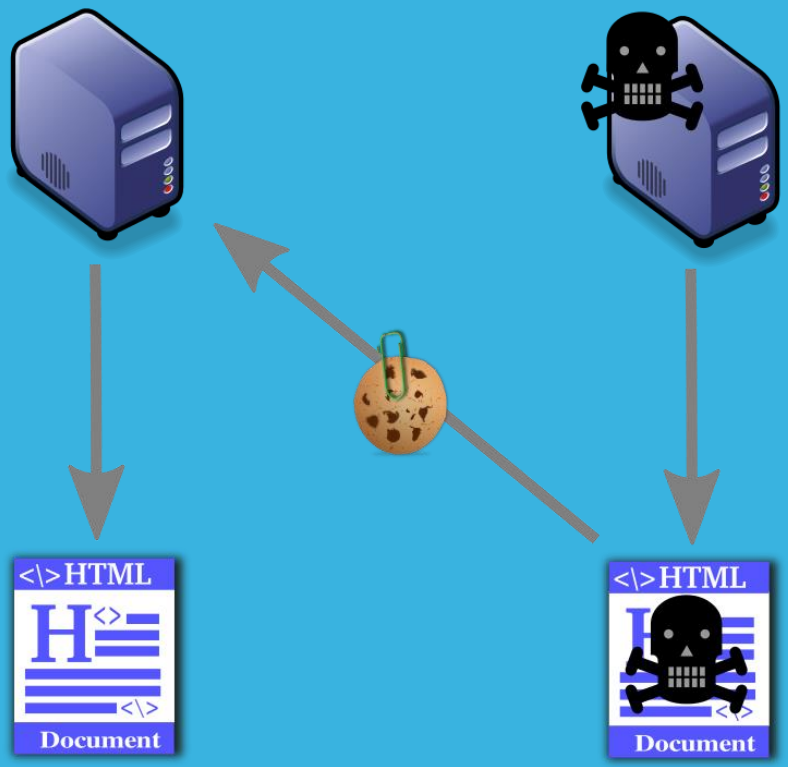
AUDITANDO CSRF

CARLOS GARCÍA GARCÍA



Hack & Beers Córdoba, Julio 2014

CONTENIDO

1. Sobre mi
2. Introducción
3. Auditando un caso práctico
4. Ejemplos reales
5. Detectar y prevenir CSRF

@ciyinet

SOBRE MI

Ingeniero en Informática (UCO)

Técnico en Seguridad de Redes y Sistemas

2010 – 2012

Consultor Seguridad Informática

Proxy Servicios y Consulting S.L.U. (Córdoba, España)

2013

Network Security Specialist

IBM (Brno, República Checa)

@ciyinet

SOBRE MI

2014

Security Architect – Pen Tester

Accenture (Praga, República Checa)

- **Pen Tests:**
  - Aplicaciones web
  - Aplicaciones móviles (iOS y Android)
  - SAP
  - Redes
- **Vulnerability Assessment**
- **Revisión de Código**
- **Revisión seguridad Arquitecturas de Red**

@ciyinet

INTRODUCCIÓN

- Cross Site Request Forgery (CSRF)
- Ataque que fuerza a un usuario a ejecutar acciones no deseadas en una aplicación web en la que el usuario está actualmente logueado.
- Atacante envía un enlace o lo publica y la víctima accede a él.
- Por lo general, con cada petición el **navegador incluirá automática la sesión del usuario**, HTTP basic auth crendentials, dirección IP, etc.
- Por lo tanto, la aplicación web no sabrá si la petición proviene o no de un usuario legítimo.

INTRODUCCIÓN

El alcance de CSRF **depende del rol de la víctima**:

- **Usuario normal:** datos del usuario y sus operaciones pueden ser comprometidas.
- **Administrador:** la web al completo podría ser comprometida.

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific

INTRODUCCIÓN

Ejemplo de escenario de ataque:

La aplicación vulnerable permite al usuario realizar una transferencia bancaria **sin incluir nada secreto y único** en la petición. Por ejemplo:

