ATAQUES DE APLICACIÓN Y DE RED

Estas notas de apoyo tienen como objetivo suplementar las transparencias y facilitar la comprensión de los contenidos vistos en clase. Son, por tanto, un complemento y no un sustituto de estas.

1 ATAQUES A APLICACIONES WEB

Es necesario destacar (como hemos ido haciendo a lo largo de la asignatura) que proteger aplicaciones web requiere un enfoque distinto al tradicional basado en firewall y filtrado de red ya que, evidentemente, se debe permitir el tráfico HTTP para que la aplicación web funcione.

Por ello, es de suma importancia que la seguridad se considere como parte del diseño de la aplicación y no simplemente como un elemento externo. No obstante, existen elementos específicos para mejorar la seguridad de cualquier aplicación web como puede ser un Web Application Firewall (WAF) o servicios externos como redes de distribución de contenidos (CDN) que muchas veces incluyen protección contra ataques de denegación de servicio y otros.

1.1 CROSS SITE SCRIPTING (XSS)

Se debe tener claro que, si bien el ataque consiste en inyectar JavaScript en el servidor, ese mismo código JavaScript se ejecuta luego en el navegador de los visitantes a ese sitio web. Es por lo tanto una inyección en el servidor, pero con el objetivo de atacar al cliente.

Para que sea posible este ataque es necesario que el servidor acepte entradas sin validar (es decir que permita código JavaScript como parte de la entrada) y que luego utilice dicha entrada como parte de la web sin filtrarla.

En el ejemplo (transparencia 5), se muestra una aplicación de *bookmarks* que acepta el nombre del usuario (ABBY en este caso) y luego lo utiliza en el mensaje de agradecimiento sin filtrar. Si alguien introdujera código JavaScript en ese campo de nombre de usuario, podría lograr un ataque XSS.

Hay mucha información sobre este tipo de ataques en OWASP.

1.2 INYECCIÓN SQL

De forma similar a XSS, una inyección SQL consiste en utilizar entradas que luego se emplean como partes de una consulta para lograr ejecutar código SQL arbitrario en el servidor de bases de datos. En este caso, el ataque va dirigido al servidor y permite realizar acciones como obtener nombres de campos y tablas, modificar datos, obtener datos e, incluso, borrar la base de datos.

Entre otras cosas, una inyección SQL puede posibilitar salvar los controles de autentificación si las contraseñas se guardan en la base de datos: a lo mejor no podemos recuperar las contraseñas si se han almacenado convenientemente (PBKDF + sal) pero, en muchos casos, sí podemos sustituirlas por otras que conozcamos.

De nuevo, en OWASP hay información adicional sobre este tipo de ataque.

1.2.1 INYECCIÓN XML

El concepto de inyección XML (o inyección JSON) está íntimamente ligado al de inyección SQL. La idea es que parte de un documento XML o JSON (u otro lenguaje descriptivo similar) se utiliza como parte de una consulta y podría permitir realizar una inyección SQL.

Si bien es muy frecuente filtrar las entradas que provienen de formularios web, a veces, es menos evidente que la información proveniente de ficheros puede servir también como vector de ataque para inyecciones SQL y no se filtran convenientemente.

1.3 INYECCIÓN DE COMANDOS

Un ataque de inyección de comandos o de recorrido de directorios consiste en intentar escapar de los directorios preestablecidos para documentos web e intentar lanzar comandos o abrir ficheros que, en principio, deberían quedar fuera del alcance del servidor web.

Este ataque puede apoyarse en algún tipo de programa auxiliar (CGI o scripts) que se utilice para generar la respuesta web o, a veces, es simplemente un problema de configuración del servidor o sistema. Es un ataque muy común y hay bots en internet que prueban direcciones IP aleatorias e intentan este tipo de ataques de forma continua.

Para protegerse frente a este ataque, puede ser interesante considerar que el servidor web tenga el privilegio mínimo posible y limitar su acceso a otras partes del sistema. En plataformas BSD se puede utilizar el comando *jail* para restringir el acceso de un servidor; y, de forma más general, se puede utilizar *chroot* en sistemas Unix.

