Bezpieczeństwo frameworków WEBowych Java na przykładzie ataku CSRF

O mnie

- 12 lat doświadczenia w systemach WEB
- Java/JEE
- $(ISC)^2$ CISSP
- CTO w J-LABS

software specialists

niebezpiecznik.pl

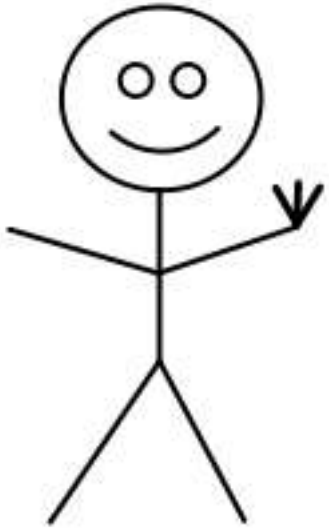
ET / HTTP/1.1
st: bank.pl
er-Agent: Mozilla/5.0

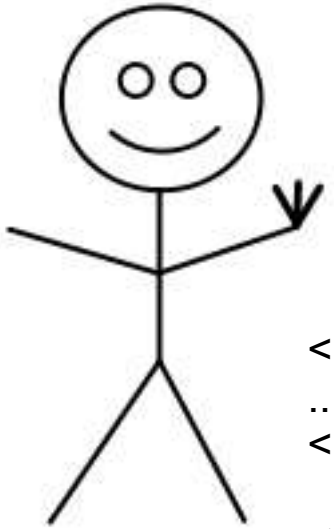
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=2UvGThyi46DQuiYXlbLp4Zft;
expires=Sun, 17-Nov-2013 12:13:13 GMT; path=/;
domain=bank.pl; httponly; Secure

TWOJA PRZEGLĄDARKA ZAWSZE WYSYŁA COOKIE

TWOJA
PRZEGLĄDARKA
ZAWSZE
WYSYŁA COOKIE

GET /favicon.png HTTP/1.1
Host: bank.pl
User-Agent: Mozilla/5.0
Cookie: JSESSIONID=2UvGThyi46DQuiYXlbLp4Zft



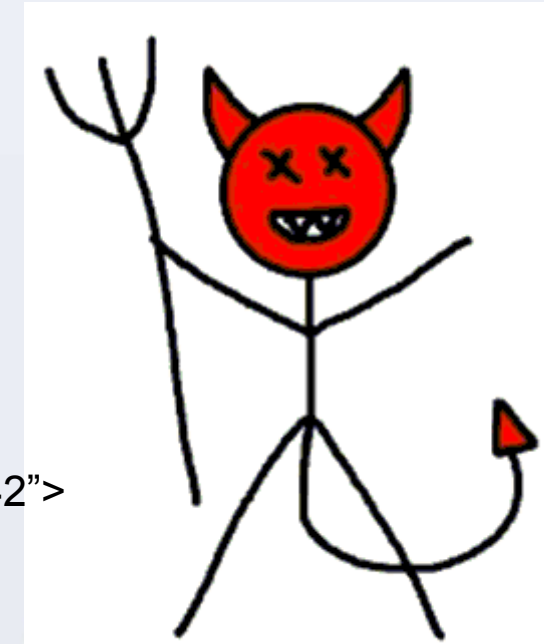


Hej zobacz jaka fajna strona

http://bit.ly/sQGznS

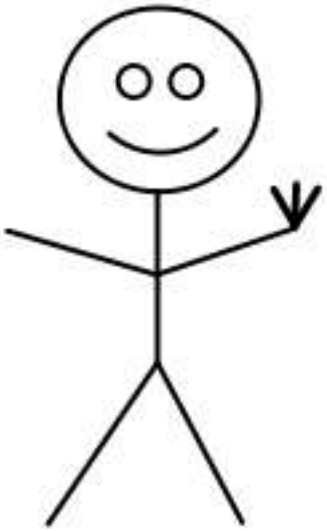
↓

```
<html>
...
<img src="http://bank.pl/przelew?z=007&do=666&kwota=42">

</html>
```



