Seguridad en servidores: explotación de vulnerabilidades

Ingeniería de servidores
Universidad de Granada

Resumen

La explotación de vulnerabilidades es un concepto que lleva en el mundo de la informática prácticamente desde que esta surgió, pero que cada vez está adquiriendo mayor importancia por la rápida informatización de las empresas. Son muchas las empresas que han sufrido grandes pérdidas económicas por no estar suficientemente protegidas o simplemente por la aparición de nuevas vulnerabilidades de las que aún no hay información, llamadas 0-days. Cuando estas vulnerabilidades dejan de ser desconocidas y se resuelven por medio de parches, pasan a llamarse exploits, y existen herramientas muy potentes como Metasploit Framework que nos permiten usarlas para aprovecharnos de sistemas desactualizados o mal protegidos. Aunque hay multitud de exploits listos para usar, desarrollar uno es una tarea complicada, ya que requiere encontrar una vulnerabilidad, conocimiento de ensamblador y del funcionamiento de la memoria, y muchas horas de trabajo; pero aún así van descubriéndose nuevas vulnerabilidades día a día. Para evitar la gran mayoría de ataques hay diversas medidas de prevención, predicción y restauración que pueden ser de gran ayuda para empresas que dependan altamente de sus sistemas informáticos. Siempre que se analice la seguridad de un sistema es necesario informarse sobre la legalidad de lo que se quiere realizar.

1. Introducción

Hoy en día la seguridad se ha vuelto una parte importantísima para cualquier aplicación, y no solo para ella, sino para todas las personas y sistemas que están relacionadas directa o indirectamente con ella. Es una de las ramas de la informática que más rápido avanza y es muy difícil estar informado sobre las vulnerabilidades que van aparecieno día a día; sin embargo, como veremos en los siguientes apartados, es esencial estar preparado para cualquier amenaza. Todos los equipos están expuestos a multitud de amenazas, y el peligro de estas dependen enormemente de cómo estén preparados dichos equipos para actuar frente a ellas. Esto es importante ya que actualmente la gran mayoría de personas necesitan de alguna aplicación informática para su vida personal y laboral (según un estudio del Instituto Nacional de Estadística, durante 2014 usó internet sobre un 76 % de la población española [31]) por lo que es totalmente necesario mantenerlas disponibles el mayor tiempo posible y asegurando que todos los servicios que nos ofrecen son fiables. Ahí es donde recae la importancia de mantener un servidor bien protegido, un descuido en la seguridad que pudiera parecernos ridículo podría ser su punto débil ante amenzas, dejando a multitud de clientes inoperativos y/o suponiendo importantes daños económicos.

1.1. Introducción a tipos de vulnerabilidades

Actualmente existe una gran variedad de vulnerabilidades, sin embargo para introducirnos en el mundo de la seguridad veremos de forma resumida cuatro ejemplos de vulnerabilidades muy utilizadas:

- **Buffer Overflow** (Desbordamiento de buffer): Ocurre cuando una aplicación no comprueba el número de bytes que se van a almacenar en una dirección de memoria que anteriormente

fue reservada. Se aprovecha este caso para introducir una cantidad de bytes más grande de la reservada en el que se incluye código ejecutable que se posicionará en la siguiente dirección de memoria [34].

- **SQL Injection**: Es una de las vulnerabilidades más fáciles de realizar, pero también una de las más potentes en el campo de la informática. Simplemente se basa en introducir código SQL en el nivel de validación de una aplicación, es decir, introducir código SQL en formularios que harán consultas o modificaciones en la base de datos que utiliza dicha aplicación. Mayormente se utiliza en aplicaciones web, aunque también puede ocurrir en aplicaciones de escritorio. [34].
- **Format String Bugs**: Se trata de un tipo de vulnerabilidad que tiene consecuencias similares al Buffer Overflow. En pocas palabras, se basa en no comprobar el número o tipo de argumentos que se pasan por pantalla, imprimiendo directamente lo que se ha mandado (muy relacionado con las funciones de la familia **printf()**. Esto puede ocasionar que dichas funciones desvelen información sobre la memoria o la pila, ya que como argumento se pueden mandar tokens como %s(muestra string) o %x(muestra hexadecimal). [4, 33]
- **XSS** (Cross-site scripting): Es otra de las grandes vulnerabilidades conocidas en el mundo de la informática. Básicamente se basa en añadir código malicioso (en lenguaje Javascript o similar) en una web visitada por un usuario. Este código manipulará la web para que al usuario se le presente de forma diferente a como es la original, intentado que el usuario introduzca datos confidenciales y que serán robados o manipulados por el atacante. Como es de suponer, afecta principalmente a aplicaciones web y concretamente a webs dinámicas que no comprueban los datos que introduce el usuario [34]. Para ver ejemplos sobre esta vulnerabilidad Google nos ofrece una web donde podemos probar a introducir código malicioso y ver el resultado por nosotros mismos [3].

