensaje, empleando análisis de frecuencias.

5. *La máquina Enigma*. *(Procedimientos Indiv.* → *Visualización de Algoritmos* → *Enigma)*.

6. *Cifrados modernos*. Ver la presentación sobre AES, y la demostración interactiva. *(Procedimientos Indiv.* → *Visualización de Algoritmos* → *AES)*. Seleccionar y visualizar las dos primeras opciones.

3. Cuestionario

1. Cree un fichero vacío *(Archivo → Nuevo)*, escriba su nombre, ajuste su tamaño y colóquelo en la esquina superior izquierda del área de trabajo.
2. Cargue `Ejercicio.txt`, realice su criptoanálisis para el método de cifrado de Vigenère y haga una captura de pantalla donde se pueda ver la gráfica de autocorrelación y la contraseña, así como la ventana con su nombre.
3. Cifre un mensaje, empleando la máquina *Enigma*, con la siguiente configuración:
   - Rotores: IV,II,I.
   - Configuración inicial: Las tres primeras letras del nombre del alumno.
   - Mensaje: PRACTICA NUMERO TRES.
   - Anote el resultado en un documento de texto, indicando claramente la configuración inicial que ha escogido y el resultado obtenido.
   - Incluya dicho documento en su informe de prácticas.
4. Anote los valores completos de la primera columna de la salida del algoritmo AES, poniendo como entrada en la primera columna los ocho dígitos de su DNI (dos en cada casilla), y colocando todos los demás valores a cero.
5. Incluya los valores obtenidos en un documento de texto, indicando los ocho dígitos de su DNI.

Notas:

- El informe de prácticas debe contener una imagen con la captura de pantalla, y dos documentos de texto, con los resultados del resto de actividades.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 4

Universidad de Jaén

1. Introducción

Para esta sesión emplearemos de nuevo la herramienta *Cryptool*, ya introducida en la sesión anterior. En este caso, nos centraremos en las secciones dedicadas a las funciones *hash* y los algoritmos asimétricos.

1.1. Material

Para la realización de esta práctica emplearemos una máquina virtual con todo el software y los ficheros auxiliares preinstalados.

2. Actividades

1. Empleando la opción del menú *Demostración Hash... (Procedimientos Indiv.* $\rightarrow$ *Hash)*, seleccione el algoritmo SHA-1 y modifique el texto que aparece en la caja correspondiente. Observe cómo el número de bits que aparecen en rojo está siempre próximo al 50 %. Esto se debe a la apariencia de *aleatoriedad* que debe tener una buena función *hash*.

2. *Algoritmo RSA*. Seleccione *Procedimientos Indiv.* $\rightarrow$ *RSA Criptosistema* $\rightarrow$ *Demostración RSA*. Una vez generados dos números primos en el rango $(2^{128}, 2^{129})$, mediante el método de Rabin-Miller, seleccione *números* en el diálogo correspondiente, y cifre un valor numérico cualquiera. Observe el resultado.

3. *Generación de una firma Digital*. Seleccione la opción del menú *Firma Digital/PKI* $\rightarrow$ *Demostración de Firma*. Siga paso a paso el proceso completo, pulsando sobre los bloques en rojo. Seleccione la función *hash* SHA-1, y genere la clave asimétrica empleando los valores por defecto. Para el certificado, ponga su nombre y apellidos, seleccione un PIN y pulse *Crear Certificado y PSE*. Pulse sobre los bloques azules para ver los resultados intermedios del proceso. Para terminar, pulse sobre *Store Signature* y *Cryptool* generará una copia del documento inicial, firmada digitalmente, que será mostrada en formato hexadecimal.

4. *Extracción y análisis de una Firma Digital*. Con el documento resultante de la actividad anterior en el foco, seleccione la opción *Firma Digital/PKI* → *Extraer Firma*. Observe los diferentes parámetros de la firma creada en la actividad anterior.