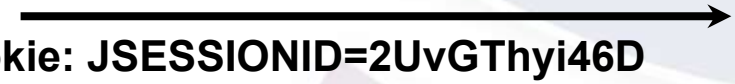
Demo

Jak to się stało?



www.fajnastrona.pl

http://bank.pl/przelew?z=007&do=666&kwota=42



Cookie: JSESSIONID=2UvGThyi46D

A można też….

```
<html>
...
<form action="http://bank.pl/przelew" method="POST" id="form">
	<input type="hidden" name="splackarte" value="true"/>
</form>
<script>
	var csrf = document.getElementById("form");
	csrf.submit();
</script>
</html>
```

Skala zjawiska – 5 w OWASP top10


FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFF
FFFFFFFFFF
FFFFF
FFUUU
UUU
UUUU
UUU
UUUU
UUU
UUUU
UUUU
UUUU
UUUU
UUUU
UUUUU
UUUU

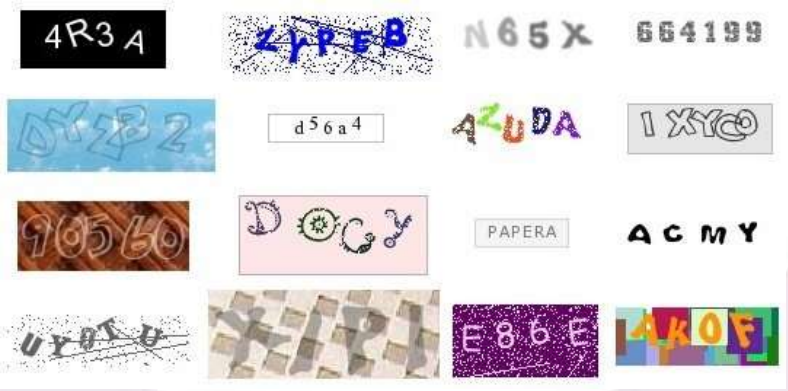
Ochrona - TOKEN

```
String token = generateRandomToken();
HttpSession session = request.getSession(false);
session.setAttribute(TOKEN, token);


PrintWriter out = response.getWriter();
out.println("<input type='hidden' name='token' value='" + token +"'/>");


… i weryfikacja…
String requestToken = request.getParameter("token");
String sessionToken = session.getAttribute(TOKEN).toString();
if ( ! sessionToken.equals(requestToken)) {….}
```

Ochrona

- ponowne uwierzytelnienie

- captcha itp.



Ochrona (od strony użytkownika)