1.2. Cómo afecta la seguridad a las empresas

No hace falta estar muy informado para saber que grandes empresas como Apple, Microsoft, Facebook, Google,... sufren continuamente ataques informáticos, pero la pregunta es: ¿qué se consigue con ello?. La seguridad en empresas es muy delicada ya que un ataque informático puede ocasionar que se revelen datos de los trabajadores y clientes, o simplemente pueden hacer que el servidor deje de funcionar, lo que ocasionaría que todos los clientes no puedan usar un servicio por el que han pagado, o dando con ello una ventaja a la competencia. Esto suele ocasionar pérdidas económicas inmensas, relacionadas con el daño que se produce en el momento del ataque y por las consecuencias que ocasiona, ya que suele golpear de forma contundente la confianza de sus clientes. Según fuentes como [2], en España, el 91 % de las pequeñas y medianas empresas (pymes) sufren ataques a diario, donde el 33 % son troyanos y el 25 % es spyware; y normalmente proceden de páginas webs poco seguras, con un 39 % o simplemente de descargar programas o correo electrónico infectados. Otras fuentes, como la británica [10] aseguran que los ataques informáticos cuestan a las empresas de su país unos 34.000 millones de libras.

1.3. Casos históricos

- Ataque a **Sony Pictures (2014)**: fue un ataque que paralizó los sistemas informáticos de la compañía durante varios días filtrando multitud de información sobre correos electrónicos privados e información sobre registros financieros de ejecutivos de Hollywood. Fue ocasionada por una polémica película (The Interview) donde intervenía el líder de Corea del Norte. Fuentes del FBI acusan a hackers norcoreanos mientras que otras fuentes acusan a ex-empleados de Sony que tan solo buscaban venganza [11, 9, 8].
- Ataque a **Apple**: Apple confirma que se han colado unas 40 aplicaciones infectadas en su Apple Store, entre las que destacan WeChat, Angry Birds 2 y WINZIP, entre otras. Ante esto tuvieron que retirar dichas aplicaciones y corregir los problemas de seguridad para volverlas a colocar en la tienda oficial de Apple [7].

Podríamos seguir con una lista interminable de casos, pero no es nuestro propósito; podemos concluir que la seguridad en las empresas es un aspecto fundamental en la actualidad, y se sabe que los ataques se van a incrementar con el paso de los años, por lo que habrá que invertir aún más esfuerzo para combatirlos. Aún así, es imposible estar completamente protegido, prácticamente todos los días se descubren nuevas vulnerabilidades que no figuran en ninguna base de datos, esto es una gran ventaja para los atacantes que intentarán aprovecharlas rápidamente antes de que sean corregidas.

2. Exploits

2.1. Definición

Un exploit es una secuencia de acciones, comandos, código escrito, que tiene el fin de aprovechar una vulnerabilidad. Por medio de un error de un sistema, programa, etc, busca obtener un acceso privilegiado en un sistema, provocar una deficiencia en el servicio (ataque de denegación de servicio), o cualquier fin consecuente del aprovechamiento de una brecha en la seguridad del sistema. [34]

Es inevitable hablar de *payload* a la hora de hablar de exploits. El payload contiene el código que desea ejecutarse en la máquina vulnerada mediante el exploit, es el código que se desea "ìnyectar". El objetivo puede ser devolver una shell al atacante, o abrir algún tipo de "backdoor" que pueda proporcionar un acceso al atacante, o incluso extraer información relevante, deteriorar el servicio que ofrece la máquina, usarla como "bot", o como medio lograr llegar a otro objetivo (por ejemplo, en una red empresarial).

2.2. Exploits conocidos

Es enorme la cantidad de exploits que se conocen en la actualidad, todos ellos fruto del trabajo de investigación, tanto con buena como con malas intenciones, que se ha ido realizando a lo largo de la historia. Todos los días se descubren nuevas brechas en la seguridad, a veces se descubren fallos en software moderno y puntero, los cuales pueden afectar a un gran número de sistemas, tanto servidores como usuarios, y otras algo más concretos o en versiones menos utilizadas.

Existen repositorios online que tratan de mantenerse al día con las nuevas apariciones. El más conocido es *exploit-db* [35] (en la fecha con más de 35.000 exploits registrados). En ellos podemos encontrar una extensa lista de exploits ordenados por fechas y sistemas a los que afecta.

