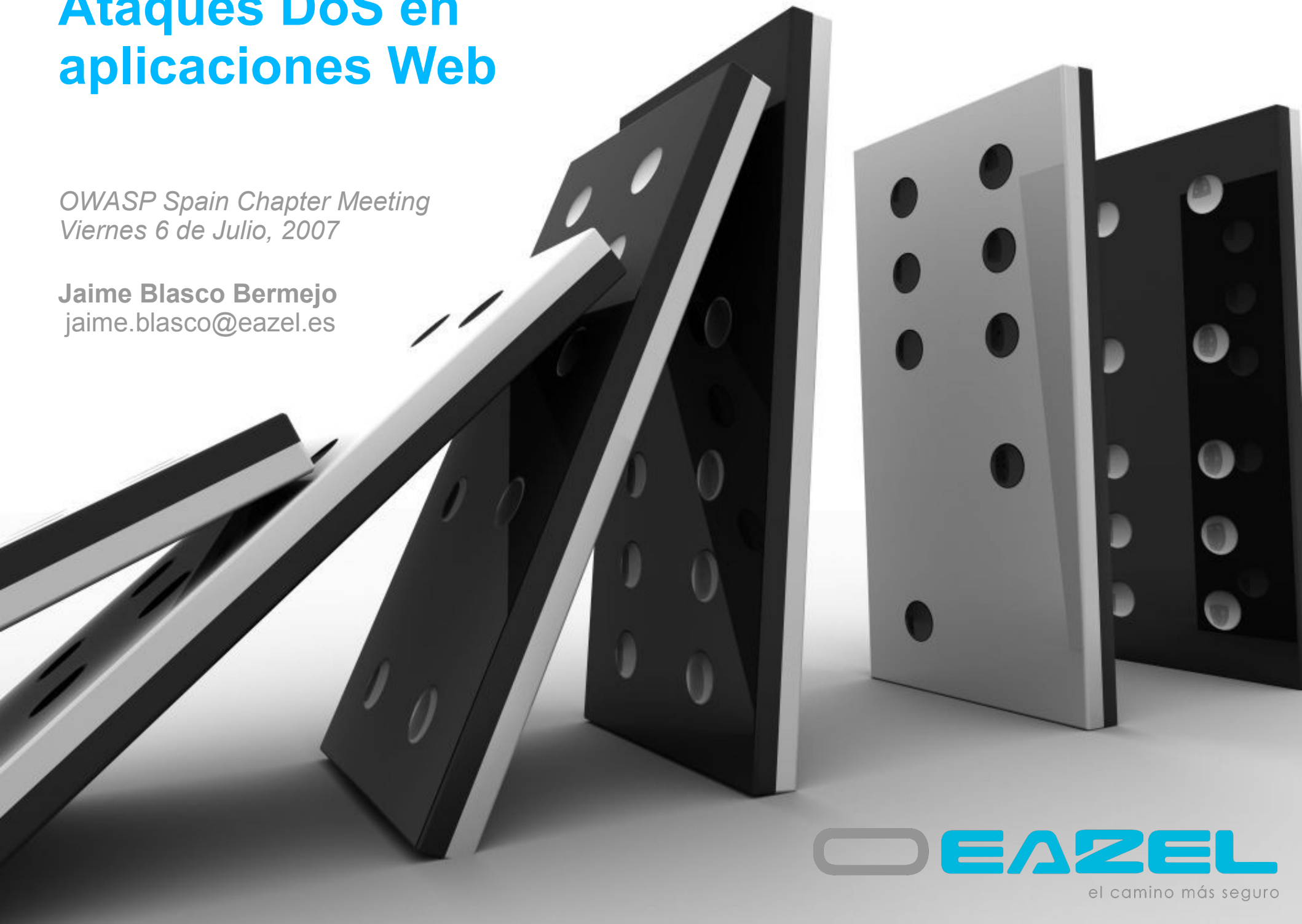
Ataques DoS en aplicaciones Web

OWASP Spain Chapter Meeting
Viernes 6 de Julio, 2007

Jaime Blasco Bermejo
jaime.blasco@eazel.es



EAZEL
el camino más seguro

Contenido

- Introducción a los ataques DoS y DDoS
- Ataques DoS contra aplicaciones Web
- Ejemplos
- Medidas de protección

Introducción a los ataques DoS

- Los ataques de **Denegación de Servicio (DoS)** tienen la finalidad de provocar que un servicio o recurso sea innacesible para los usuarios legítimos.

