# Web 服务器加固

## Apache

- 2013-07-22Apache HTTP Server mod_session_dbd 远程安全漏洞(CVE-2013-2249)
- 2013-07-17Apache HTTP Server mod_dav.c 拒绝服务漏洞(CVE-2013-1896)
- 2013-07-09Apache CXF 多个远程拒绝服务漏洞(CVE-2013-2160)
- 2013-06-28Apache XML Security 签名伪造漏洞
- 2013-05-13Apache HTTP Server 日志内终端转义序列命令注入漏洞
- 2013-03-06Apache Commons FileUpload 不安全临时文件创建漏洞(CVE-2013-0248)
- 2013-02-26Apache HTTP Server 多个模块主机名和 URI 跨站脚本漏洞
- 2013-02-26Apache HTTP Server balancer_handler 函数跨站脚本漏洞(CVE-2012-4558)
- 2012-09-13Apache HTTP Server envvars 本地权限提升漏洞
- 2012-09-12Apache 'mod_pagespeed'模块跨站脚本执行和安全限制绕过漏洞
- 2012-08-21Apache 2.4.2 HTTP Server HTML 注入和信息泄露漏洞
- 2012-08-09Apache QPID NullAuthenticator 验证绕过漏洞
- 2012-06-09Apache CXF Failed Element Verification(CVE-2012-2379)
- 2012-05-21Oracle Weblogic Apache Connector POST Request Buffer Overflow
- 2012-04-17Apache HTTP Server 'LD_LIBRARY_PATH'不安全库装载任意代码执行漏洞
- 2012-04-17Apache HTTP Server 'LD_LIBRARY_PATH'不安全库加载任意代码执行漏洞
- 2012-03-22Apache Traffic Server HTTP 主机标头处理缓冲区溢出漏洞
- 2012-03-21Apache Wicket 隐藏文件信息泄露漏洞
- 2012-03-21Apache Wicket 'pageMapName'参数跨站脚本执行漏洞
- 2012-03-20Apache 'mod_fcgid'模块 2.3.6 拒绝服务漏洞
- 2012-02-06Apache HTTP Server "mod_proxy"反向代理安全限制绕过漏洞
- 2012-02-01Apache httpOnly Cookie Disclosure(CVE-2012-0053)
- 2012-01-23Apache HTTP Server "httpOnly" Cookie 信息泄露漏洞
- 2012-01-13Apache 2.2.x Scoreboard 本地安全限制绕过漏洞
- 2011-12-16Remote Apache Denial of Service Exploit
- 2011-12-09Apache Range Header Denial Of Service
- 2011-11-24Apache HTTP Server mod_proxy 反向代理模式安全限制绕过漏洞
- 2011-11-10Apache HTTP Server 'ap_pregsub()'函数本地拒绝服务漏洞(CVE-2011-4415)
- 2011-11-02Apache HTTP Server "ap_pregsub()"函数本地权限提升漏洞
- 2011-10-22Apache HTTP server Denial of service venerability



## Nginx

- 2013-07-11nginx 1.3.9/1.4.0 x86 Brute Force Remote Exploit
- 2013-05-17nginx 1.3.9-1.4.0 DoS PoC

- 2013-05-13Nginx proxy_pass 模块远程安全漏洞(CVE-2013-2070)
- 2013-05-07nginx 'ngx_http_parse.c'栈缓冲区溢出漏洞
- 2013-04-25nginx 'ngx_http_close_connection()'远程整数溢出漏洞
- 2013-04-19nginx 0.6.x Arbitrary Code Execution NullByte Injection
- 2013-02-21Nginx 'access.log'不安全文件权限漏洞
- 2013-01-03nginx 中间人攻击漏洞(CVE-2011-4968)
- 2012-10-15nginx "location"受限制资源服务漏洞
- 2012-07-06Nginx Naxsi 模块'nx_extract.py'脚本远程文件泄露漏洞
- 2012-04-12nginx 'ngx_http_mp4_module.c'缓冲区溢出漏洞
- 2012-03-15nginx 'ngx_cpystrn()'信息泄露漏洞(CVE-2012-1180)
- 2012-03-15Nginx 1.0.x 标头解析内存泄露漏洞
- 2011-12-26nginx 配置错误而导致目录遍历漏洞
- 2011-11-17nginx DNS 解析器远程堆缓冲区溢出漏洞
- 2011-07-20Nginx %00 空字节执行任意代码(php)漏洞
- 2010-08-29nginx v0.6.38 Heap Corruption Exploit
- 2010-06-14nginx HTTP 请求源码泄露和拒绝服务漏洞
- 2010-06-12Nginx <= 0.7.65 / 0.8.39 (dev) Source Disclosure / Download Vulnerability
- 2010-06-12nginx 0.8.36 Source Disclosure and DoS Vulnerabilities
- 2010-05-30nginx [engine x] http server <= 0.6.36 Path Draversal
- 2010-05-30Nginx 0.8.35 Space Character Remote Source Disclosure
- 2010-05-28nginx 不正确处理 URI 导致目录遍历漏洞
- 2010-05-20nginx 文件路径处理远程命令执行漏洞
- 2009-10-27nginx ngx_http_process_request_headers()函数空指针引用拒绝服务漏洞
- 2009-10-26New nginx packages fix denial of service
- 2009-10-24nginx dos exploit
- 2009-10-23nginx 0.7.0-0.7.61 0.6.0-0.6.38 0.5.0-0.5.37 0.4.0-0.4.14 PoC
- 2009-09-23nginx WebDAV 目录遍历漏洞
- 2009-09-23nginx 0.7.61 WebDAV directory traversal