2 ATAQUES EN EL CLIENTE

En este tipo de ataques, el objetivo es explotar vulnerabilidades en las aplicaciones del cliente (por ejemplo, el navegador). Se puede basar en la interacción con un servidor comprometido o, incluso, al procesar datos maliciosos (como una imagen con la cabecera manipulada).

Tradicionalmente, el foco de atención a la hora de proteger sistemas siempre ha estado puesto en el servidor, lo que ha facilitado en cierta medida los ataques al cliente. En algunos casos, estos ataques no requieren nada más que la visita a un servidor comprometido (*drive-by download* o descarga por visita).

Describimos a continuación algunos ataques comunes.

2.1 MANIPULACIÓN DE CABECERAS

El protocolo HTTP tiene muchos campos de cabecera.

En concreto, el campo *referer* sirve para que una web pueda establecer estadísticas de cuáles son los orígenes más frecuentes para sus visitas (determinar si provienen de un buscador, de otra web popular, etc.). Algunas veces, este campo se emplea como parte de la validación de la autentificación en una especie de sesión: por ejemplo, si el *referer* de la visita es la propia web, se relajan las restricciones de seguridad. Esto es un grave error, ya que cualquier atacante puede cambiar el valor de este campo al realizar las peticiones HTTP. No se debe utilizar para ningún aspecto relacionado con la seguridad.

Otro campo problemático es el *Accept-Language* que permite establecer una serie de preferencias en cuanto al idioma de la página web. Este campo se puede emplear en una consulta a la base de datos (en función del idioma, obtener un contenido distinto) lo que podría llevar a una inyección SQL que no se había considerado en principio. También puede llevar a una inyección de comandos o recorrido de

directorios si este campo sirve para establecer el nombre del fichero a leer por el servidor en función del idioma.

2.2 COOKIES

Como HTTP es un protocolo que no tiene estado, no permite controlar las sesiones en sí. Para ello, surgen las *cookies (galletas)* como un mecanismo para almacenar cierta información del usuario en el cliente.

Hay diferentes tipos de *cookies*:

- De *primera mano*. Son aquellas que provienen del sitio (o dominio) principal que se está visitando.
- De *terceros*. Son aquellas que provienen de dominios o entidades distintas del sitio principal que se está visitando. Esto ocurre mediante contenido accesorio, por ejemplo, un anuncio o un botón asociado a una red social, que en realidad se sirve desde un dominio distinto de la página principal. Este tipo de *cookies* permiten establecer perfiles y, en cierta medida, controlar qué sitios visitamos.
- De *sesión*. No son persistentes, se almacenan en memoria RAM exclusivamente y desaparecen al cerrar el navegador.
- *Persistente*. Se almacenan en disco y sobreviven entre distintas sesiones de navegación.
- *Segura*. Son aquellas que se transmiten mediante un canal seguro (generalmente HTTPS) y son esenciales para gestionar las sesiones de forma segura.
- *Flash*. En lugar de estar asociadas al protocolo HTTP, las *cookies* flash están asociadas a los complementos desarrollados con la tecnología Adobe Flash. Esto les permite almacenar mucha más información, sobrevivir al borrado de *cookies* del navegador, etc. En algunos casos se pueden emplear para restaurar *cookies* que habían sido borradas o bloqueadas.

Si bien las *cookies* tienen usos perfectamente legítimos, a veces pueden presentar problemas de privacidad y seguridad. Por ejemplo, las *cookies* de primera mano pueden ser adivinadas o robadas para hacerse pasar por el usuario (asalto de sesión); mientras que las *cookies* de terceros pueden servir para monitorizar el historial de navegación (para evitarlo puede ser útil emplear extensiones orientadas a la privacidad como *disconnect* o *adblockers*, etc.).

