UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Tema de Exposición

”Ingeniería Inversa”

Alumno: Pineda Galindo Ricardo Angel

Profesor: Ing. Gunnar Eyal Wolf Iszaevich

Asignatura: Sistemas Operativos

Grupo: 06

Fecha de exposición: 30 de abril de 2024

Contenido

Definición

Es el proceso llevado a cabo con el objetivo de obtener información o un diseño a partir de un producto u objeto, con el fin de determinar cuáles son sus componentes, de qué manera interactúan entre sí y cuál fue el proceso de fabricación.

En el caso de la ciberseguridad hay una rama que se enfoca a eso, con el objetivo de ver cómo es que se ha realizado un determinado programa. Esto puedo ser aplicado con distintos propósitos, pero principalmente hay dos usos que se le puede dar:

1. Poder cambiar el comportamiento de un programa para saltarse alguna protección o verificación y en algunos casos poder copiar el software para sacarle provecho u obtener algún beneficio.
2. Descubrir cómo fue creado algún malware para saber que vulnerabilidad aprovechó y así poder contrarrestarlo en casos futuros o solucionar esa vulnerabilidad.

Sin importar el uso al aplicar la ingeniería inversa, es necesario conocer muy bien el lenguaje ensamblador porque con ayuda de algunas herramientas se pasa el código máquina a código ensamblador.

Se puede encontrar una gran variedad de herramientas instaladas en la distribución Kali Linux, la cual es un sistema operativo que viene configurado con muchas herramientas que son de utilidad en varias ramas de la ciberseguridad.

Los analistas de software utilizan estas herramientas después de aplicar otras técnicas de des-ofuscación de código, ya que el malware actual cuenta con esta tecnología para dificultar el reversing cambiando constantemente las variables, insertando diferentes codificaciones de caracteres o insertando comentarios en medio de las líneas de código.

Tipos de análisis:

Análisis estático:

Consiste en analizar el código en ensamblador sin ejecutar el binario asociado. No existe riesgo de infección puesto que no es necesario ejecutarlo.

Para realizar este análisis es necesario utilizar un desensamblador para poder decodificar el código máquina del malware en lenguaje ensamblador, con esto se podría comprender el flujo de ejecución y saber lo que hace sobre la máquina infectada.

A pesar de ser una buena forma de comprender el comportamiento del malware, en algunos casos puede resultar muy complicado de leerlo porque algunas funciones se codifican y para facilitar el proceso puede resultar mejor ejecutar el binario.

Análisis dinámico:

Consiste en estudiar las acciones del binario durante su ejecución, cabe destacar que cuando se ejecuta el binario, la máquina donde se ejecuta se verá infectada. Por eso es importante utilizar una máquina virtual.

Para poder realizar este análisis, es necesario utilizar un depurador para poder ejecutar un binario instrucción a instrucción, establecer breakpoints, detener la ejecución del binario y consultar su estado.

Técnicas de ofuscación:

La ofuscación consiste en transformar un programa de manera que, aunque mantiene su comportamiento, se dificulta su comprensión. La idea es hacer que el código del programa sea lo menos intuitivo posible, de modo que su lectura resulte compleja.

Esta técnica se ocupa para 3 situaciones:

- Proteger la propiedad intelectual
- Ralentizar los análisis, en el caso del análisis de malware, pues cuanto más tiempo lleve el análisis, más tiempo estará funcional el malware
- Hacer que los antivirus sean menos eficaces, pues tras la ofuscación el archivo binario puede parecer diferente y no corresponderse con una firma existente.

Existen varias formas de ofuscar un código, pero las más utilizadas son:

- Ofuscar las cadenas de caracteres: Una forma muy sencilla es utilizando el algoritmo ROT13, el cual consiste en desplazar 13 caracteres cada letra del alfabeto.

```
rootbsd@lab:~/$ strings herpesnet.exe
[...]
tcerfhygy
uggc://qq.mrebkpbqr.arg/urecarg/
74978o6rpp6p19836n17n3p2pq0840o0
uggc://jjj.mrebkpbqr.arg/urecarg/
sgc.mrebkpbqr.arg
uggc://sex7.zvar.ah/urecarg/
hcybnq@mrebkpbqr.arg
hccvg
ujsdsdbbngfgjhhuugfgfujd
rffggghooo
Ashfurncsmx
[...]
```