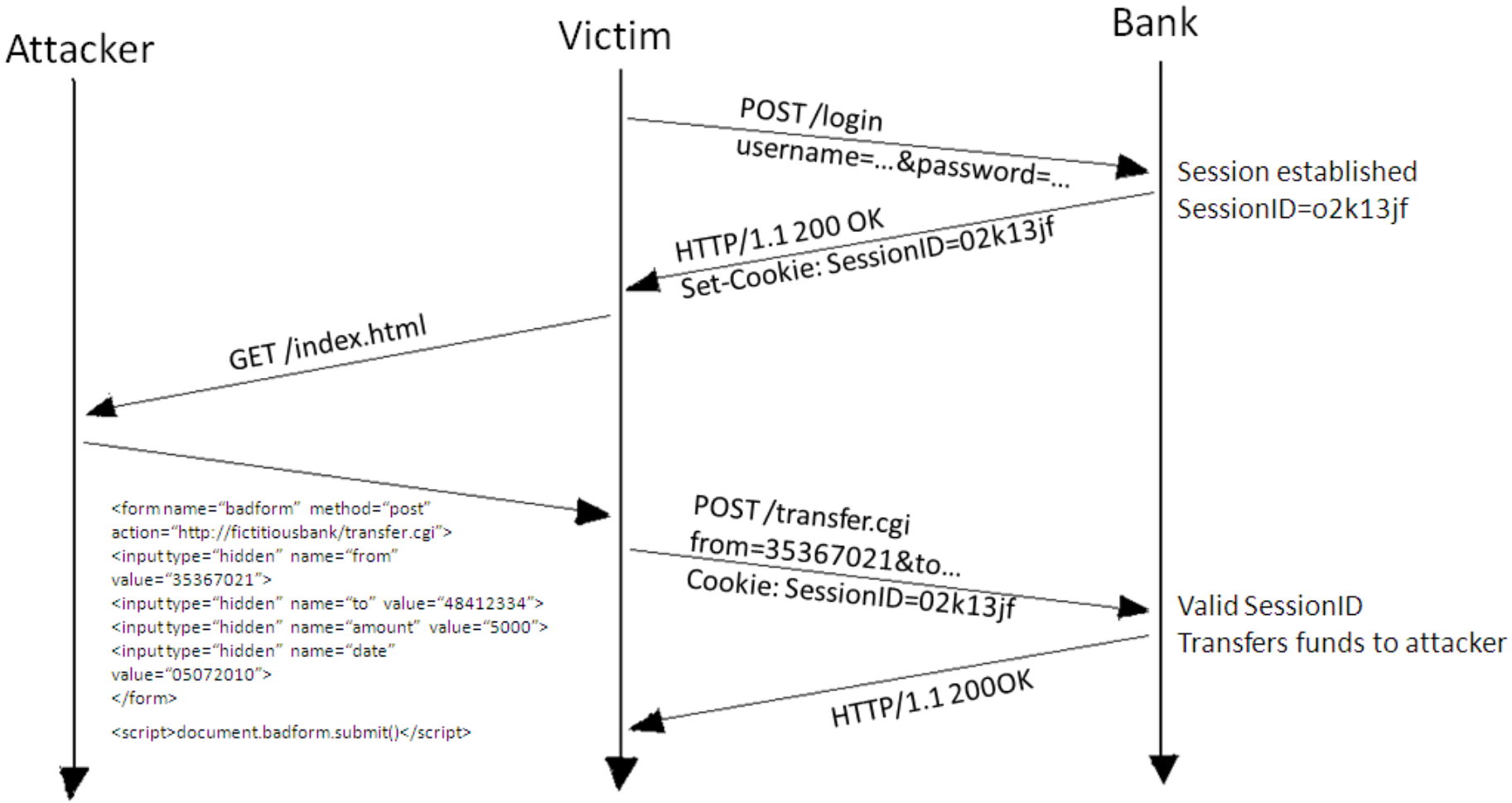
```
http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243
```

El atacante construye un exploit para realizar una petición que transfiera el dinero desde la cuenta de la víctima a la cuenta del atacante. Para ello:

```
<img src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#" width="0" height="0" />
```

Si la víctima visita la web estando logueado en la web vulnerable, la petición se realizará utilizando la sesión del usuario víctima y por lo tanto realizando la transferencia de dinero.

INTRODUCCIÓN



AUDITANDO UN CASO PRÁCTICO

Aplicación de chat escrita en PHP

1. Navegar aplicación usando Burp Suite
2. Encontrar puntos vulnerables a CSRF
3. Construir exploit
4. Ejecutar exploit
5. Comprobar éxito del ataque

EJEMPLOS REALES

Google Fixes Gmail Cross-site Request Forgery Vulnerability

Google has fixed a vulnerability in their Gmail web based email service which would have allowed internet attackers to steal mail messages from users without being noticed.