5. *Verificación de la firma*. Con el documento resultante de la actividad anterior en el foco, seleccione la opción *Firma Digital/PKI* → *Verificar Firma*. Marque en la lista el certificado que se creó para firmar el documento, y pulse *Verificar Firma*.

6. *Cifrado híbrido*. Seleccione la opción del menú *Cifrar/Descifrar* → *Híbrido* → *RSA-AES Cifrar*. Pulse sobre los diferentes bloques rojos para ir ejecutando cada paso, y sobre los bloques azules para observar los valores intermedios.

3. Cuestionario

1. Empleando la opción del menú *Demostración Hash... (Procedimientos Indiv.* $\rightarrow$ *Hash)*, seleccione el algoritmo SHA-1 y cambie el texto por su número de DNI (sin letra), sin espacios y sin retornos de carro al final. ¿Cuáles son los cinco primeros valores hexadecimales que aparecen en la caja de texto *Valor Hash del documento actual*?

2. Empleando como base los números primos $p = 12675043304375316817$ y $q = 16048240382861180269$, con un valor de $e = 65537 = (2^{16} + 1)$, cifre el valor numérico de su DNI, e indique, junto con su DNI, los ocho primeros dígitos del resultado.

Notas:

- El informe de prácticas debe un único fichero `pdf` donde se indiquen los valores obtenidos en los dos puntos del cuestionario.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 5

Universidad de Jaén

1. Introducción

Las contraseñas *(lo que sé)* son uno de los métodos, solos o en combinación con otros, más utilizados en la actualidad para garantizar la presencia de una persona frente a un sistema informático.

El ataque más sencillo a un sistema de este tipo consiste en probar muchas veces, hasta *adivinar* la contraseña. Por eso se suele limitar tanto el número de intentos permitidos, además de introducir un retardo entre intentos.

Otra vía de ataque consistiría en tener acceso al sistema objetivo, y extraer las contraseñas de los usuarios, para poder suplantarlos. Por esta razón, almacenar simplemente las contraseñas en el sistema es una estrategia errónea, ya que podrían filtrarse o extraerse si el sistema queda comprometido. En su lugar, las contraseñas se almacenan cifradas mediante una función de una sola dirección (una función *hash)*.

No obstante, un atacante podría precalcular los *hashes* de millones de contraseñas, construyendo un diccionario que podría comparar con los *hashes* reales del sistema objetivo. Para evitar esto, se añade lo que se denomina *sal*: una cadena aleatoria que se concatena con la contraseña, que obligaría a un eventual atacante a repetir todos los cálculos para cada una de las posibles contraseñas de su diccionario.

Aún en el caso de usar *hashes* y *sal*, si nuestra contraseña es lo suficientemente fácil de adivinar, bien por ser demasiado corta, bien por ser *predecible*, un ataque por la fuerza bruta podría tener éxito. Por eso es tan importante escoger buenas contraseñas.

En esta práctica explicaremos estrategias para escoger buenas contraseñas, y aprenderemos el uso básico de John the Ripper, un programa para extraer contraseñas por la fuerza bruta.

1.1. Material

Para la realización de esta práctica emplearemos una máquina virtual con todo el software y los ficheros auxiliares preinstalados.