2.3 ASALTO DE SESIÓN

Consiste en el robo del token o identificador de sesión, permitiendo suplantar al usuario frente al servidor. Este ataque puede consistir en diversas técnicas:

- Capturar el tráfico si no está cifrado (por ej. cuando se está conectado a una red inalámbrica en una cafetería o similar). Por esto es tan importante habilitar conexiones cifradas incluso si el servicio no contiene información sensible.
- Adivinar dicho token en caso de que se emplee un generador aleatorio inseguro. Es muy importante cerciorarse de que se generan números aleatorios impredecibles para su uso en aplicaciones de seguridad.
- Utilizar otro tipo de ataque, como XSS, para obtener ese token de sesión.

2.4 EXTENSIONES MALICIOSAS

Si bien las extensiones de los navegadores se han convertido en algo prácticamente esencial, dada la funcionalidad que ofrecen, se debe tener en cuenta que pueden representar riesgos considerables de seguridad.

En el caso de las extensiones de navegadores como Chrome o Firefox, es importante entender que estas extensiones tienen acceso a todo el tráfico descifrado (tras el túnel HTTPS, si no, no podrían funcionar) del navegador. Por lo que, cuando instalamos una extensión, debemos tener la confianza de que no se trata de malware; esto resulta muy complicado, especialmente si hay algún sistema de actualización automática sin revisión previa.

Otro tipo de extensiones, son aquellos complementos orientados a ejecutar código nativo para extender la funcionalidad del navegador. Tecnologías antiguas como ActiveX de Microsoft se ejecutaban con un nivel de privilegio muy elevado, siendo la firma digital la única garantía de seguridad; esto ha provocado numerosos problemas de seguridad a lo largo de la historia. Otro enfoque más actual, sería Native Client (NaCl de Google Chrome) que permite la ejecución de código nativo, pero dentro de una máquina virtual a modo de *sandbox* (contenedor seguro) y con un nivel de privilegios mínimo.

2.5 DESBORDAMIENTO (BUFFER OVERFLOW)

Los desbordamientos de búfer son problemas asociados, comúnmente, a los lenguajes de bajo nivel o sin gestión segura de memoria. Estrictamente hablando, se produce un desbordamiento de búfer cuando un programa permite que datos de entrada se escriban más allá del final del búfer asignado, pero hay otros problemas que pueden tener consecuencias similares.

Estas consecuencias pueden ser de un espectro muy amplio, desde un bloqueo hasta que el atacante obtenga el control completo de la aplicación, y si la aplicación se ejecuta como un usuario de alto nivel, el control de todo el sistema operativo.

Aunque se podría pensar que únicamente los programadores descuidados son víctimas de desbordamiento de búfer, el problema es bastante complejo y cualquiera que haya escrito suficiente código C/C++ ha cometido, con toda probabilidad, este tipo error.

Existen algunas medidas que se pueden tomar para minimizar este tipo de problemas:

- Protección de segmentos de memoria
- Aleatorización de direcciones
- Uso de lenguajes con gestión de memoria segura (como Go, Java y otros).

3 ATAQUES DE RED

Si bien la mayoría de los ataques actuales van dirigidos a la aplicación, siguen existiendo muchos ataques a la capa de red que deben ser considerados en el diseño y desarrollo de cualquier sistema seguro.

3.1 DENEGACIÓN DE SERVICIO

Aunque habitualmente entendemos la denegación de servicio como un ataque dirigido a sobrecargar un servidor, en realidad puede consistir en la extinción de cualquier tipo de recurso. Simplemente hace falta que el atacante pueda agotar dicho recurso antes de que se pueda reponer.

El caso más común es la denegación de servicio distribuida (o DDoS) que permite a un grupo o individuo realizar ataques masivos de forma sencilla. Los objetivos más comunes son sobrecargar servidores,

sobrecargar conexiones de red, provocar fallos o interrupciones en servidores o atacar una dependencia. Este último caso puede consistir en una vía sencilla cuando se busca atacar un objetivo muy potente: por ejemplo, tal vez no se pueda atacar a un proveedor internacional de productos (recursos ingentes), pero sí a las mensajerías que los distribuyen al usuario final (menor nivel de recursos y protección) provocando un daño similar.