The attack works by forcing a logged-in user to add a mail filter to their Gmail account, thereby allowing their mail to be forwarded to an external mail address controlled by the attacker. Because the Gmail service did not adequately verify the origin of such requests, it was possible for attackers to create their own web pages that used JavaScript to automatically make such requests on behalf of their victims. In essence, a Gmail user would visit one of these pages and have their account compromised without necessarily realising anything is awry. Only close inspection of the Filters tab in the Gmail Settings menu would reveal what had happened.

Proof of concept exploits used JavaScript to make a silent POST request to the Gmail service and add the attacker's filter. With the results of the request hidden in an iframe, it is highly unlikely that a victim will have noticed that their Gmail account would have been compromised, particularly while they are browsing a completely different website. While this attack scenario would only be successful if the victim was logged in, many Gmail users remain constantly logged in throughout the day, thus increasing the likelihood of a successful attack.

The technique used by this exploit is known as CSRF (Cross-site Request Forgery) and is becoming an increasingly common method to attack web applications. If a web application is vulnerable to CSRF, it will allow unauthorised attackers to carry out arbitrary actions in the context of an authorised, logged in user of the application. Not only does this make a hacker's life easier, but it also helps them to cover up their tracks, as malicious actions will appear to be carried out, unwittingly, by authorised users of the system.

EJEMPLOS REALES

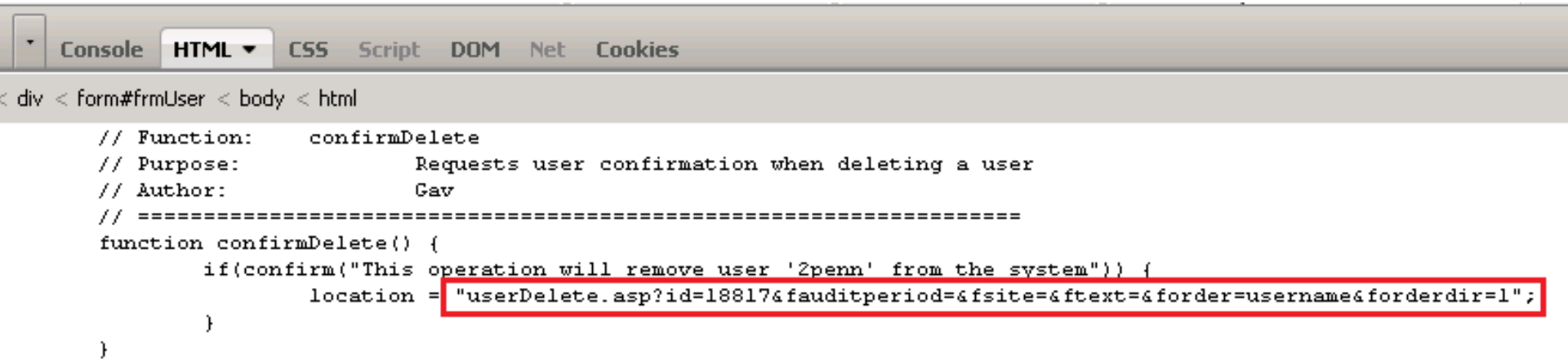
*index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<title>CSRF PoC</title>
<head>
</head>
<body onload="document.createElement('form').submit.call(document.getElementById('myForm'))">
    <form id="myForm" name="myForm" enctype="application/json;"
        action="https://[redacted].com/api/users/resetpassword" method="POST">