Ejemplo del malware HerpesNet utilizando el comando strings

Con la salida tenemos un indicio de que se están utilizando cadenas que tienen un formato parecido a las URLs, entonces utilizando un comando llamado rot13 que está disponible en Linux se obtiene la siguiente salida:

```
rootbsd@lab:~$ strings herpesnet.exe | rot13
[...]
gpresultl
http://dd.zeroxcode.net/herpnet/
74978b6ecc6c19836a17a3c2cd0840b0
http://www.zeroxcode.net/herpnet/
ftp.zeroxcode.net
http://frk7.mine.nu/herpnet/
upload@zeroxcode.net
uppit
hwfqfqooatstwuuhhtstshwq
essttubbb
Nfusheapfzk
[...]
```

- Ofuscación del uso de la API de Windows: No invocar directamente las funciones de la API de Windows por sus nombres, sino por un código que las identifique

El malware Duqu utilizó esta técnica con una función importante, la cual el la llamó FUN_4017E2.

```
00401E34        push    eax
00401E35        push    5FC5AD65h
00401E3A        push    [ebp+viewonNTDLL]
00401E3D        push    [ebp+NTDLLmodhdl]
00401E40        push    [ebp+imports]
00401E43        call    FUN_4017E2
```

Esta función recibe 5 parámetros o argumentos donde el argumento 5FC5AD65 es el identificador de la función que se va a ejecutar y el argumento ebp+viewonNTDLL es un puntero a la biblioteca en memoria. En este caso, la forma en que funciona es que la función va a recorrer toda la biblioteca en memoria y hacer un hash de cada una de las funciones disponibles. Cuando el hash corresponda con el que se pasa como argumento, se devolverá su dirección que en este caso la función que se iba a utilizar era ZwCreateSection().

Como se puede apreciar, este método de ofuscación es algo complicado, por lo que se dará más preferencia al análisis dinámico.

- Utilizar packers que permiten codificar, cifrar o comprimir un binario para disimular su código original.

Esta es una de las técnicas más utilizadas para complicar los análisis de malware. Un packer es una aplicación que permite comprimir, codificar e incluso cifrar un archivo binario sin alterar su funcionamiento y para poder analizar este tipo de malware ofuscado es necesario encontrar o restaurar el archivo binario original, esta operación se conoce como unpack.

Cada packer tiene su propio funcionamiento, pero en la mayoría de los casos, el binario inicial se cargará en memoria completa o parcialmente, de modo que el objetivo del análisis es encontrar este binario en memoria y extraerlo.

En algunos casos, los packers utilizan el heap para almacenar el binario original tras descomprimirlo y el binario inicial se puede utilizar directamente. Otros packers utilizan la pila para almacenarlo, en cuyo caso será necesario aplicar correcciones a este binario extraído de la memoria para hacer que sea utilizable. Por último, algunos packers extremadamente complejos utilizan máquinas virtuales para descifrar el malware.

La técnica que cada vez es más utilizada es Anti-VM, la cual sirve para detectar máquinas virtuales y detener la ejecución del binario para poder impedir que se pueda analizar el malware mediante herramientas como sandboxes. Existen muchas formas para comprobar si un binario se ejecuta sobre una máquina virtual ya que el malware puede controlar el hardware, los drivers, aplicaciones instaladas para controlar la máquina virtual, etc. Además de esto, algunos aprovechan los bugs de algunas herramientas de virtualización como por ejemplo VirtualBox y la vulnerabilidad CVE-2012-3221 que fue descubierta en 2011, la cual consiste en que una máquina virtual falla si se ejecuta una interrupción 0x8.

Para aprovechar esta vulnerabilidad bastaba con agregar la siguiente función y así evitar el análisis del malware.

```
int crash() {
 asm (
   "int $0x8;"
   : // output: none
   : // input: none
   :"%eax", "%ebx", "%ecx", "%edx"   // clobbered register
 );
 return(0);
```

Por eso es tan importante actualizar las herramientas de virtualización.

Además de estas técnicas, existen muchas más y para no consumir tanto tiempo solo las mencionaré:

- Si el malware detecta el uso de puntos de ruptura (breakpoints) entonces se detiene el programa.
- Si hay breakpoints de hardware también se detiene el programa.
- Si el programa tiene controlado el tiempo que tarda en ejecutar ciertas secciones de código y tarda más de lo esperado, el análisis se interrumpe.
- Con ayuda de la API de Windows, el binario puede comprobar si está en depuración o no, y si es el caso se termina el programa.