# ModeSecurity

https://www.owasp.org/index.php/Category:OWASP_Securing_WebGoat_using_ModSecurity_Project

\\vmware-host\Shared Folders\桌面\安全\安全防御\WAF\ModSecurity

# Web 应用服务器加固

## JBOSS

- 2013-07-12JBoss RichFaces 远程代码执行漏洞(CVE-2013-2165)
- 2013-06-26Red Hat JBoss Application Server 密码信息泄露漏洞(CVE-2013-3734)
- 2013-02-20JBoss Enterprise Portal GateIn Portal XML 解析任意文件读取漏洞
- 2013-02-20JBoss Enterprise Portal GateIn Portal 未验证站点导入漏洞
- 2013-01-08JBoss Enterprise Portal Platform 多个跨站脚本执行漏洞
- 2012-12-19JBoss Enterprise Application Platform 安全绕过漏洞(CVE-2012-4549)
- 2012-12-18JBoss Enterprise Application Platform 安全绕过漏洞
- 2012-07-23JBoss Enterprise Application Platform 安全限制绕过漏洞
- 2012-06-29JBoss Enterprise Application Platform SecurityAssociation.getCredential() 安全绕过漏洞
- 2012-06-22JBoss Enterprise BRMS Platform JGroups Diagnostics Service 信息泄露漏洞
- 2012-06-20JBoss Enterprise Application Platform/JBoss Enterprise Web Platform 安全绕过漏洞
- 2012-02-21JBoss Enterprise Application Platform 多个安全绕过漏洞(CVE-2012-0874)
- 2012-02-02JBoss Web 远程拒绝服务漏洞(CVE-2011-4610)
- 2012-02-02JBoss Operations Network 多个安全限制绕过漏洞
- 2012-01-18JBoss 'mod_cluster'安全绕过漏洞
- 2012-01-12JBoss Cache 'NonManagedConnectionFactory.java'本地信息泄露漏洞
- 2011-12-16JBoss Operations Network 多个跨站脚本执行漏洞
- 2011-12-02JBoss Application Server 管理控制台跨站脚本执行漏洞
- 2011-12-02JBoss Application Server 跨站请求伪造漏洞
- 2011-11-17JBoss Enterprise SOA Platform 调用程序身份验证绕过漏洞
- 2011-11-17JBoss Enterprise SOA 平台 Servlet 调用器验证绕过漏洞
- 2011-10-13JBoss AS Remote Exploit v2
- 2011-10-03JBoss addURL Misconfiguration Attack
- 2010-09-01JBoss RichFaces Online Persistent Xss Vulnerability
- 2010-07-27JBoss Seam 参数化 EL 表达式远程代码执行漏洞
- 2010-04-26JBoss 企业应用平台多个非授权访问漏洞
- 2010-01-08JBOSS addURL()函数 远程代码执行漏洞
- 2009-12-10JBoss 企业应用平台多个安全漏洞
- 2008-09-24JBoss Enterprise Application Platform 类文件信息泄漏漏洞
- 2008-08-06JBoss Enterprise Application Platform 信息泄漏漏洞

# JBOSS 远程命令执行

JBOSS 默认配置会有一个后台漏洞，漏洞发生在 **jboss.deployment** 命名空间

中的 **addURL()**函数,该函数可以远程下载一个 war 压缩包并解压

访问 http://www.safe3.com.cn:8080/jmx-console/ 后台，如下图


ObjectName Filter (e.g. "jboss:*", "*:service=invoker,*") :