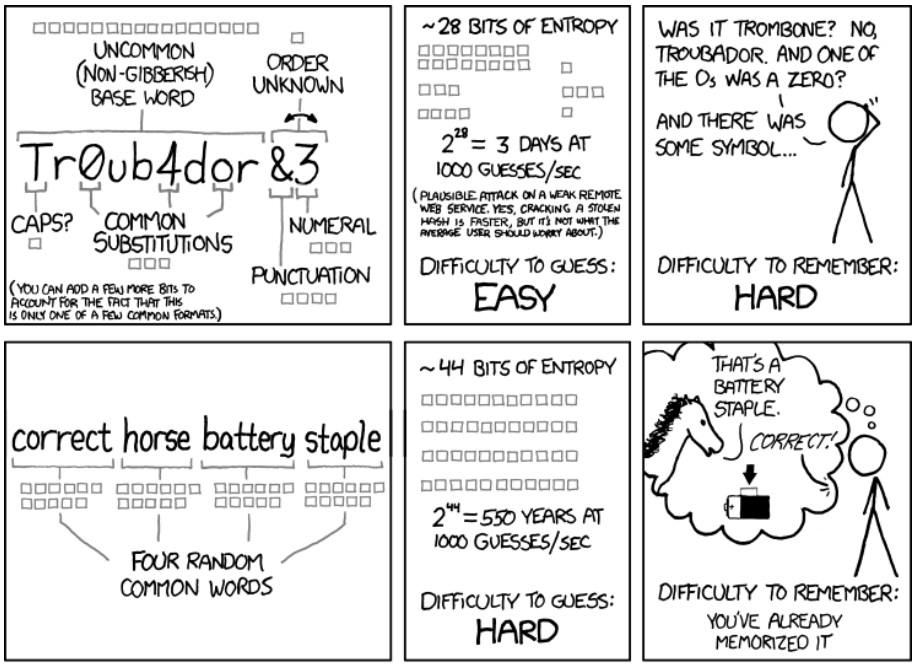
2. Actividades

1. *El método de creación de contraseñas Diceware*. Permite obtener contraseñas totalmente aleatorias, empleando un dado y una lista de palabras. Existen versiones adaptadas a diferentes idiomas, por lo que las contraseñas resultantes son fáci-

les de memorizar. Generaremos una contraseña usando este método, a modo de ejemplo.

Si no se dispone de dados, pueden usarse dados virtuales en línea, pero recuerde que no tendremos garantías de que la contraseña generada sea realmente aleatoria.

2. *Creación automática de contraseñas seguras*. Existen tanto servicios en línea, como programas para diversas plataformas:
   - Strong Password Generator.
   - Free Password Generator.
   - `pwgen`, `makepasswd` (Linux) y PWGen (Windows).
3. *Otras técnicas*. Existen diversas reglas nemotécnicas para generar contraseñas.
   - Emplear las iniciales de una frase más o menos compleja. *En un lugar de la Mancha* ⇒ EuldlM.
   - Añadir o intercalar símbolos que no sean letras ni números.
   - Emplear códigos *leet* (1337 5p34k).
   - Usar una misma contraseña, con diferentes prefijos/sufijos para diferentes servicios.



4. *Gestores de contraseñas*. Existen programas y/o servicios que almacenan nuestras contraseñas en una base de datos, que a su vez se desbloquea mediante el uso de una *contraseña maestra*. La mayoría incorpora a su vez herramientas de generación automática de contraseñas aleatorias.

   - LastPass.
   - KeePass/KeePassX.
   - Password Safe.
   - SuperGenPass es un *bookmarklet* en JavaScript que genera contraseñas a partir de una contraseña maestra y el nombre del dominio donde queramos emplearla.

5. *Descifradores de hash*. Cuando se almacenan únicamente los *hashes* de las contraseñas, sin sal, es muy fácil construir diccionarios enormes con los *hashes* de millones de palabras.

   Entre en la página web MD5 Decrypter e introduzca el siguiente valor en la caja de la izquierda:

   ```
   117a520adbd19eff51100215aa7a7fbf
   ```

6. *John the Ripper*. En la carpeta correspondiente de la máquina virtual se encuentra el ejecutable de este programa para la obtención de contraseñas por la fuerza bruta. Además del ejecutable `johh.exe` propiamente dicho, se suministra un archivo `contrasenias.txt`.

   Abra el archivo `contrasenias.txt` con un editor de texto. Cada una de sus líneas contiene el nombre de un usuario, junto con el *hash* (y la sal) de su contraseña, en un formato estándar de UNIX, separados por dos puntos (:). *John the Ripper* reconoce diversos tipos de formatos de archivos de contraseña.