- Este tipo de ataques pueden provocar:
  - Parada de todos los servicios de una máquina
  - La máquina sólo puede dar determinados servicios
  - La máquina no puede dar servicio a determinados usuarios

Modos de ataque

Los ataques DoS se pueden llevar a cabo de diferentes formas y cubren infinidad de servicios. Existen tres tipos básicos de ataque:

- Consumo de recursos limitados.
- Destrucción o alteración de datos.
- Destrucción o alteración física de componentes de la red.

Atacantes

Algunos de los grupos que pueden llevar a cabo este tipo de ataques son:

- Script Kiddies
- Competencia
- Militares
- Empleados incompetentes

Introducción a los ataques DoS

Ejemplos de ataques DoS

- Consumo de ancho de banda:
  - Smurf Attack
  - ICMP Ping Flood
  - Fraggle Attack
- Ataques a la conectividad
  - SYN Flood Attacks

Introducción a los ataques DoS

Consumo de Ancho de Banda

- **Smurf Attack** : Este ataque se basa en mandar un gran número de peticiones echo (ICMP) a direcciones de Broadcast usando una IP de origen falsa. Esto provoca que la IP de origen sea inundada con multitud de respuestas.

- **ICMP Ping Flood:** En este ataque se inunda a la víctima con paquetes ICMP Echo Request.

- **Fraggle Attack:** Es similar al ataque Smurf pero en este caso se envía tráfico UDP en lugar de ICMP.

Introducción a los ataques DoS

Ataques a la conectividad

- **SYN Flood Attack:** Consiste en enviar muchos paquetes TCP/SYN con la dirección de origen falseada. Esto provoca que el servidor espere las respuestas que nunca llegan, provocando un consumo elevado de recursos que afectan al rendimiento del servidor.

Introducción a los ataques DDoS

- Un ataque de **Denegación de Servicio Distribuido (DdoS)** es un tipo especial de ataque DoS en el que se utilizan varios equipos para realizar un ataque coordinado contra una máquina.

- En este tipo de ataque se suelen utilizar máquinas denominadas *Zombies* que el atacante consigue controlar gracias a algún tipo de malware.

- Al conjunto de máquinas *Zombies* que controla un atacante se las suele denominar BotNets.

Introducción a los ataques DDoS

Existen multitud de herramientas de DdoS conocidas, algunas de las más importantes son:

- **Trinoo**: Primera herramienta conocida de este tipo.
- **TFN y TFN2K:** ICMP Flood, SYN Flood, UDP Flood.
- **Stacheldraht:** ICMP Flood, SYN Flood, UDP Flood y Smurf.
- **Shaft:** SYN flooding, UDP flooding, ICMP flooding, and Smurf

Introducción a los ataques DDoS

Ejemplos de ataques DdoS reales:

- En **Febrero de 2000** ualgunas compañías incluyendo Yahoo, Amazon y Ebay sufrieron una serie de ataques DdoS que les dejaron sin servicio.
- En **Julio de 2001** una serie de ataques DdoS tuvieron efecto sobre la web de la Casa Blanca.
- En **Octubre de 2002** se realizó un ataque DdoS contra los servidores DNS raíz quelograron afectar a 9 de 13 de los servidores DNS que controlan Internet.
- En **Mayo de 2007** Estonia sufrió un ataque DdoS masivo que afectó a multitud de servidores incluyendo la presidencia, el parlamento, varios ministerios, bancos, telecomunicaciones, etc.

Ataques DoS contra aplicaciones web

Ataques DoS contra aplicaciones web

- Este tipo de ataque tiene como objetivo dejar sin servicio a la propia aplicación en lugar de la máquina.

- Una de las ventajas que tiene el atacante es que no necesita tantos recursos para llevar a cabo el ataque como un ataque DoS normal.

- En muchas ocasiones este tipo de ataques son más difíciles de detectar debido a que se camuflan en peticiones aparentemente comunes.

Ataques DoS contra aplicaciones web

Los ataques DoS contra aplicaciones web se pueden llevar a cabo de diferentes maneras:

- **Cuelgue de la Aplicación:**
  - Buffer Overflows
  - Varios
- **Modificación y destrucción de datos**
- **Consumo de recursos**
  - CPU
  - Ancho de Banda
  - Memoria
  - Espacio en Disco

Cuelgue de la Aplicación: Buffer Overflow

- Los fallos de desbordamiento de Buffer pueden estar presentes si escribimos aplicaciones Web en C o lenguajes similares. Este tipo de ataques se producen cuando se escriben datos en un buffer que sobreescriben otros adyacentes, se suelen producir al copiar cadenas de caracteres de un buffer a otro.