        <input id="json" name="source" value="{'userPrincipalName':'[redacted].com','newPassword':'MyNewPass2014'}"/>
        <input id="submit" value="CSRF"/>
    </form>
</body>
</html>
```

@ciyinet

EJEMPLOS REALES



@ciyinet

DETECTAR CSRF

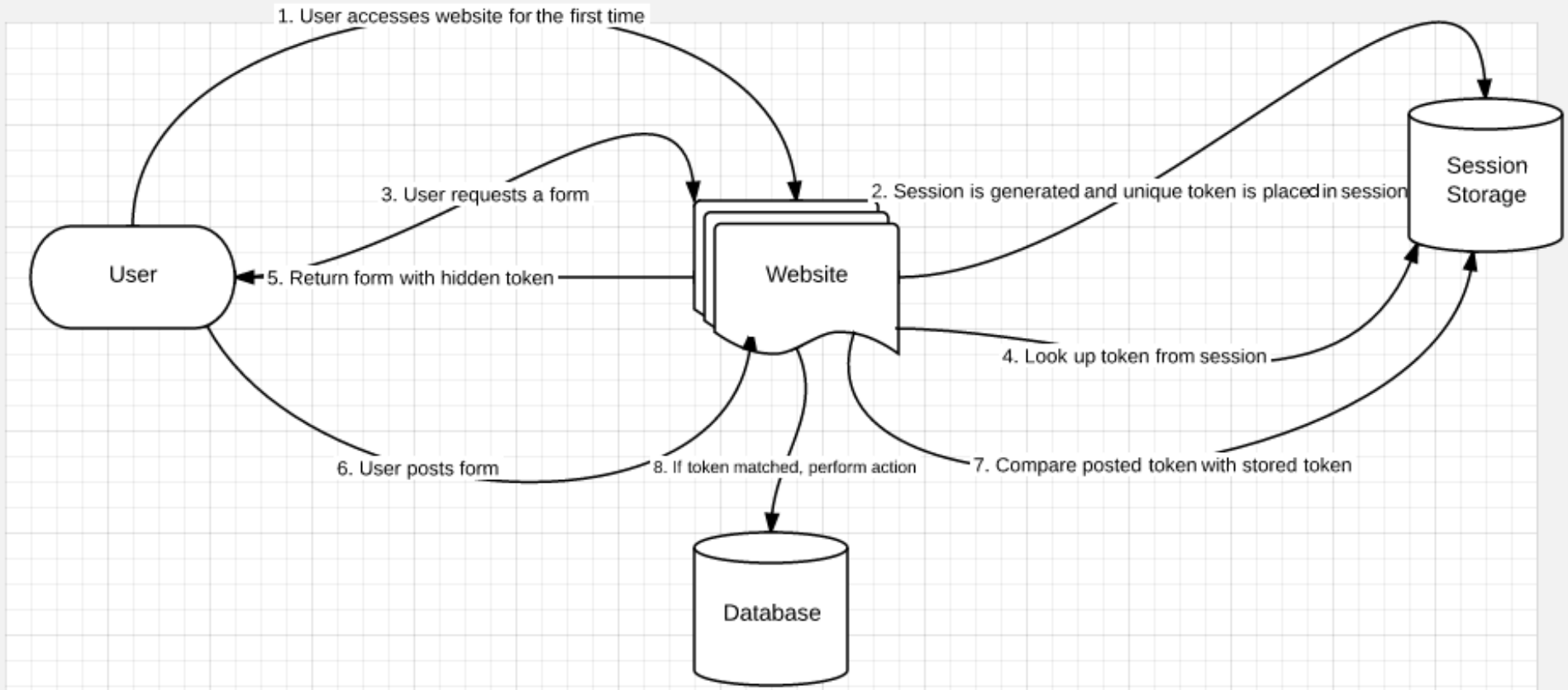
¿Es la aplicación vulnerable a CSRF?

- Comprobar si existe algún enlace o formulario sin token CSRF impredecible. Alternativa: reautenticar o usar CATPCHA.
- Especial atención en las funciones o peticiones que provocan cambios en el sistema.
- Transacciones que requieren varios pasos también son vulnerables. El atacante puede fácilmente forzar a la víctima a enviar varias peticiones seguidas.

PREVENIR CSRF

- **Token CSRF**
  - Incluir un token que no sea predecible en cada petición HTTP. El token debe ser único por cada usuario.
  - Preferiblemente el token debe ser un valor POST oculto (variables GET están más expuestas).
    - OWASP's CSRF Guard
    - OWASP's ESAPI
    - NoCRSF
    - Clsfsrpm

PREVENIR CSRF



@ciyinet

PREVENIR CSRF

- **CAPTCHA**
  - Es una alternativa que requiere al usuario probar que es real y consciente de que está realizando la petición.
- **Comprobar campo HTTP Referer**
  - Si la petición proviene de un dominio externo, asumir que es un ataque. No solventa el problema al 100%.

PREVENIR CSRF

Medidas que NO funcionan:

- **Usar token secreto en las cookies.** Las cookies son enviadas en cada petición automáticamente, sin importar si el usuario ha sido engañado para enviar la petición o no.
- **Aceptar sólo peticiones POST.** La víctima puede ser engañada a enviar un formulario en una página del atacante:
  - El formulario puede ser enviado automáticamente con JavaScript.
  - La víctima puede ser engañada a enviar un formulario con campos ocultos pensando estar enviando un formulario con otro propósito.

¡GRACIAS!

¿Preguntas?

@*ciyinet* en:

black hat®

August 2014. Las Vegas, USA