Hay otras también con bastantes registros como *0day.today*. [1]

2.3. Un terror constante: 0-day

Una vulnerabilidada Zero Day es una vulnerabilidad desconocida, y por lo tanto para la cual aún no hay parches ni actualizaciones que la cubran. Puede ser una vulnerabilidad, como se ha dicho anteriormente, importante y que afecte a muchos sistemas de empresas o usuarios y que estos lo desconozcan. Día a día se pueden descubrir nuevos 0-days, muchos de ellos se informan y pasan a formar parte de los exploits conocidos, para los cuales los fabricantes de software tratan de sacar una solución lo antes posible, de modo que se puedan ver afectados el menor número posible de usuarios. Pero un 0-day es información "privilegiada", los afectados y los desarrolladores pueden no saber de sus existencia aún. Un 0-day puede causar un gran daño a un sistema, o una ventaja económica para un competidor o atacante.[34]

Existe todo un mercado negro de 0-days en la red. Webs como *0day.today* [1], mencionada anteriormente, vende 0-days por precios que pueden rondar los 5.000$. Es por ello que algunas empresas retan a los usuarios a encontrar 0-days en sus sistemas. Por ejmeplo, un 0-day encontrado en iOS fue pagado desde Apple por 1.000.000€.[5, 6]

2.4. Frameworks: Metasploit, Core Impact y Canvas

Hay frameworks que ofrecen un entorno completo para analizar la seguridad de las máquinas y sistemas ("pentesting"), permitiendo probar los distintos exploits existentes (los cuales almacena de forma clasificada), con facilidades para administrar las máquinas vulneradas, aplicar payloads, desarrollar y ejecutar exploits y payloads, etc. [33, 34]

Uno de los más conocidos es *Metasploit*. Es un proyecto Open Source compuesto de módulos con distintas funcionalidades, los cuales pueden ser desarrollados por los usuarios y la comunidad, dándole así un gran potencial. Es una herramienta usada también a nivel profesional. Respecto a la interfaz: los módulos pueden ser usados directamente, o desde Metasploit. Metasploit dispone de varias interfaces gráficas (*Armitage*, *web UI*), pero también puede ser usado desde la consola (*msfconsole*, *Msfcli*). Además se incluyen herramientas y plugins, como *Msfpayload* (capaz de generar payloads en cualquier lenguaje), *msfencode* (codificar payloads para dificultar su detencción), etc. El núcleo de Metasploit está compuesto por bibliotecas, de las cuales dependen el resto de funcionalidades. [34]

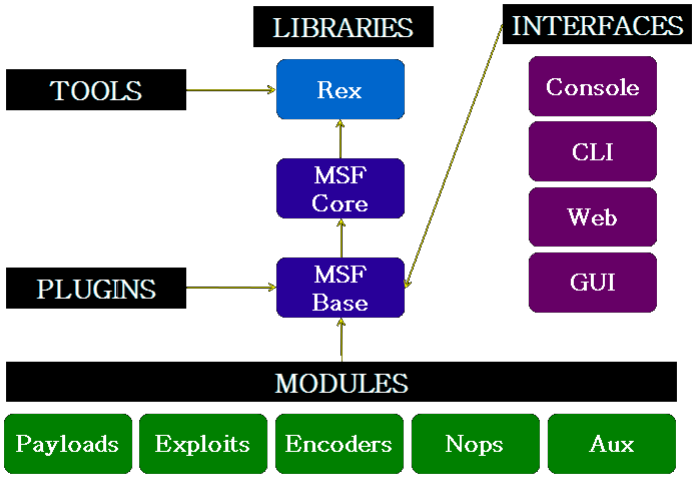

Figura 1: Arquitectura Metasploit Framework [34]

También hay que mencionar entornos como Core Impact o Canvas, los cuales también son bastante potentes y ofrecen distintas ventajas en las auditorías de seguridad. El inconveniente es que, por ejemplo, Core Impact cuesta unos 40.000$, y no cuenta con la misma facilidad que Metasploit para que los usuarios desarrollen y distribuyan módulos, pero en cambio cuenta con el soporte de los desarrolladores (lo cual también lo incluye la versión express y pro de Metasploit, por un precio inferior) [14, 21, 13, 24]

2.5. Auditando un servidor con Metasploit Framework

Lo que haremos en esta sección será utilizar Metasploit Framework (mencionado anteriormente) para probar un exploit ya desarrollado sobre un laboratorio virtual llamado "Metasploitable 2". Este laboratorio consiste en una máquina virtual preparada para ser vulnerada mediante un exploit. Para ello podemos utilizar Vmware, VirtualBox o cualquier software de virtualización similar, aquí hemos optado por VirtualBox [18]. Para ello podemos seguir el manual recomendado por rapid7 (empresa desarrolladora de Metasploit) que está disponible en [16].
Las vulnerabilidades que contiene metasploitable 2 son:

- Servicios mal configurados: configuraciones por defecto o sin las debidas precauciones de seguridad.
- Backdoors: posibilidad de acceso a un sistema remoto evitando autentificación.

- Contraseñas inseguras: contraseñas cortas o típicas, de modo que son posibles de averiguar mediante ataques como fuerza bruta.
- Servicios webs vulnerables: servicios webs en los cuales existe algún tipo de error de programación, provocando que el sistema sea vulnerable.

Podemos verlo de forma más detallada en Offensive Security [18].