Herramientas:

- Desensamblador: Convierte el código de lenguaje máquina a lenguaje ensamblador para que pueda ser más legible para las personas.
- Decompilador o compilador inverso: Recrea un código en lenguaje de alto nivel a partir del lenguaje máquina binario.
- Depurador: Es un programa que permite el control de otro programa, dando opción a analizarlo paso a paso para poder ir viendo el estado de las variables, del uso de memoria y afectación del programa en el equipo.
- Editor Hexadecimal: Es un programa que permite editar archivos binarios.
- Detector de packers: Este tipo de herramienta muestra si el programa está protegido por un packer y que packer es .

Ejemplos de herramientas:

- Ghidra: Es una herramienta de análisis estático hecha pública por la NSA (Agencia Nacional de Seguridad) a inicios del 2019. Puede desensamblar archivos binarios y contiene un descompilador que permite obtener un pseudocódigo C del binario que se está analizando.
- Immunity Debbuger: Es un depurador de 32 bits para Windows que permite ver las instrucciones ejecutadas, el estado de la memoria y de los registros. La ventaja que tiene es que permite el lenguaje de Python para automatizar ciertas tareas.
- IDA: Es una herramienta muy completa que permite realizar tanto análisis estático como dinámico, también tiene desensamblador, decompilador, debugger, capacidad para renombrara variables, crear scripts para automatizar procesos, etc.
- x64dbg: Es un debugger que soporta arquitecturas de 64 y 32 bits.
- HxD: Es un editor hexadecimal

Aplicaciones:

Es importante mencionar que el enfoque de ingeniería inversa es muy distinto en todas las áreas donde se utiliza, lo único que comparten es la idea de que a partir de algo se puede saber como se compone y si fuera necesario poder replicarse a partir de toda la información obtenida durante el proceso.

<u>En la industria</u>

- Recreación de productos que no tienen dibujos 2D o datos CAD 3D para poder reproducirlos.

- Problemas con el fabricante de equipos originales (OEM): Si el OEM ya no opera o ha perdido las medidas de diseño, entonces a través de la ingeniería inversa se puede obtener información vital del producto para continuar con la fabricación de ese objeto.
- Análisis de la competencia: Cualquier organización puede analizar los productos de la competencia y así poder robar sus conocimientos.
- Objetos antiguos y a medida: Cuando no hay información sobre las dimensiones de un objeto, excepto el elemento físico en sí, la forma más rápida es utilizando la ingeniería inversa porque cuando el producto tiene una forma orgánica puede ser difícil diseñarlo en CAD.

Ejemplo del uso de ingeniería inversa en una pieza antigua

<u>En la ciberseguridad</u>

- Análisis de malware: El ingenier@ especializado utiliza distintas técnicas de ingeniería inversa para desmontar y comprender el funcionamiento interno de programas maliciosos.
- Retroingeniería: Se emplea para obtener información sobre tecnología patentada o productos existentes en el mercado con el objetivo de desarrollar productos similares o compatibles. Esta práctica es legal siempre y cuando no infrinja los derechos de propiedad intelectual.
- Identificar y comprender vulnerabilidades en software o sistemas.
- Destrucción de protección anticopia: Algunas personas utilizan estas técnicas para poder copiar videojuegos o copiar música bajo la protección de un sistema anticopia (DRM, Digital Rights Management).
- Interoperabilidad: Los desarrolladores utilizan las técnicas de reverse engineering para reescribir tareas que permitan utilizar los productos sobre otras plataformas diferentes a la soportada por el fabricante (es el caso de muchos drivers en Linux).
- Control de calidad de las funciones del software

Ejemplo: Análisis del malware WannaCry utilizando técnicas de reversing

Este malware es un ransomware que se introduce en dispositivos informáticos, infectándolos y bloqueándolos de manera parcial o total hasta que se abone la cantidad de dinero exigida.

Se produjo el 12 de mayo de 2017 e infectó a más de 230 mil computadoras en 150 países, afectando a instituciones estatales y empresas a nivel mundial.

En todas las computadoras infectadas se mostraba en pantalla el siguiente programa:

Este ransomware un exploit de Windows llamado EternalBlue, el cual fue un descubierto por la NSA de Estados Unidos y en lugar de notificarlo a Microsoft prefirió mantenerlo en secreto para poder aprovecharlo.

Esta vulnerabilidad se encuentra en la implementación del protocolo Server Message Block (SMB). Lo cual permitía que se aceptaran paquetes específicos de atacantes remotos, permitiendo ejecutar código en el ordenador de forma remota.

