Práctica 1
Vulnerabilidades de Seguridad - Primavera 2018

Presentación

Práctica 1 de vulnerabilidades de seguridad sobre detección de malware y seguridad de bajo nivel. Una vez resueltos los ejercicios, recordad hacer las siguientes comprobaciones:

	Lo he hecho?
Parte de la solución se tiene que entregar en el registro de evaluación continua en un único archivo en formato pdf. No se admite ningún otro formato (doc, docx, odt, zip, rar, tar.gz, ...).	☐
Todas las páginas tienen que estar numeradas y tienen que contener el nombre y los apellidos.	☐
La fecha máxima de entrega es el **29 de Marzo de 2019 (a las 24 horas UTC/GMT +2 hora).**	☐
Esta actividad se puede resolver, con la nota máxima, de forma relativamente breve. En ningún caso la extensión de la solución debe superar las **5 páginas** con fuente tamaño mínimo 10pt (incluido el texto en las figuras) con interlineado simple. Por favor limitad el uso de capturas de pantalla a cuando sean estrictamente necesarias y mirad que no ocupen demasiado espacio en las páginas.	☐
Razonad la respuesta a todos los ejercicios e indicad todos los pasos que habéis realizado para obtener la solución. Las respuestas sin justificación, que sean una copia de una fuente de información y/o que no contengan las referencias utilizadas, no recibirán ninguna puntuación.	☐

Ejercicio 1 (se entrega vía cuestionario) – 40% de la nota

Para el primer ejercicio estudiaremos como realiza la detección de malware un antivirus. Concretamente, trabajaremos con ClamAV, un antivirus de código abierto que realiza una detección basada en firmas. Os podéis descargar el antivirus de su web en https://www.clamav.net para usuarios de Windows, o a través del gestor de paquetes para vuestra distribución Linux.

Este ejercicio se realiza mediante un cuestionario en linea. El cuestionario está disponible en el aula en el enlace «Cuestionarios». Allí encontraréis los enlaces correspondientes al cuestionario para esta práctica. Este cuestionario consta de 2 preguntas:

Pregunta 1 –- Hacer una firma de tipo «*hash*» para un malware concreto.

Pregunta 2 –- Hacer una firma de tipo «*logical*» para un malware concreto.

En los dos casos tendréis disponibles ficheros infectados con el virus, y ficheros no infectados para poder hacer las pruebas.

Aspectos importantes:

- Tenéis que dar la solución en el propio cuestionario en linea, en un campo de texto disponible en cada pregunta. La solución se limita a introducir la firma, no tenéis que dar ninguna información adicional (justificación, explicación de los pasos, etc.)
- El formato de la firma tiene que ser el correcto tal como queda introducido en el cuestionario. Verificad que efectivamente ClamAV hace las detecciones de manera correcta, puesto que la corrección se hará del mismo modo (ejecutando ClamAV con vuestra firma sobre los ficheros proporcionados). Es decir, un pequeño error de formato puede hacer que vuestra firma no funcione (en cuyo caso se dará la respuesta como errónea).
- Las firmas que generéis no deben dar falsos positivos. Esto quiere decir que no deben dar positivo para archivos que no contengan el virus.
- En el cuestionario, podéis realizar tantos intentos como queráis. Una vez enviada la respuesta ya no se puede modificar, pero sí que podéis realizar un nuevo intento. Tenéis que tener en cuenta que si **volvéis a contestar** el cuestionario, entonces se vuelve a generar la pregunta de forma aleatoria y los ficheros para los que tenéis que hacer la firma son diferentes. Por lo tanto **la firma también será diferente**. Sólo se tendrá en cuenta el último intento en la corrección.

Para preparar este ejercicio os pueden ser de utilidad las siguientes fuentes o indicaciones:

1. En la documentación de ClamAV (https://www.clamav.net/documents/) encontraréis información detallada sobre el formato de firmas que usa ClamAV y algunas herramientas de ayuda para su creación. Algun ejemplo:
   - https://www.clamav.net/documents/creating-signatures-for-clamav
   - https://www.clamav.net/documents/extended-signature-format
   - https://www.clamav.net/documents/file-hash-signatures
   - https://www.clamav.net/documents/logical-signatures
2. Para verificar una firma lo más fácil es guardar o escribir la firma en un fichero (con extensión *.hdb* o *.ldb*) y usar el comando: `clamscan -d <fichero-firmas> <fichero>`

   donde *fichero-firmas* es el fichero que contiene la firma y *fichero* es el fichero del que queréis verificar si contiene el virus o no.
3. No se trata de un malware real, el ejercicio se ha realizado con finalidades educativas. En este caso una forma interesante para encontrar el malware puede ser mirar las diferencias que hay entre el fichero original y el mismo fichero infectado.
4. ClamAV incorpora la herramienta *sigtool*, que os puede ser útil para trabajar. Mirad p.e. las opciones *--test-sigs*, *--decode-sigs*, o *--hex-dump* .

Ejercicio 2 (se entrega vía registro de evaluación continua) – 60% de la nota

El conocido fabricante de yunques ACME tiene una unidad de negocio destinada al desarrollo de software en el ámbito de las tecnologías Blockchain. Estas tecnologías son de las que ofrecen más oportunidades nuevas de negocio, entre otras razones por el éxito de redes como *Bitcoin* o *Ethereum*.