Para probar estas herramientas se ha optado por realizar la instalación siguiendo el manual descrito anteriormente, y configurando la red con una interfaz “Host-only”. Una vez instalado el sistema iniciamos sesión como superusuario:

- **Usuario**: msfconfig
- **Contraseña**: msfconfig

Podemos comprobar que la máquina virtual se ha instalado correctamente realizando un ping desde la máquina anfitriona, veamos el ejemplo en Figura2. Si no obtuviéramos resultado, podría ser debido a que no tenga ninguna IP asignada, en nuestro caso la hemos asignado manualmente.

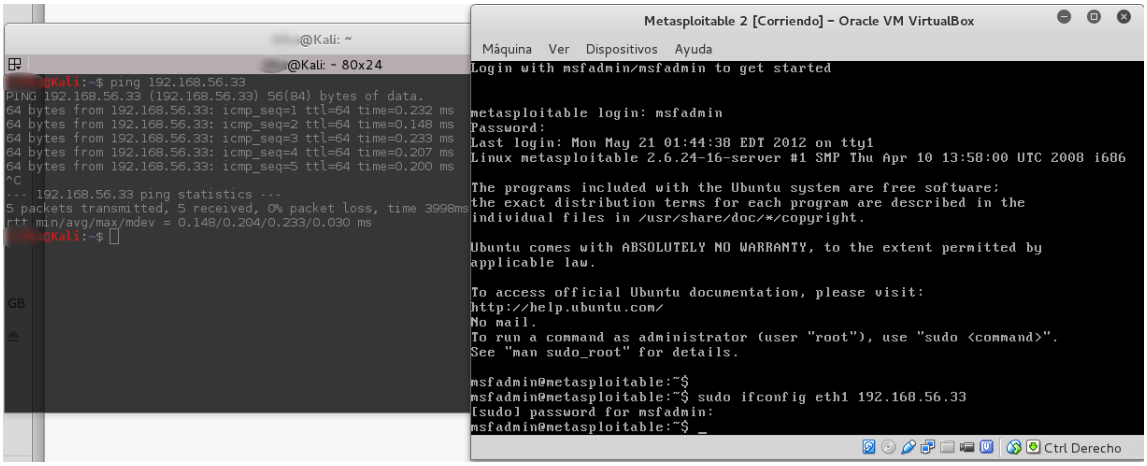

Figura 2: Metasploitable2 funcionando en Virtual Box y conectado con un sistema operativo anfitrión (Kali) [34]

En primer lugar vamos a ejecutar **Nmap** sobre el objetivo para averiguar qué servicios tiene activos y por consiguiente qué puertos tiene abiertos. Según su web oficial [17, 12] esta herramienta es capaz de explorar una red y hacer una auditoría de seguridad, esto nos será de gran ayuda ya que nos dirá qué puertos tiene abiertos el sistema al que queremos acceder. En nuestro caso usaremos la siguiente orden:

```
sudo nmap -sV -O -p "*" 192.168.56.33
```

En el anterior comando estamos indicándole a Nmap que nos analice todos los puertos (-p “*”) junto al servicio que usa dicho puerto (-sV) y el sistema operativo (-O) que está corriendo en la máquina solicitada. En nuestro caso esta máquina será 192.168.56.33 que es la metasploitable2 que instalamos anteriormente. Podemos ver El resultado de nmap

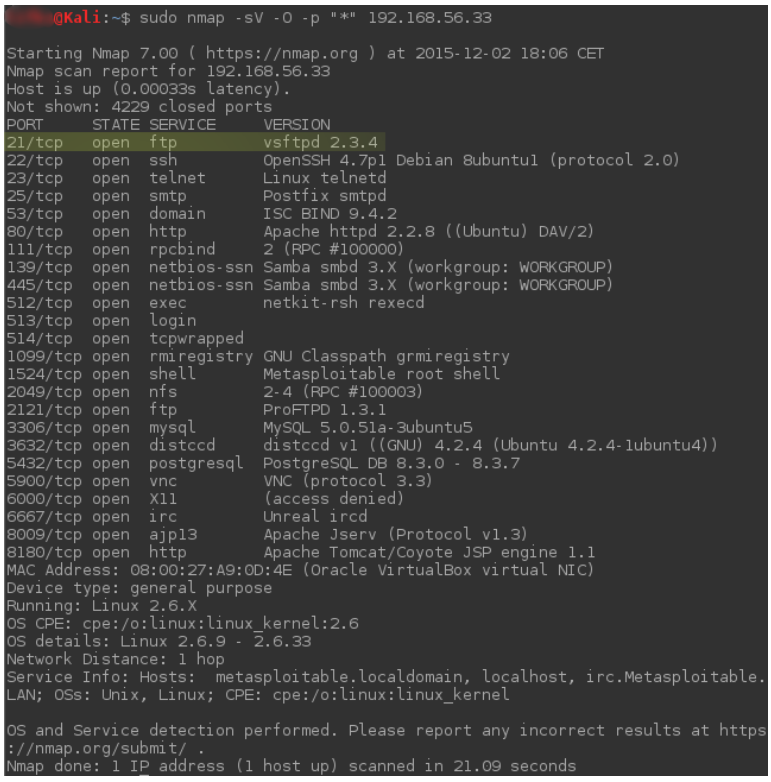

Figura 3: Analizando puertos de la máquina objetivo mediante *Nmap* [34]