Para una DDoS, es necesario reclutar máquinas que ejecuten el ataque. Aquí es muy habitual o bien hacer un reclutamiento manual o utilizar técnicas automatizadas asociadas a una botnet (ver tema de malware). En muchos casos, el control de estos ataques se hace mediante canales públicos (redes sociales o protocolos populares) para evitar su detección. Cabe destacar que existen herramientas que ya realizan todos estos procesos sin necesidad de conocimientos.

Otros tipos de ataques DoS más tradicionales consisten en utilizar algún mecanismo que permita amplificar la capacidad del atacante frente al servidor.

Es importante entender que es imposible prevenir un ataque de denegación de servicio de forma completa. Las estrategias más frecuentes están orientadas a sobrevivir dicho ataque. Algunos factores a tener en cuenta:

- Guardar logs para tener capacidad de trazabilidad frente a un ataque y monitorizarlos en busca de actividad sospechosa (por ejemplo, mediante un IDS).
- Conocer las novedades respecto a ataques DDoS y defensas relacionadas (por ejemplo, CDN).
- Establecer rutinas de actualización, escaneo y monitorización, comprobando de forma regular que las máquinas no pertenezcan a una botnet o que nuestros servidores no sean vulnerables por otros motivos.

Entre muchas estrategias, una de las más comunes es el uso de computación en la nube para intentar escalar más rápido que el ataque. Si bien es una opción válida en principio, se debe considerar que muchos sistemas cloud no son capaces de escalar suficientemente rápido para resistir un ataque masivo; además, en muchos casos, la escalabilidad conlleva un incremento del coste asociado a la infraestructura muy considerable.

3.2 INTERCEPCIÓN

Un ataque de intercepción es similar a un ataque de hombre-en-el-medio criptográfico, pero aplicado al tráfico de red. Un atacante se puede intercalar entre dos extremos de comunicación y capturar la información (ataque pasivo) o modificar la información antes de ser retransmitida (ataque activo).

Además, se puede realizar un ataque de reproducción en el que se reproduce con posterioridad la información almacenada para lograr ciertos objetivos. Por ello, es muy importante que los protocolos de seguridad tengan algún tipo de gestión del tiempo (con *timestamps* o similar) para evitar este tipo de ataques de reproducción.

3.3 ENVENENAMIENTO ARP

Al igual que el protocolo DNS convierte nombres comunes a direcciones IP, el protocolo ARP trabaja a nivel de red local convirtiendo direcciones IP a direcciones MAC de Ethernet. Es un protocolo muy antiguo que no tienen provisión de seguridad (sin autentificación), por lo que permite manipular la identidad de ciertos nodos de la red posibilitando ataques como robo de datos, intercepción, denegación de acceso o servicio, etc.

3.4 ENVENENAMIENTO DNS

El envenenamiento DNS surge al tampoco proveer un mecanismo de autentificación seguro. En el diagrama de la transparencia 23 podemos ver el siguiente flujo:

1. El atacante realiza una petición DNS a un servidor válido respecto a un dominio malicioso que controla.
2. Este servidor solicita la información acerca de ese domino al servidor DNS asociado.
3. Este servidor, además de proporcionarle la información sobre ese dominio, le da información adicional errónea sobre otros dominios distintos.
4. Cuando un usuario válido consulta sobre estos dominios al servidor DNS válido recibe la información errónea proporcionada por el servidor DNS maligno.

De esta forma se puede redirigir tráfico a una web de phishing, hacer intercepción, etc.

4 AMPLIACIÓN

Existen muchos materiales en O'Reilly Safari (acceso gratuito con la cuenta de la UA) para ampliar, entre otros:

- Libro "Attacking Network Protocols"
- Libro "24 Deadly Sins of Software Security"
- Topic "ComptTIA Security+"