   Ejecute la orden `john.exe contrasenias.txt`.

3. Cuestionario

1. Busque en Internet información sobre la entropía de una contraseña, y calcule cuántas palabras necesita una contraseña *Diceware* para tener una entropía superior a 70 bits. Explique y justifique su respuesta en el informe de prácticas, incluyendo los enlaces a las fuentes empleadas.

2. Genere una contraseña empleando el método Diceware en Español, con el número de palabras obtenido en la actividad anterior. Incluya en el informe de prácticas todos los resultados de las diferentes tiradas, así como la contraseña final.

3. Busque documentación sobre *John the Ripper* y explique qué tendría que añadir a los ficheros de configuración, y cómo tendría que ejecutarlo, para crear un método de búsqueda incremental que solo use contraseñas de una longitud máxima de 4 letras minúsculas.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 6

Universidad de Jaén

1. Introducción

Esta práctica trata sobre ingeniería inversa. En ella emplearemos OllyDbg, un programa que permite desensamblar y trazar el código de un programa ejecutable de Windows 32bits, así como modificarlo en tiempo de ejecución.

1.1. Material

Para la realización de esta práctica emplearemos una máquina virtual con todo el software y los ficheros auxiliares preinstalados.

2. Actividades

1. *Ingeniería Inversa*. Vamos a analizar el programa `vault.exe`.
   a) Ejecutar el programa. Observaremos que pide un nombre de usuario y una contraseña.
   b) Abrir el programa con OllyDbg. Pulsando el botón derecho sobre el diálogo de abajo a la izquierda *(Hex dump)*, y escogiendo la opción *Search for all referenced strings* podemos observar que hay seis cadenas a las que se hace referencia en el código del programa. Las dos primeras parecen interesantes... intente *aprovechar* dichos valores.
   c) Seleccionando la cadena *Robin Banks* y pulsando Enter, el *debugger* nos llevará a el punto del programa donde se hace referencia a esa cadena. Vemos que poco más abajo hay una llamada a la función `KERNEL32.lstrcmp`, que sirve para comparar cadenas. Poco más adelante vemos una estructura similar, esta vez relacionada con la contraseña.
   d) Cada una de las llamadas vistas en el paso anterior va seguida de un `OR EAX,EAX` y de un salto condicional `(JNE SHORT...)`. Esos saltos condicionales se llevarán a cabo cuando las cadenas introducidas por el usuario no coincidan con las que espera el programa.
   e) Haciendo doble click sobre cada uno de los `JNE` podemos sustituirlos por la orden `NOP` para inhabilitarlos. Observemos lo que ocurre en cada caso.
2. Utilice *OllyDbg* para analizar el programa `crackme.exe`. Este programa muestra una *nag-screen* al iniciarse, y posee un menú secreto, que aparece inhabilitado. Intente eliminar dichas restricciones.

3. Cuestionario

1. Modifique el programa `crackme.exe` para que no salga la ventana inicial, y el menú *Help* → *Secret* sea seleccionable. Modifique el mensaje mostrado al pulsar sobre *Secret*, cambiándolo por su número de DNI.

2. En su informe de prácticas, explique detalladamente, con capturas de pantalla donde pueda verse su nombre, todo el proceso seguido para modificar el programa del punto anterior, qué modificaciones ha hecho en el código y el efecto que tiene cada una de ellas.

Grado en Ingeniería Informática

Seguridad en Tecnologías de la Información

Curso 2017/18

Práctica Número 7

Universidad de Jaén

1. Introducción

Los Sistemas de Detección de Intrusiones (en inglés, IDS) permiten saber si se ha producido (o se está produciendo) una intrusión en un sistema informático. Para ello analizan diferentes fuentes de información (archivos de registro, configuración del sistema, tráfico de red, etc) y, mediante un mecanismo de decisión, usualmente basado en reglas, avisan si detectan alguna anomalía.