Tras analizar el resultado de Nmap podemos ver que el puerto del servicio ftp (vsftpd) se encuentra abierto. Para comenzar se ha decidido buscar algún exploit que sea capaz de aprovechar la vulnerabilidad presente en dicha versión (2.3.4). Para buscarlo lo primero que debemos hacer es actualizar la base de datos de Metasploit mediante el comando **msfupdate** y tras dicha actualización iniciaremos una sesión en Metasploit en modo consola mediante el comando **msfconsole**. Una vez iniciada la sesión buscaremos en la lista de exploits disponibles si existe alguna vulnerabilidad para el servicio que hemos seleccionado. Para ello ejecutaremos el comando:

```
search vsftpd
```

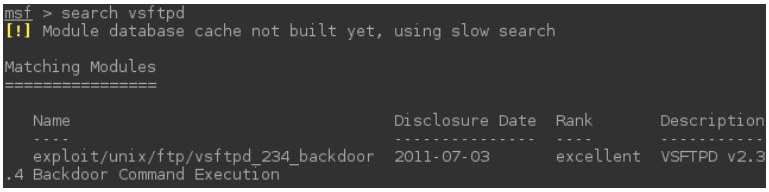

Figura 4: Buscando un exploit en Metasploit para un servicio seleccionado, *vsftpd* [34]

Como podemos ver en la figura anterior (Figura4) Metasploit ha encontrado una vulnerabilidad para la versión que tenía instalada nuestra máquina objetivo, por lo que vamos a proceder a lanzar un ataque por medio de este exploit. Para seleccionar el exploit deseado utilizaremos el comando **use** indicando la ruta del exploit que se va a utilizar:

```
use exploit/unix/ftp/vsftpd_234_backdoor
```

Puede observarse que Metasploit organiza los exploits en función del sistema operativo y del servicio al que pertenece la vulnerabilidad, lo cual nos podría facilitar la búsqueda de exploits escalando por sus directorios.

Una vez seleccionado el exploit tenemos que configurar sus variables, estas se configuran mediante el comando **set**. En este caso bastará con configurar la variable que indica la dirección de nuestra máquina objetivo (*RHOST*) y el payload que se va a utilizar.

```
set RHOST 192.168.56.33
```

Para configurar la variable *PAYLOAD* será necesario localizar un payload adecuado para este caso. Para ello Metasploit nos ofrece una lista de payloads compatibles con el exploit seleccionado previamente, esto podemos realizarlo mediante el comando **show**.

```
show payloads
```

En este caso Metasploit solo nos muestra un payload, que será el que se va a utilizar mediante la orden **set**.

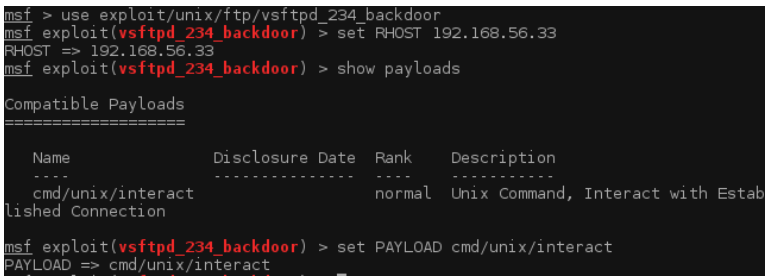

Figura 5: Seleccionando exploit a utilizar y configurando las variables

```
set PAYLOAD cmd/unix/interact
```

Como podemos ver el directorio *cmd/* es el directorio que contiene todos los payloads que tiene disponible Metasploit, a diferencia de los exploits que se encuentran en el directorio *exploit/*.
Una vez hecho esto ya solo tenemos que lanzar el exploit, esto lo haremos mediante la orden **exploit**, y si todo ha ido bien deberíamos haber accedido al servicio vsftpd con privilegios de super usuario, podemos ver un ejemplo en la siguiente figura (Figura6).