## Catalina

- type=Server
- type=StringCache

## JMImplementation

- name=Default,service=LoaderRepository



下拉找到如下图所示

## jboss.deployment

- flavor=URL,type=DeploymentScanner

## jboss.ejb

点击 **flavor=URL,type=DeploymentScanner** 进入

**void addURL()**

MBean Operation.

| Param | ParamType | ParamValue | ParamDescription |
|---|---|---|---|
| p1 | java.lang.String | http://www.safe3.com.cn/( | (no description) |

Invoke

在输入框中写入 war 压缩文件 webshell 的 url 地址，如上图

点击 invoke 执行界面获得一个 jsp 的 webshell，如下图




临时漏洞修补办法：给 jmx-console 加上访问密码

1.在 S{jboss.server.home.dir}/deploy 下面找到 jmx-console.war 目录编辑 WEB-INF/web.xml 文件 去掉 security-constraint 块的注释，使其起作用

2.编辑 WEB-INF/classes/jmx-console-users.properties 或 server/default/conf/props/jmx-console-users.properties (version >=4.0.2)

和 WEB-INF/classes/jmx-console-roles.properties

或 server/default/conf/props/jmx-console-roles.properties(version >=4.0.2) 添加用户名密码

3.编辑 WEB-INF/jboss-web.xml 去掉 security-domain 块的注释，security-domain 值的映射文件为 login-config.xml （该文件定义了登录授权方式）

## Tomcat

- 2013-06-01Apache Tomcat 表单验证功能安全绕过漏洞

- 2013-05-29Apache Tomcat 摘要验证不完整修复安全漏洞
- 2013-05-28Apache Tomcat 不安全临时文件处理漏洞(CVE-2013-1976)
- 2013-05-10Apache Tomcat 信息泄露漏洞(CVE-2013-2071)
- 2013-05-03Apache Tomcat DIGEST Authentication 重放攻击漏洞(CVE-2013-2051)
- 2013-02-22Apache Tomcat 'log/logdir'目录不安全文件权限漏洞
- 2012-12-04Apache Tomcat 跨站请求伪造漏洞
- 2012-12-04Apache Tomcat FORM 身份验证安全绕过漏洞
- 2012-11-27Apache Tomcat 拒绝服务漏洞(CVE-2012-5568)
- 2012-11-05Apache Tomcat DIGEST 身份验证多个安全漏洞(CVE-2012-3439)
- 2012-01-18Apache Tomcat Request Information Disclosure
- 2012-01-18Apache Tomcat Large Number Denial Of Service
- 2012-01-17Apache Tomcat 请求对象安全限制绕过漏洞
- 2011-12-29Apache Tomcat Web 表单哈希冲突拒绝服务漏洞
- 2011-11-09Apache Tomcat 管理应用程序安全限制绕过漏洞
- 2011-08-15Apache Tomcat 信息泄露漏洞(CVE-2011-2481)
- 2011-07-29Apache Tomcat SecurityManager Security Bypass Vulnerability
- 2011-07-14Apache Tomcat sendfile 请求安全限制绕过和拒绝服务漏洞
- 2011-06-27Apache Tomcat "MemoryUserDatabase"信息泄露漏洞
- 2011-03-09Apache Tomcat "@ServletSecurity" 注释安全限制绕过漏洞
- 2010-07-29Apache Tomcat < 6.0.18 UTF8 Directory Traversal Vulnerability
- 2010-04-22Apache Tomcat v. 5.5.0 to 5.5.29 & 6.0.0 to 6.0.26 information disclosure vulnerability
- 2010-04-22Apache Tomcat 认证头信息泄露漏洞
- 2010-01-30Apache Tomcat v.5.5.26 Directory Traversal
- 2009-12-01Apache Tomcat v3.2.1 404 Error Page Cross Site Scripting Vulnerability
- 2009-12-01Apache Tomcat 404 错误页跨站脚本漏洞
- 2009-11-09Apache Tomcat Windows 安装程序默认空口令漏洞
- 2009-11-09Apache Tomcat Cookie Quote Handling Remote Information Disclosure Vulnerability
- 2009-11-09Apache Tomcat Form Authentication Username Enumeration Weakness
- 2009-06-04Apache Tomcat XML 解析器非授权文件读写漏洞



参考：

https://www.owasp.org/index.php/Securing_tomcat

http://wiki.apache.org/tomcat/FAQ/Security

http://tomcat.apache.org/security.html

# HPP 防御

Web 编程设计时注意不同服务器的取值顺序

关于 **Http Parameter Pollution**http://hi.baidu.com/aullik5/item/860da508a90709843c42e2ca

据 harry 的小道消息说，他几个月前就看到这篇 paper 了。#_#

黑哥前几天拿到了，给我们看了下，确实挺有意思，也可以算作是一种新的攻击类型。

因为今天 wisec（作者）的演讲已经出来了，所以就可以在 blog 上写写了。

以下摘自作者 blog：http://www.wisec.it/sectou.php?id=4a129d3c810ec

On May 14th @ 2009OWASP Appsec Poland, me & Luca Carettoni presented a new attack category called Http Parameter Pollution (HPP).