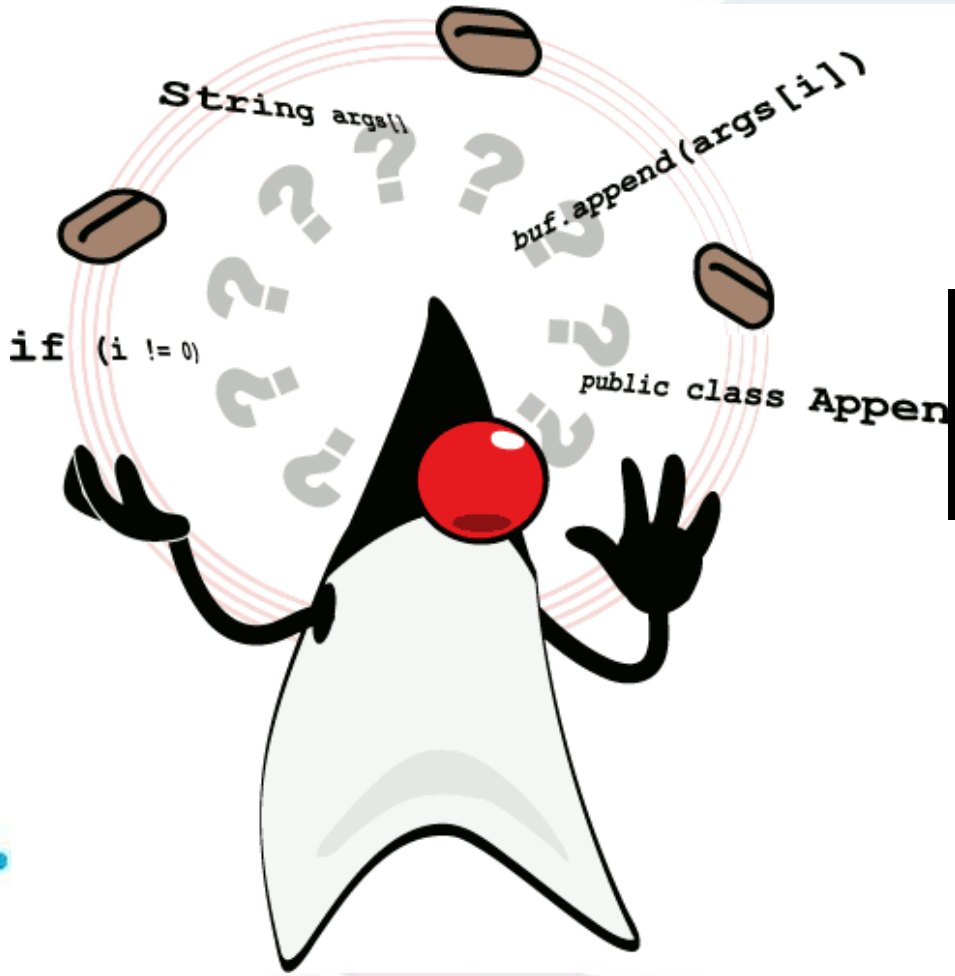
- Wyloguj się
- NoScript


GWT


apache
tapestry 5
Code less, deliver more


Seam


vaadin


String args[]
buf.append(args[i])
if (i != 0)
public class Appen


Stripes

JAVASERVER
PAGES™


JSF


Struts™


labs
software specialists


niebezpiecznik.pl


```
<%
   String token = generateRandomToken();
   session.setAttribute(TOKEN, token);
%>

<form>
   <input type='hidden' name='token' value='<%=token%>' />
   <!-- .... -->
</form>


<%
   String requestToken=request.getParameter("token");
   String sessionToken=session.getAttribute(TOKEN).toString();

   if( ! sessionToken.equals(requestToken)){//.... }
%>
```

Struts™

struts.xml:

```
<interceptors>
    <interceptor-stack name="defaultSecurityStack">
        <interceptor-ref name="defaultStack"/>
        <interceptor-ref name="tokenStack">
            <param name="excludeMethods">*</param>
        </interceptor-ref>
    </interceptor-stack>
</interceptors>

<action>
    <interceptor-ref name="defaultSecurityStack">
        <param name="tokenSession.includeMethods">*</param>
    </interceptor-ref>
</action>
```

…i w formularzu…

```
<s:token />
```


```
public class CsrfForm extends HtmlForm {
    public void encodeBegin(FacesContext context) throws IOException {
        TokenInput tokenInput = new TokenInput();
        getChildren().add(tokenInput);
        super.encodeBegin(context);
    }
}

public class TokenInput extends UIComponentBase {
    public void encodeEnd(FacesContext context) throws IOException {
        //generowanie: <input type="hidden" name="token" value="losowy token"/>
    }

    //… weryfikacja w
    public void decode(FacesContext context) {
        //weryfikacja
    }
}
```


```
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:s="http://jboss.com/products/seam/taglib">

…

<h:form>
   …
   <s:token />
</h:form>
```

Double Submit Cookies

RequestBuilder

```
RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, url);
rb.setHeader("X-XSRF-Cookie", Cookies.getCookie("myCookieKey"));
rb.sendRequest(null, myCallback);
```