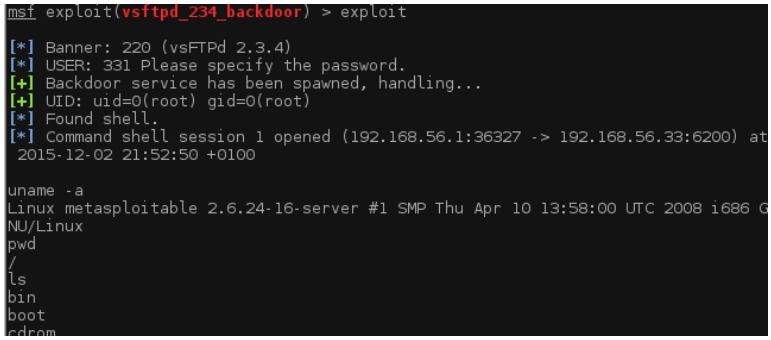

Figura 6: Lanzando un exploit

Esto es solo un ejemplo sencillo de lo que se podría hacer por medio de un exploit. El laboratorio virtual que hemos instalados nos ofrece muchas más posibilidades. Por motivos de espacio no es posible realizar un análisis exhaustivo, para conocer más acerca de esto se recomienda consultar algunos manuales, como [19, 23]. La propia empresa de Metasploit nos ofrece de forma resumida los comandos de la consola **msfconsole** utilizada anteriormente [20] con ejemplos prácticos para entenderlos mejor.

2.6. Cómo se desarrolla un exploit

Desarrollar un exploit no es una tarea sencilla. Es necesario comprender el funcionamiento de la memoria y tratar de manipular el comportamiento del programa para lograr que use la memoria y se dirija a donde nosotros deseemos. Nuestro objetivo es inyectar un código en la memoria (payload) y lograr que el software vulnerado lo ejecute.

Para lograr lo anterior, es necesario encontrar un medio que nos permita manipular el funcionamiento del programa a nuestro antojo, y este medio es una vulnerabilidad, un error de programación (bug) que nos proporcione un acceso a la memoria. Para ello es importante conocer en qué consiste la vulnerabilidad y qué hace exactamente. Se recomienda analizar la vulnerabilidad por medio de un depurador y desensamblar la aplicación, de modo que podamos determinar con precisión qué es lo que ocurre exactamente, si se trata de una instrucción ilegal, un fallo de segmentación, etc. Conocer el tipo de error nos interesa más que saber el lugar exacto en el que se localiza, ya que en función del error podremos obtener ciertas ventajas sobre el sistema. También nos interesa conocer si esta vulnerabilidad viene dada por una entrada del usuario. Analizar los registros del procesador nos puede llevar a determinar qué estado es generado por un proceso determinado del software.

Un registro decisivo es *EIP*, este nos indica cuál es la siguiente instrucción a ejecutar. [27] Si *EIP* apunta al problema es debido a que se trata de un acceso no autorizado a memoria, pero sin embargo, si contiene una instrucción desconocida, nuestra labor debería ser determinar si se encuentra apuntando a la sección de código o datos. Si se encuentra apuntando a código, el problema probablemente sea debido a otro registro, por lo que tendríamos que analizarlos todos. Pero si apunta a datos, habría que determinar si es posible inyectar código en esa referencia, siendo así podríamos obtener un exploit. De no ser así se podría tratar de averiguar el motivo que lo ha llevado a apuntar a esos datos concretos, y tratar de averiguar cómo hacer que apunte a datos proporcionados por el usuario (modificando*EIP*). Analizando el valor de los registros podemos tratar de averiguar cómo provocar que *EIP* apunte a otro lugar.

Si todo lo anterior falla, hay que probar buscando cualquier operación de escritura que podamos controlar de un modo u otro. Para esto sería necesario investigar cómo se calcula la dirección de escritura, y sobretodo, si se trata de datos que el usuario pueda manejar de algún modo mediante los datos de entrada. Se desea encontrar cualquier situación que nos pueda llevar a escribir en memoria en una dirección deseada. Se podría escribir en memoria mediante desbordamiento de búfer y vulnerabilidades de cadanas de formato, con el objetivo de modificar la dirección de retorno guardada en la pila, de modo que apunte hacia nuestro código, el problema es que esta dirección sea siempre fiable. Aunque esto se logre la probabilidad de éxito no es segura, ya que aunque se consiga alojar código en una posición de memoria, y que el programa acceda a una referencia deseada, la memoria es difícil de predecir. Hay que tratar de lograr usar referencias relativas para conseguir un acceso indirecto a la memoria, y así no vernos afectados por los desplazamientos de memoria. [36]

3. Prevención, detección y recuperación

Tener nuestro sistema completamente seguro es prácticamente imposible, sin embargo existen muchas formas de prevenir la mayoría de los ataques existentes. La primera es **evitar instalar servicios que no vamos a ofrecer** en nuestro servidor, algo tan trivial pero que tantas veces de incumple. Esto es un problema ya que, como se vio en secciones anteriores, cada uno de estos servicios abre un puerto, y ese puerto es una puerta de entrada para posibles amenazas [32]. La segunda recomendación es mantener los **servicios actualizados** a las últimas versiones, ya que estas contienen multitud de parches que corrigen los posibles agujeros de seguridad que se hayan encontrado en sus versiones precedentes. Otra recomendación muy importante es mantener el **firewall bien configurado**, de manera que bloquee todos los intentos de conexión a servicios que no queremos ofrecer en la red [15].