HPP attacks can be defined as the feasibility to override or add HTTP GET/POST parameters by injecting query string delimiters.
It affects a building block of all web technologies thus server-side and client-side attacks exist.
Exploiting HPP vulnerabilities, it may be possible to:

- Override existing hardcoded HTTP parameters.
- Modify the application behaviors.
- Access and, potentially exploit, uncontrollable variables.
- Bypass input validation checkpoints and WAFs rules.



Just to whet your appetite, I can anticipate that by researching for real world HPP vulnerabilities, we found issues on some Google Search Appliance front-end scripts, Ask.com, Yahoo! Mail Classic and several other products.

You can download the slides of the talk here (pdf) or browse it on Slideshare .

简单来说，就是提交两个相同的参数，不同的服务器会有不同的处理。

比如提交 **/?a=fvck&a=svck**

那么在有的 webserver 和语言中，会只取第一个。

在别的环境的中，比如.net 环境中，会变成 a=fvck,svck

这在绕过一些逻辑判断的时候，会非常有用。

作者把成果可以整理为一张表：

| Technology/HTTP back-end | Overall Parsing Result | Example |
|---|---|---|
| ASP.NET/IIS | All occurrences of the specific parameter | par1=val1,val2 |
| ASP/IIS | All occurrences of the specific parameter | par1=val1,val2 |
| PHP/Apache | Last occurrence | par1=val2 |
| PHP/Zeus | Last occurrence | par1=val2 |
| JSP,Servlet/Apache Tomcat | First occurrence | par1=val1 |
| JSP,Servlet/Oracle Application Server 10g | First occurrence | par1=val1 |
| JSP,Servlet/Jetty | First occurrence | par1=val1 |
| IBM Lotus Domino | Last occurrence | par1=val2 |
| IBM HTTP Server | First occurrence | par1=val1 |
| mod_perl,libapreq2/Apache | First occurrence | par1=val1 |
| Perl CGI/Apache | First occurrence | par1=val1 |
| mod_perl,lib???/Apache | Becomes an array | ARRAY(0x8b9059c) |
| mod_wsgi (Python)/Apache | First occurrence | par1=val1 |
| Python/Zope | Becomes an array | ['val1', 'val2'] |
| IceWarp | Last occurrence | par1=val2 |
| AXIS 2400 | All occurrences of the specific parameter | par1=val1,val2 |
| Linksys Wireless-G PTZ Internet Camera | Last occurrence | par1=val2 |
| Ricoh Aficio 1022 Printer | First occurrence | par1=val1 |
| webcamXP PRO | First occurrence | par1=val1 |
| DBMan | All occurrences of the specific parameter | par1=val1~~val2 |

我个人倒是觉得，比较好的做法是返回一个数组

比如 GAE（Google App Engine）里有个方法就是 get()，会返回相同参数名的第一个参数值，还有方法是 get_all()，会返回所有的相同参数的值，为一个 list

虽然 paper 中有提到利用此绕过 IE 8 的 XSS filter，不过后来证实是 YY 的。

# 应用层加固



**HttpServletRequest** 输入验证

https://www.owasp.org/index.php/How_to_add_validation_logic_to_HttpServletRequest

**How to perform HTML entity encoding in Java**

## Status

Deleted 9/16/2010

## Why was this article deleted

HTML Entity Encoding is not enough to stop XSS in web applications. Please see XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet for more information.

https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

## Why Can't I Just HTML Entity Encode Untrusted Data?

HTML entity encoding is okay for untrusted data that you put in the body of the HTML document, such as inside a  tag. It even sort of works for untrusted data that goes into attributes, particularly if you're religious about using quotes around your attributes. But HTML entity encoding doesn't work if you're putting untrusted data inside a <script> tag anywhere, or an event handler attribute like onmouseover, or inside CSS, or in a URL. So even if you use an HTML entity encoding method everywhere, you are still most likely vulnerable to XSS. **You MUST use the escape syntax for the part of the HTML document you're putting untrusted data into.** That's what the rules below are all about.