En esta práctica veremos algunas de las estrategias más comunes para detectar intrusiones, así como algunos programas que las implementan.

1.1. Material

Para la realización de esta práctica emplearemos una máquina virtual con todo el software y los ficheros auxiliares preinstalados. Para poder usarla correctamente hay que tener en cuenta lo siguiente:

- A la hora de importarlo conviene marcar la casilla de generación de una nueva MAC para la tarjeta de red.
- Tanto el nombre de usuario como la contraseña son `user`.

2. Actividades

1. *Detección basada en archivos de registro: Fail2ban*. Existen determinados patrones de ataque que quedan reflejados en los archivos de registro del sistema operativo.

   Mostramos a continuación un fragmento del fichero `/var/log/auth.log`, de un sistema Linux, que registra las entradas al sistema. Se puede observar una serie de intentos consecutivos de *login* al servicio *ssh*, con diferentes nombres de usuario, desde cuatro IPs distintas:

```
Dec 11 13:55:13 mlucena-desktop sshd[1489]: Invalid user oracle from 115.119.232.166
Dec 11 13:55:16 mlucena-desktop sshd[1491]: Invalid user test from 115.119.232.166
Dec 12 00:58:00 mlucena-desktop sshd[12621]: Invalid user admin from 49.212.84.108
Dec 12 00:58:04 mlucena-desktop sshd[12625]: Invalid user test from 49.212.84.108
Dec 12 00:58:07 mlucena-desktop sshd[12632]: Invalid user guest from 49.212.84.108
Dec 12 00:58:10 mlucena-desktop sshd[12634]: Invalid user webmaster from 49.212.84.108
Dec 12 00:58:13 mlucena-desktop sshd[12636]: Invalid user mysql from 49.212.84.108
Dec 12 00:58:16 mlucena-desktop sshd[12639]: Invalid user oracle from 49.212.84.108
Dec 12 00:58:20 mlucena-desktop sshd[12641]: Invalid user library from 49.212.84.108
```

```
Dec 12 02:29:08 mlucena-desktop sshd[13998]: Invalid user admin from 211.240.2.85
Dec 12 04:28:25 mlucena-desktop sshd[15572]: Invalid user oracle from 85.17.208.131
Dec 12 04:28:31 mlucena-desktop sshd[15585]: Invalid user oracle from 85.17.208.131
Dec 12 04:28:47 mlucena-desktop sshd[15617]: Invalid user oracle from 85.17.208.131
Dec 12 04:28:54 mlucena-desktop sshd[15629]: Invalid user nagios from 85.17.208.131
Dec 12 04:28:55 mlucena-desktop sshd[15631]: Invalid user oracle from 85.17.208.131
Dec 12 04:28:56 mlucena-desktop sshd[15633]: Invalid user postgres from 85.17.208.131
```

Fail2ban supervisa (entre otros) este archivo y, mediante el uso de expresiones regulares, detecta actividades *sospechosas*, y aplica acciones contra las IP correspondientes, como por ejemplo su inclusión en el fichero `/etc/hosts.deny`, su bloqueo en el cortafuegos, etc. Este programa posee un mecanismo configurable `(jail.conf)` donde se especifica por cuánto tiempo se aplicará la acción correspondiente sobre una IP tras detectarse un comportamiento anómalo. También hay una serie de filtros (situados en `/etc/filter.d`) que permiten detectar diferentes tipos de ataques.

Examine en la máquina virtual suministrada los ficheros de configuración de *Fail2ban:*

`sudo less /etc/fail2ban/fail2ban.conf` (contraseña: `user`)

`sudo less /etc/fail2ban/jail.conf`

`sudo less /etc/fail2ban/filter.d/sshd.conf`

`sudo less /etc/fail2ban/action.d/hostdeny.conf`

Con el comando `sudo fail2ban-client status` podremos ver qué filtros *(jails)* están activos, y configurarlos.