Una vez tengamos el firewall configurado sería interesante que algún programa nos avisara cuando detecte actividad indebida o anómala en nuestro sistema, como por ejemplo que se detecte una comprobación de puertos o se intente identificar el sistema operativo (como vimos en la sección 2.5, Nmap hace precisamente eso), o simplemente detecte que se esta intentado iniciar sesión demasiadas veces desde la misma máquina. Para llevar este control existen los denominados IDS (**Sistemas de detección de intrusos**) encargados de avisarnos de todo ello [22]. Existen dos tipos de IDS, los basados en red y los basados en host. En resumidas palabras, los primeros actúan de forma parecida a los *sniffers* examinando todo el tráfico de paquetes [30],

mientras que los segundos actúan analizando la información generada por las herramientas de auditoría del propio host (logs, registros de actividad, etc) [29]. Ambos pueden instalarse en nuestro sistema para aumentar el grado de seguridad. Aunque la mayoría de ellos son comerciales (Cisco, ISS, ISA SERVER, etc) también podemos encontrarnos otros IDS Open Source como Bro o Snort [28].

En cuanto al apartado de *recuperación*, tenemos que tener en cuenta que continuamente hay ataques hacia nuestro sistema y que tarde o temprano alguno de los ataques podría llegar a ser exitoso. Hay que estar preparado para ello, la mejor forma de actuar es realizando copias de seguridad para que cuando ocurra se pueda recuperar el estado del sistema lo más rápido posible. Actualmente hay multitud de software especializado en esto, como por ejemplo: Acronis [25], AOMEI Backupper Professional [26], etc.

4. Legalidad

Como hemos estado viendo en secciones anteriores, analizar la seguridad de un sistema puede llegar a ser una tarea compleja y hacerlo de forma legal es más difícil aún, ya que puede desembocar en multas económicas inmensas o incluso años de cárcel. Por ejemplo, comprometer la seguridad de un sistema (aunque no se hayan obtenido datos), puede conllevar una multa o hasta 1 año de cárcel, 10 años de cárcel si no es la primera vez. Esa misma pena podría afectar también a quien realice ataques de Denegación de Servicio (DDoS) a agencias gubernamentales (según la ley federal de Estados Unidos, en 2005) [36]. Es muy difícil establecer un límite entre lo legal y lo ilegal en este tema, por lo que antes de actuar deberíamos informarnos de la ley vigente del país concreto en el que se realiza. Antes de realizar cualquier práctica o auditoría, esta debe ser recogida claramente en un contrato y reconocido por todas las partes afectadas. Existen multitud de libros que diferencian entre ambas formas de ver el “hacking”, por ejemplo el libro titulado Hacking Ético [36].

5. Conclusión

Como conclusión podemos hablar de muchos temas, lo primero es que ningún sistema está exento de ataques informáticos en la actualidad. Podemos tener el *firewall* más robusto que existe, o el software más caro de prevención y detección de ataques, que siempre irán apareciendo nuevas vulnerabilidades que multitud de atacantes querrán aprovechar para conseguir los máximos beneficios posibles. Entre que una vulnerabilidad es descubierta (0-day) hasta que se crea un parche que la soluciona, muchas empresas pueden perder cantidades importantes de dinero, y como podemos ver esto ocurre muy a menudo.

Además podemos concluir que usar exploits ya diseñados es bastante sencillo (como vimos con Metasploit), incluso existen escáneres que realizan la tarea de análisis de forma automática, y que los atacantes pueden emplear. Pero crear nuevos exploits que afecten a sistemas actualizados es bastante más complicado ya que hay que probar multitud de combinaciones de ataques posibles para encontrar algún *bug* que nos deje acceder a la memoria para inyectar el código malicioso.

Por esta razón no debemos escatimar en la seguridad en nuestro servidor, aunque se descubran nuevas amenazas diariamente, si somos capaces de mantenernos al día y evitar los ataques que circulan por la red tomando las medidas adecuadas, podremos evitar grandes costes y mantener la calidad de nuestro servicio.

Referencias

[1] 0day.today. http://es.0day.today/, Online; visto el 1 de Diciembre de 2015.

[2] Blog panda security: El 91informáticos. http://www.pandasecurity.com/spain/mediacenter/notas-de-prensa/pymes-ataques-informaticos/, Online; visto el 1 de Diciembre de 2015.

[3] Google learning: Introduction to cross-site scripting. https://www.google.com/about/appsecurity/learning/xss/#WhatIsIt, Online; visto el 1 de Diciembre de 2015.

[4] Microsoft: Format string bugs. https://msdn.microsoft.com/es-es/es/library/ee823826(v=cs.20).aspx, Online; visto el 1 de Diciembre de 2015.

[5] Precios 0day elmundo. http://www.elmundo.es/tecnologia/2015/11/03/5638b6d1ca4741c3788b45af.html, Online; visto el 1 de Diciembre de 2015.