Este exploit fue robado por un grupo que se hacía llamar Shadow Brokers en el mes de abril de 2017, un mes después de que Windows emitiera el parche para solucionar esa vulnerabilidad, pero como muchas computadoras no se habían actualizado con el parche se pudieron infectar.

La forma en que funcionó el malware fue que se comporta como un gusano, es decir, se propagaba a través de las redes usando un protocolo de uso compartido de archivos llamado SMBv1 que permite a las computadoras comunicarse con otros dispositivos conectados a la misma red. Una vez que WannaCry se instala en un equipo, escanea la red buscando direcciones IP de manera aleatoria y se propaga utilizando la vulnerabilidad mencionada anteriormente.

¿Cómo se solucionó? El investigador de ciberseguridad Marcus Hutchings descubrió que una vez que WannaCry se alojaba en un sistema, intentaba ponerse en contacto con una dirección URL concreta (http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com/) y si no hallaba esta dirección entonces el ransomware procedía a infectar el sistema. Lo que hizo Marcus fue registrar un nombre de dominio para crear un sumidero DNS, el cual es un servidor DNS que intercepta solicitudes de DNS que intentan conectarse a dominios maliciosos y devuelven una dirección IP falsa.

Con esto se detuvo el ransomware pero hubo una gran pérdida monetaria, de hecho, la empresa de modelado de riesgo cibernético Cyence estimó que el coste total fue de 400 millones de dólares.

Conclusiones

La ingeniería inversa siempre ha tenido un papel muy importante a lo largo del tiempo en distintas áreas, pero en este trabajo se dio prioridad al sector de la tecnología involucrada con la evolución de las computadoras y todo lo relacionado a estas, porque de lo contrario, hemos visto el impacto económico y social que puede generar.

Actualmente con el repentino avance que se ha tenido estos últimos años, las vulnerabilidades se han hecho más presentes ya que debido al desconocimiento del tema, los ciberdelincuentes han abusado de los exploits en momentos clave para obtener mucha información incluso de empresas muy grandes.

La tarea de analizar programas cada vez se vuelve más compleja debido a que ya existen muchas formas para ofuscar un programa, lo cual hace que el analista además de poder descifrar el objetivo del programa deba ser capaz de poder quitar todas las capaz de protección del programa y así poder prevenir daños o reducirlos. Esto se traduce a que la gente que trabaja en esto tenga una gran responsabilidad y certeza para encontrar todos los problemas.

Referencias:

- Cortés, A. (2021). Estudio del malware WannaCry utilizando técnicas de reversing (Tesis de grado). Universidad de Alcalá Escuela Politécnica Superior. https://ebuah.uah.es/dspace/bitstream/handle/10017/49194/TFG_Cortes_%20Arranz_2021.pdf?sequence=1&isAllowed=y
- Ingeniería inversa para maquinaria y procesos. (2023). ALTERTECNIA. https://altertecnia.com/ingenieria-inversa-maquinaria-y-procesos/
- La ingeniería inversa de software. (2020). IONOS. https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/ingenieria-inversa-de-software/
- Lares, D. (2022). Malware Sandboxing and Reverse Engineering Overview. Medium. https://medium.com/nerd-for-tech/malware-sandboxing-and-reverse-engineering-overview-66f6ad84e8c9
- Latto, N. (2020). ¿Qué es WannaCry?. Avast. https://www.avast.com/es-es/c-wannacry
- Meza, P. (2023). Reversing. IHack. https://ihack.red/reversing/
- Qué es y para qué sirve la ingeniería inversa. (2023). Asorcad. https://asorcad.es/blog/que-es-y-para-que-sirve-la-ingenieria-inversa/
- Rascagneres, P. (2020). Seguridad informática y malwares: análisis de amenazas e implementación de contramedidas (2nd ed.). Editorial ENI.
- REDACCIÓN. (2020). Aplicaciones industriales más comunes de la ingeniería inversa. Tecnología Para La Industria. https://tecnologiaparalaindustria.com/aplicaciones-industriales-mas-comunes-de-la-ingenieria-inversa/
- Serra, J. (2021). Técnicas y herramientas la ingeniería inversa. Tecnología++. https://blogs.uoc.edu/informatica/es/ingenieria-inversa-que-es-herramientas-y-tecnicas/
- yadav, G. (2021). A brief introduction to Packing and Obfuscation. https://medium.com/ax1al/packing-and-obfuscation-fe6b03bbc267