El comando `sudo fail2ban-client status sshd` muestra información relativa al estado del filtro SSH.

2. *Comprobación de la integridad de ficheros ejecutables*. Aunque hay muchos programas que incorporan esta funcionalidad, la misma es muy fácil de implementar en UNIX. Basta con ejecutar el siguiente código:

   ```
   find /usr/bin/ -type f -exec sha1sum {} \;
   ```

   y redireccionar la salida a un fichero (`> fichero.txt`). Posteriormente, puede ejecutarse la siguiente orden para detectar anomalías:

   ```
   sha1sum -c fichero.txt
   ```

   Ejecute la primera de las instrucciones y modifique, con la ayuda de un editor de textos, el valor SHA1 de cualquier fichero. Compruebe que la salida de la segunda instrucción devuelve una advertencia. Puede añadir un filtro (`grep`) para visualizar únicamente los ficheros alterados:

   ```
   sha1sum -c fichero.txt | grep "no coincide"
   ```

Llevando a cabo estas operaciones de manera periódica podemos detectar alteraciones en cualquier fichero (o conjunto de ellos) en el sistema que queremos vigilar.

3. *Detección basada en la integridad del sistema: Rootkit Hunter*. Este programa mantiene una base de datos de posibles indicios de la presencia de algunos de los *rootkits* más comunes en un sistema UNIX.

   Ejecute en la máquina virtual el programa *Rootkit Hunter*:

   `sudo rkhunter --check` (contraseña: `user`)

   y observe el resultado.

4. *Detección basada en el tráfico de red*. *Sguil* es una aplicación que monitoriza el tráfico de red de una computadora, detecta y avisa de situaciones anómalas. El procedimiento para ponerla en marcha en nuestra máquina virtual es el siguiente:

   a) Habilitamos el servicio de captura y análisis de paquetes. Para ello, escribimos en la consola lo siguiente:

   `sudo nsm_server_ps-restart`

   b) Arrancamos la aplicación *Sguil*, dentro de la sección *Security Onion* del menú de inicio. El usuario es `user` y la contraseña es `user1234`.

   c) En la siguiente ventana, seleccionamos `eth1` y *Start SGUIL* después.

   Como resultado obtendremos una ventana con tres paneles principales, donde se muestran los eventos en tiempo real, su nivel de peligrosidad, los detalles individuales de cada incidencia, y otra información de interés. Para simular un ataque, podemos ejecutar las siguientes órdenes en consola:

   ```
   cd /opt/samples
   tcpreplay -i eth1 -t example.com-7.pcap
   ```

   Pulsando sobre la alerta, podremos visualizar los detalles del paquete que la ha generado, la regla que se ha activado, etc. Pruebe con otros archivos `pcap` y observe el resultado.

3. Cuestionario

1. Obtenga, instale y ponga en marcha el ordenador virtual suministrado para esta sesión.

2. Ejecute la siguiente orden:

   ```
   date && echo "[NOMBRE]"| sha1sum
   ```

   sustituyendo `[NOMBRE]` por su nombre y apellidos. Realice una captura de pantalla e inclúyala en el cuestionario.

3. Revise el fichero de configuración de *Fail2ban* `jail.conf` y comente el significado de las siguientes directivas:

   ```
   bantime
   maxretry
   findtime
   port
   banaction
   ```

4. Ejecute:

   ```
   sudo rkhunter --check
   ```

   Anote todos los avisos *(warnings)* que emita el programa durante su ejecución, e indique cuál cree que puede ser el significado de cada uno de ellos.

5. Ejecute las siguientes órdenes (sustituya `[NOMBRE]` por su propio nombre y apellidos):

   ```
   echo "[NOMBRE]"
   ```

   ```
   find /usr/bin/ -type f -exec sha1sum {} \; | sha1sum
   ```

   Realice una captura de pantalla e inclúyala en el cuestionario.