En un esfuerzo para mejorar la seguridad de sus aplicaciones, ACME ha decidido liberar parte del software que se utiliza en el desarrollo de su propia red *Blockchain*. Concretamente ha liberado como módulo independiente aquel que se encarga en cada uno de los nodos que forman la red de añadir una nueva transacción al próximo bloque de la cadena *Blockchain.*

```
[root@localhost ACMEBlockchain]# ./ACMEBlockchain
      ___          _______   __    __       _______
     /   |        / ______/ /  |  /  |      / ______/
    / /| |       / /        /   | /   |     / /____
   / /_| |      / /        / /| |/ /| |    / ______/
  / ____  |    / /____    / / |___/ | |   / /_____
 /_/    |_|   /_______/  /_/         |_|  /_______/

                                    [BLOCKCHAIN]

[*] Usage is: acmeblockchain input_file output_file
```

El programa recibe como entrada un fichero que contiene tanto la transacción pendiente para a añadir a la cadena de bloques como el bloque actual de la cadena *Blockchain*. Después de procesar la transacción y añadirla al bloque actual, como salida se genera otro fichero con la transacción añadida al bloque y del que se ha eliminado la transacción pendiente.

Como analistas de seguridad recibís el encargo de buscar posibles vulnerabilidades en el software. Podéis descargar una copia en:

http://deic.uab.cat/~mistic/ACMEBlockchain.zip

(sha256=7efd32a0cf85429130defb8b3c091326bbfa1fbed15d9253da2567c9139b7dcb)

Como dato adicional ACME ha publicado las especificaciones del fichero de entrada para que sea compatible con sus software:

Field	**Length**
Block Header	32 bytes
Transaction	20 bytes
Block	140 bytes

(NOTA:-> Cada bloque puede almacenar un máximo de 5 transacciones. Si ya está completo obtendremos un mensaje indicativo de que necesitamos un bloque nuevo).

Además para facilitar el trabajo de análisis de seguridad, con el fichero ACMEBlockchain.zip, se suministran dos ficheros de entrada (input_file) de prueba:

- Block-ID10 → Con una transacción pendiente y ninguna en el bloque.
- Block-ID11 → Con una transacción pendiente y otra en el bloque.

Apartado 1 (1 / 10 pt)

Comenzad haciendo algunas pruebas de ejecución del software usando los bloques Block-ID10 y Block-ID11. Posteriormente utilizando el Block-ID11 mostrad dos capturas de pantalla:

1. La primera del fichero original (Block-ID11) indicando los bytes que guardan la cantidad *(coins)* de la transacción pendiente de añadir al bloque.
2. La segunda del fichero resultado (Block-ID11.out) indicando los bytes donde se ha almacenado la cantidad *(coins)* pendiente dentro del bloque.

NOTA:-> Para las capturas, utilizar alguna herramienta que muestre el contenido del fichero en hexadecimal. Tienen que quedar claramente indicados los bytes donde se almacenan las cantidades.

Apartado 2 (1 / 10 pt)

Analizamos el código del programa y vemos que parece haber un *overflow*. Para entenderlo mejor decidís hacer una tabla como la siguiente (rellenarla correctamente). En la tabla tienen que aparecer las variables siguientes:

Platform is {x86_64-linux, i686-widows, x86_64-osx, etc.}						
variable				**memory region**	**memory address**	
type	**name**	**line**	**Size (in bytes)**	**(Stack / Heap / Global / Text)**	**absolute**	**relative (to the first row)**
char*	inputFile					
char*	outputFile					
Struct	header					
Struct	transaction					
Struct	block					

Notas:

1- Podéis usar gdb (o cualquier otro depurador) para obtener esta información. Indicad vuestra elección e la tabla.

2- Recordad indicar la plataforma para la que habéis hecho esta tabla.

3- Las direcciones relativas son el resultado de restar la dirección absoluta de la fila actual y la primera fila. E.g., 0x7fffffffe066-0x7fffffffe070=-0x0A.

4- Ordenad las filas de la tabla por la columna «line» de menor a mayor.

Apartado 3 (1 / 10 pt)

Con la información que ya tenéis de las pruebas realizadas y la información de la tabla anterior, os dais cuenta de que un atacante que intercepte un bloque puede modificarlo para cambiar la dirección de recepción de la transferencia a otra bajo su control. Para demostrarlo utilizar la dirección FFFFFFFF.

Coged el fichero Block-ID11 y haced una copia que nos permita realizad algunas pruebas en apartados posteriores (*ejemplo: cp block-id11 block-id12*).

a) Una vez realizada la copia, identificad la transacción legítima que se quiere realizar:

account-sender	
account_receiver	
amount	

b) Y completad la tabla con la transacción que queremos esconder:

account-sender	
account_receiver	FFFFFFFF
amount	

Apartado 4 (7 / 10 pt)

Con todos los datos que ya hemos recopilado y que tenemos recogidos en los apartados 2 y 3, vamos a llevar a cabo el ataque.

a) Calculad la distancia entre variables necesaria para poder explotar el *overflow*. Justificad los cálculos que habéis realizado.

b) Con los cálculos que hebéis realizado, modificad el fichero de entrada de tal forma que se pueda reemplazar la cuenta *(account_receiver)* de la transacción. Explicad bien todos los pasos que habéis dado y mostrad capturas de pantalla donde se pueda comprobar el resultado final del fichero.

c) Nos damos cuenta que el usuario sigue viendo el mensaje de la transferencia como si no hubiera pasado nada. Como analista de seguridad, qué pequeña modificación del código aconsejarías para reflejar la salida de la consola que se ha enviado dinero a otra cuenta?

Nota: **Propiedad intelectual**

A menudo es inevitable hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica, siempre que esto se documente claramente y no suponga plagio en la práctica.

Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se presentará junto con ella un documento en el que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y el su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.

Nota: esto se refiere al copyright del documento que entregáis en el registro de evaluación continua y no al copyright de las herramientas que podáis haber usado. Por ejemplo, al usar una imagen en la respuesta de un ejercicio debéis seguir lo aquí indicado, pero por el contrario, si usáis el sistema operativo Debian para resolver el enunciado, entonces no es necesario.