- Estos desbordamientos de buffer pueden provocar muchas veces ataques de denegación de servicio contra la aplicación.

    - Hay que prestar especial atención a funciones como strcpy(), strcat(), sprintf(), gets()..

Cuelgue de la Aplicación: Varios

- Los fallos de Format String se pueden utilizar en ocasiones para realizar ataques de denegación de servicio.

- Muchas veces un atacante puede aprovechar un ataque de SQL injection, Remote File Inclusion, etc si los permisos del servidor no están debidamente configurados para llamar a comandos del sistema que paren las aplicaciones o servicios.

Modificación y destrucción de datos

Este tipo de ataques no provocan que el servidor deje de dar un servicio, no obstante estos ataques afectan al funcionamiento normal del servicio provocando un mal funcionamiento del mismo.

Modificación y destrucción de datos: Ejemplo 1

En el primer ejemplo el atacante aprovecha un fallo de SQL Injection para borrar o modificar tablas de la base de datos.

Imaginemos una aplicación .NET en la que los usuarios sin identificarse pueden listar noticias y los usuarios registrados pueden realizar más acciones.

Imaginemos el código de http://servidor/noticias.aspx?id=1

Ataques DoS contra aplicaciones web

```
private void Page_Load(object sender, System.EventArgs e) {
        string id = "";
        id = Request.QueryString["id"].ToString();
        string conexion = "Data Source=localhost;User id=sa;Password=sa;Initial
        Catalog=aplic";
        SqlConnection conector = new SqlConnection (conexion);
        SqlDataAdapter adapt = new SqlDataAdapter("select * from contactos where id=" + id,
        conector);
        DataSet datos = new DataSet();
        adapt.Fill(datos);
        DataGrid1.DataSource = datos;
        DataGrid1.DataBind();
    }
```

Si el atacante hace una petición a:
http://servidor/noticias.aspx?id=1 union DROP TABLE usuarios
y los permisos de la base de datos son incorrectos, el atacante será capaz de borrar la tabla de usuarios con las consecuencias que esto provoca en el funcionamiento normal de la aplicación.

Modificación y destrucción de datos: Ejemplo 2

Imaginemos una aplicación que usa un sistema de autentificación en el cual la cuenta del usuario queda bloqueada tras ciertos intentos cuando se introducen contraseñas inválidas para evitar ataques de fuerza bruta contra las cuentas de usuario.

Un atacante podría conseguir una lista de usuarios y construir un script que bloquease multitud de cuentas de usuario provocando que el servicio no funcionase correctamente.

Modificación y destrucción de datos: Ejemplo 3

En este ejemplo vamos a utilizar una aplicación en en la que se utilizan consultas XPATH para la autentificación de los usuarios, cuyos datos están guardados en un fichero XML. Cuando se da de alta un usuario no se comprueban las entradas y los datos de usuario se guardan en el fichero XML. Si al darse de alta un usuario introduce en su datos el caracter & ó < provocará que el fichero XML sea corrupto y cuando los usuario intenten entrar en el sistema y la aplicación intente leer el fichero XML dará una excepción debido a que el fichero es corrupto.
Unhandled Exception: System.Xml.XmlException: a name did not start with a legal character 60 (<)

Consumo de recursos : CPU : Ejemplo1

En este ejemplo vamos a realizar un ataque de SQL Injection ejecutando una consulta que consume muchos recursos.

```
$id = $_GET['id'];
mysql_select_db("bbdd", $link);
$sql = "select * from news where auction_id=".$id;
$res = mysql_query($sql);
```

El atacante puede acceder a esta URL:
http://server/inject.php?id=1 **union select benchmark(50000000, sha1(1)),1,1,1,1,1;**

Consumo de recursos : CPU : Ejemplo1

La función benchmark(count, expr) de MySQL ejecuta la expresión expr un número count de veces..
Esto permite al atacante meter al servidor en un bucle que consume mucha CPU sin apenas invertir recursos para ello.