GWT RPC

```
public interface ExampleInterface extends RemoteService {
    public boolean exampleMethod(String cookieValue);
    public void differentExampleMethod(String cookieValue, String arg);
}
```

wersja: 2.3+

```
public class MyServiceImpl extends XsrfProtectedServiceServlet implements
      MyService {

    public String myMethod(String s) {…}
}


public interface MyService extends XsrfProtectedService {
    public String myMethod(String s);
}
…albo…
@XsrfProtect
public interface MyService extends RemoteService {
    public String myMethod(String s);
}
```

vaadin }>

Domyślnie posiada ochronę!

Można ją wyłączyć poprzez:

```
<init-param>
    <param-name>disable-xsrf-protection</param-name>
    <param-value>true</param-value>
</init-param>
```

WICKET ma zabezpieczenie!

labs
software specialists

niebezpiecznik.pl

Nie ma wbudowanej ochrony

```
public class CSRFFilterImpl implements
    ComponentEventRequestFilter, LinkFactoryListener{
    …
}
```

… i dodajemy do modułu aplikacji.

Stripes

```
@Intercepts(LifecycleStage.EventHandling)
public class CSRFInterceptor implements Interceptor {

public Resolution intercept(ExecutionContext ctx) throws Exception {
    HttpServletRequest request = ctx.getActionBean().getContext().getRequest();
    HttpSession session = request.getSession(false);

    … weryfikacja tokenu

    return ctx.proceed();
}
```

```
<init-param>
    <param-name>Interceptor.Classes</param-name>
    <param-value>
        pl.package.name.csrf.CSRFInterceptor
    </param-value>
</init-param>
```

```
<stripes:hidden name="token" value="${sessionScope.token}" />
```

- Analiza statyczna
  - skanery kodu źródłowego (Yasca, Fortify)
  - Uwaga! Skaner nie sprawdzi, czy token jest weryfikowany!

- Analiza dynamiczna
  - kopiowanie formularzy, umieszczanie w innej domenie
  - otwarcie strony z poziomu zalogowanego użytkownika

Więcej informacji:

http://old.zope.org/Members/jim/ZopeSecurity/ClientSideTrojan
http://www.tux.org/~peterw/csrf.txt
http://www.isecpartners.com/files/XSRF_Paper_0.pdf
http://en.wikipedia.org/wiki/Samy_%28XSS%29
http://www.darkreading.com/security/application-security/208804131/index.html
http://directwebremoting.org/blog/joe/2007/01/01/csrf_attacks_or_how_to_avoid_exposing_your_gmail_contacts.html
https://www.owasp.org/index.php/Top_10_2007-A5
http://en.wikipedia.org/wiki/Cross-site_request_forgery
https://www.owasp.org/index.php/Top_10_2010-A5
http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html
http://java.dzone.com/articles/preventing-csrf-jsf-20
http://code.google.com/webtoolkit/articles/security_for_gwt_applications.html#xsrf
http://code.google.com/webtoolkit/doc/latest/DevGuideSecurityRpcXsrf.html
http://nickcoblentz.blogspot.com/2008/11/csrf-prevention-in-struts-2.html
http://wiki.apache.org/tapestry/Tapestry5CSRF
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29
https://www.owasp.org/index.php/Reviewing_code_for_Cross-Site_Request_Forgery_issues
http://www.cgisecurity.com/csrf-faq.html
http://shiflett.org/articles/cross-site-request-forgeries
http://pl.wikipedia.org/wiki/Cross-site_request_forgery
http://www.w3schools.com/JS/js_cookies.asp
http://docs.jboss.org/seam/2.1.2/reference/en-US/html/controls.html#d0e28825
http://seamframework.org/Documentation/CrossSiteRequestForgery

Dziękujemy

Piotr Bucki
piotr.bucki@j-labs.pl

Szymon Janowski
szymon.janowski@j-labs.pl

Piotr Konieczny
pk@niebezpiecznik.pl