[6] Precios 0day ios9. https://www.zerodium.com/ios9.html, Online; visto el 1 de Diciembre de 2015.

[7] Ataque a la appstore. http://www.elmundo.es/tecnologia/2015/09/21/560025f1e2704e3c038b45b5.html, Online; visto el 2 de Diciembre de 2015.

[8] Ataque a sony pictures, bbc. http://www.bbc.com/mundo/ultimas_noticias/2014/12/141219_ultnot_corea_norte, Online; visto el 2 de Diciembre de 2015.

[9] Ataque a sony pictures, elpais. http://cultura.elpais.com/cultura/2014/12/18/actualidad/1418933088_481909.html, Online; visto el 2 de Diciembre de 2015.

[10] Cebr: 60firms from cyberattacks. http://www.cebr.com/reports/60-of-british-ctos-say-uk-government-is-performing-poorly-in-protecting-firms-from-cyberatta Online; visto el 2 de Diciembre de 2015.

[11] CnnexpanxiÓn: ¿quién causó realmente el ataque a sony? http://www.cnnexpansion.com/negocios/2014/12/24/quien-causo-realmente-el-ataque-a-sony, Online; visto el 2 de Diciembre de 2015.

[12] Comando nmap. http://linux.die.net/man/1/nmap, Online; visto el 2 de Diciembre de 2015.

[13] Coreimpact. http://www.coresecurity.com/products/core-impact/recent-exploits-and-updates, Online; visto el 2 de Diciembre de 2015.

[14] Coreimpact magazine. http://www.scmagazine.com/core-impact-professional/review/3791/, Online; visto el 2 de Diciembre de 2015.

[15] Cortafuegos. http://web.mit.edu/rhel-doc/3/rhel-sag-es-3/ch-basic-firewall.html, Online; visto el 2 de Diciembre de 2015.

[16] Manual de instalacion oficial metasploitable2. https://community.rapid7.com/message/4137#4137, Online; visto el 2 de Diciembre de 2015.

[17] Manual nmap. https://nmap.org/man/es/, Online; visto el 2 de Diciembre de 2015.

[18] Metasploit framework oficial. https://www.offensive-security.com/metasploit-unleashed/requirements/, Online; visto el 2 de Diciembre de 2015.

[19] Metasploit, universidad de vigo. http://ccia.ei.uvigo.es/dojo/pentest/dojo-pentest/index.html, Online; visto el 2 de Diciembre de 2015.

[20] msfconfig. https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/, Online; visto el 2 de Diciembre de 2015.

[21] Precios metasploit editions. https://community.rapid7.com/docs/DOC-2287, Online; visto el 2 de Diciembre de 2015.

[22] Sistemas de detección de errores, ids. http://www.adminso.es/index.php/4.2.2._%C2%BFQu%C3%A9_son_los_IDS%3F, Online; visto el 2 de Diciembre de 2015.

[23] Taller de metasploit. http://ucys.ugr.es/download/taller3/Taller3_Metasploit_Part1.pdf, Online; visto el 2 de Diciembre de 2015.

[24] Web oficial de canvas. http://www.immunityinc.com/products/canvas/, Online; visto el 2 de Diciembre de 2015.

[25] Acronis. http://www.acronis.com/es-es/, Online; visto el 3 de Diciembre de 2015.

[26] Aomei. http://www.backup-utility.com/free-backup-software.html, Online; visto el 3 de Diciembre de 2015.

[27] Eip. http://web-sisop.disca.upv.es/gii-dso/es/t2-arquitectura/tr6.html, Online; visto el 3 de Diciembre de 2015.

[28] Ejemplos de ids. http://www.adminso.es/index.php/4.2.6._HIDS_o_NIDS_usados_en_combinaci%C3%B3n, Online; visto el 3 de Diciembre de 2015.

[29] Ids host. http://www.adminso.es/index.php/4.2.5._IDS_basados_en_host_(HIDS), Online; visto el 3 de Diciembre de 2015.

[30] Ids red. http://www.adminso.es/index.php/4.2.4._IDS_basados_en_red_(NIDS), Online; visto el 3 de Diciembre de 2015.

[31] Instituto nacional de estadística, población que usa internet (en los últimos tres meses). http://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925528782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout&param3=1259924822888, Online; visto el 30 de Noviembre de 2015.

[32] Red hat enterprise linux 4, manual de seguridad. http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-es-4/s1-risk-serv.html, Online; visto el 30 de Noviembre de 2015.

[33] David Puente Castro. *Técnicas de explotación de vulnerabilidades en Linux para la creación de exploits*. 0xWord, 2013.

[34] Pablo González Pérez. *Metasploit para Pentesters*. 0xWord, 2014.

[35] Offensive Security. exploitdb. https://www.exploit-db.com/, Online; visto el 30 de Noviembre de 2015.

[36] Chris Eagle Jonathan Ness Michael Lester Shon Harris, Allen Harper. *Hacking Ético*. Anaya, 2005.