Veamos la salida del comando top durante la ejecución de esta consulta:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
13981	mysql	21	0	370m	6944	3272	S	**99.4**	1.4	9:29.83	**mysqld**

Consumo de recursos : CPU : Ejemplo2

En el siguiente ejemplo el atacante utiliza un ataque Regular Expression Injection para ejecutar una bomba fork.
El siguiente código se utiliza para cambiar el password de un usuario:

```
$username = $_GET['username'];
$password = $_GET['password'];
$newpassword = $_GET['newpassword'];
$users = file_get_contents('users.xml');
echo $newpassword;
$users = preg_replace(
"#(<username\s*>\s*$username\s*</username>\s*
<password\s*>)$password</password>#e",
"'\\1'.strtolower('$newpassword').'</password>'",
$users
);
file_put_contents('users.xml', $users);
?>
```

Consumo de recursos : CPU : Ejemplo2

Como nos cuenta Christian Wenz en su artículo, si pasamos la siguiente cadena como variable newpassword:

```
'.system('comando').'
```

La cadena de reemplazo se transforma en:

```
"'\\1'.strtolower(".system('comando').").'</password>'"
```

Con lo cual hemos inyectado la llamada a system.

Por ejemplo un atacante podría ejecutar una bomba fork:

```
:(){ :|:& };:
ó
perl -e "fork while fork" &
```

Esto saturará el espacio en la lista de procesos del sistema.

Consumo de recursos : CPU : WebServices

En el siguiente ejemplo, el ataque se realiza contra un Web Service.

- Existen dos tipos fundamentales de parsers utilizados en los Web Services, SAX y DOM.

- Los parsers basados en DOM cargan el XML entero en memoria.

- Uno de los ataques que se podrían llevar a cabo sería mandar como entrada un fichero XML muy grande lo que provocaría un consumo de recursos en el servidor durante el parseo.

Consumo de recursos : CPU : WebServices

- Otro ataque que se podría llevar a cabo sería enviar un fichero XML usando elementos recursivos que provocarían que el servidor tuviese que utilizar más recursos todavía.

- En el caso de los parsers basados en SAX este tipo de ataques no se pueden llevar a cabo ya que parsean el XML según lo necesiten, es decir sólo almacenan en memoria los datos que necesarios.

Consumo de recursos : Espacio en Disco

• Imaginemos una aplicación interna en la que los trabajadores pueden subir sus ficheros y gestionarlos vía web. Un intruso accede a una de las cuentas y averigua que no se lleva a cabo una gestión del espacio ocupado por los usuarios. El intruso podría hacerse un pequeño Script que tras subir un archivo grande mandase peticiones al servidor para que copiase ese mismo fichero con otro nombre de tal manera que utilizando pocos recursos llenase el servidor rápidamente.

• Otro ejemplo de este tipo sería una aplicación en la que se guardan los logs detallados de algunas acciones. Si un atacante se da cuenta podría provocar repetidamente estas acciones para que se guardasen muchos logs llenando así los discos del servidor.

Consumo de recursos : Varios

- Otro ejemplo sería una aplicación que usa una base de datos muy grande. El atacante podría buscar expresiones regulares sofisticadas que consuman muchos recursos cada vez que se ejecuta la búsqueda y repetir la búsqueda continuamente.

- Un usuario malicioso puede llevar a cabo un ataque de Session ID Exhaustion que consiste en generar en un breve período de tiempo multitud de ID´s de sesión que provocará un consumo de recursos en el servidor pudiendo dejar fuera de servicio a otros usuarios que quisiesen acceder al mismo.

Denegación de servicio distribuida contra aplicaciones web

- Un atacante también puede llevar a cabo un ataque DdoS contra aplicaciones web, de hecho en muchos casos necesita menos máquinas para conseguir el objetivo que en un DoS común.

- Este ataque se puede llevar a cabo cuando a través de un DoS se consigue consumir muchos recursos del servidor con una conexión pero no los suficientes como para parar el servicio, de esta manera se puede realizar este ataque entre varias máquinas.

Nuevas herramientas para los malos

• En los últimos años han surgido nuevos conceptos en Internet que permiten el desarrollo de nuevas aplicaciones, uno de ellos es SecondLife®.

• En SecondLife existen objetos, estos objetos se pueden programar utiliza un lenguaje de scripting denominado LSL (Linden Scripting Language).

• LSL tiene algunas funciones que permiten realizar consultas a servidores como es llHTTPRequest y funciones para XML-RPC.

Nuevas herramientas para los malos : SecondLife

- La ventaja que puede encontrar un atacante en la función llHTTPRequest es que la petición la llevan a cabo los servidores de Linden Labs.

- Un atacante puede crear multitud de objetos que se metan en bucles y hagan peticiones a un determinado host, incluso se pueden crear BOTS dentro de SecondLife para que realizen esta tarea.

- Gracias a esto, un atacante se puede apoyar en SecondLife para realizar un ataque DoS ó complementarlo.

Nuevas herramientas para los malos : SecondLife

Veamos un ejemplo de un objeto que al ser creado realiza 20 peticiones GET a un host determinado:

```
key http_request_id;
default {
    state_entry() {
        integer a = 0;
        integer b = 20;
        for(; a < b; ++a) {
        http_request_id = llHTTPRequest("http://host/a.php", [], "");
        }
     }
      http_response(key request_id, integer status, list metadata, string body) {
    }
}
```

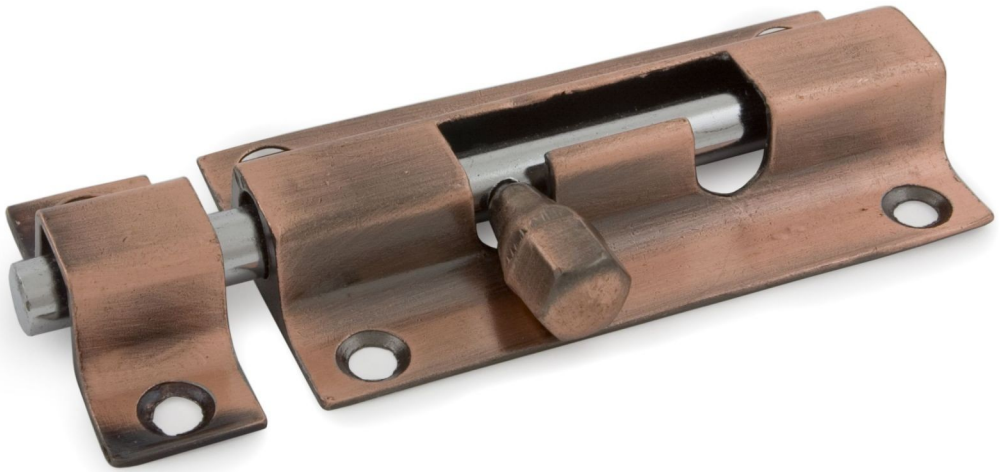
Nuevas herramientas para los malos : SecondLife

Observemos una de las peticiones recibidas:

```
69.25.104.XX - - [05/Jul/2007:21:01:53 +0200] "GET /XXX
HTTP/1.0" 200 28 "-" "Second Life LSL/1.17.2(0)
(http://secondlife.com)" "2089:468d4021:58" 0 eazel.es
```

Como vemos en el header “Second Life LSL/1.17.2(0)” es el cliente, si resolvemos la ip 69.25.104.XX veremos que es algo parecido a simXXX.agni.lindenlab.com.

Medidas de protección



Medidas de protección

- A nivel de código:
  - La regla básica es **VALIDAR LAS ENTRADAS!!!**.
  - Intentar evitar código que requiera muchas operaciones o un consumo excesivo de la CPU.
  - Comprobar el rendimiento de las funciones e intentar optimizarlas lo máximo posible.

Medidas de protección

- A nivel de red:
  - Implementar soluciones de balanceo de carga y cache si fuese necesario.
  - Implementar un firewall de aplicación como ModSecurity que ayuda a protegernos de ataques contra las aplicaciones que se nos hayan podido pasar al revisar el código.
  - Estudiar el retorno de inversión de un sistema comercial de protección contra este tipo de ataques e incluirlo en la red si es necesario.

Referencias

- **Denial of Service Attack Resource Page**
  http://www.denialinfo.com
- **Regular Expression Injection**
  http://hauser-wenz.de/playground/papers/RegExInjection.pdf
- **Regular Expression Denial of Service**
  http://www.cs.rice.edu/~scrosby/hash/slides/USENIX-RegexpW
- **Threat Classification - Web Application Security Consortium**
  http://www.webappsec.org/projects/threat/
- **XML Path Language (XPath)**
  http://www.w3.org/TR/xpath
- **LSL Portal - Second Life Wiki**
  http://wiki.secondlife.com/wiki/LSL_Portal

¿Preguntas?, ¿Comentarios?

¡GRACIAS!