## You Need a Security Encoding Library

Writing these encoders is not tremendously difficult, but there are quite a few hidden pitfalls. For example, you might be tempted to use some of the escaping shortcuts like \" in JavaScript. However, these values are dangerous and may be misinterpreted by the nested parsers in the browser. You might also forget to escape the escape character, which attackers can use to neutralize your attempts to be safe. OWASP recommends using a security-focused encoding library to make sure these rules are properly implemented.

Microsoft provides an encoding library named the Microsoft Anti-Cross Site Scripting Library for the .NET platform and ASP.NET Framework has built-inValidateRequest function that provides **limited** sanitization.



The OWASP ESAPI project has created an escaping library for Java. OWASP also provides the OWASP Java Encoder Project for high-performance encoding.

## OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project#tab=Main

## Java 验证码

https://www.owasp.org/index.php/Captchas_in_Java

## Overview

CAPTCHA stands for "**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part". A CAPTCHA typically takes the form of an image containing distorted letters and/or numbers and is often used on web sites where it is important to determine whether the user is a real person or a computer program. CAPTCHA's have some drawbacks such as the lack of accessibility for the visually impaired, high CPU usage, and possible circumvention techniques. In many cases, however, CAPTCHA's can help mitigate certain types of attacks launched by malicious individuals using automated tools. At a minimum they can raise the bar enough to deter a casual attacker.

### JCaptcha

*JCaptcha* is a Java implementation of captcha technology developed by Marc Antoine Garrigue and released as open source. JCaptcha provides Java programmers with a framework and toolset for deploying CAPTCHA's in their web applications. You can download the full package that includes JARs, API documentation, and source code from SourceForge. The latest release of JCaptcha (1.0) is available under the GNU General Public License, but later releases will be under the LGPL license.

### Examples

- A Simple Captcha Servlet: how it works | download the war

### SimpleCaptcha

*SimpleCaptcha* is another framework that provides Java programmers with the ability to easily add a CAPTCHA to their web applications. A number of default CAPTCHA servlets are provided, including a Chinese version. The visual representation of the CAPTCHA text can be altered using a number of filtering methods, but SimpleCaptcha does not provide an audio CAPTCHA, which limits its accessibility for the visually impaired.



### Examples

- SimpleCaptcha Example: J2EE Sample

### reCAPTCHA

*reCAPTCHA* is a CAPTCHA web service that provides a visual and an audio CAPTCHA. The visual CAPTCHAs generated by reCAPTCHA cannot be customized, but are derived from text in scanned books that have already failed to be recognized by OCR technology. A number of plugins for different programming languages have been developed including a Java plugin, but it is fairly straightforward to develop your own plugin. One of the major downsides of using a web service such as reCAPTCHA is that the service might go down while your application is still running. In the case of failure, your application will either need to fail open or fail closed.

# 密码强度验证

https://www.owasp.org/index.php/Password_length_%26_complexity

The generateStrongPassword() method uses the ESAPI randomizer to build strong passwords comprised of upper & lower case letters, digits, and special characters. Currently, all character sets are hard-coded in ESAPI Encoder.java. There are plans to make these user configurable. The source code is available at Authenticator.java. For more information on OWASP ESAPI follow this link: OWASP ESAPI

**Sample Code**

Below is a sample Servlet that uses the above methods to create a new password, and compare it to a blank string.

```
import org.owasp.esapi.*;
import org.owasp.esapi.interfaces.IAuthenticator;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class generateStrongPassword extends HttpServlet {

        public static final String RESOURCE_DIRECTORY =
"org.owasp.esapi.resources";
        private static String resourceDirectory =
System.getProperty(RESOURCE_DIRECTORY);

        public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException{
                response.setContentType("text/html");
                IAuthenticator passInstance =
ESAPI.authenticator();
                PrintWriter out = response.getWriter();
                String password =
passInstance.generateStrongPassword();

                try {
                    passInstance.verifyPasswordStrength(password,
"");
                    out.println("New password is strong!");
            } catch (Exception e) {
                    out.println("New password is not strong enough");
            }
                out.println(password);
```

```
                }
        }
```

# JAAS

JavaAuthenticationAuthorizationService（JAAS，Java 验证和授权 API）

https://www.owasp.org/index.php/JAAS_Tomcat_Login_Module

# Logout（退出、注销）

https://www.owasp.org/index.php/Logout

Session logout have to objective to cancel conversation established between the browser and the web server. We means here, by conversation, several browser request and response that has been linked between them.

## Steps of session logout process

Logout is composed by 2 steps:

1. Invalidate user session (indicate to web server that the session is not used anymore).
2. Cancel cookie send by the web server to track user session (and also all cookies sent by web application, this, in order to have a global clean state).



## Code sample of session logout process

```
package org.owasp.javaproject.logout;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
/**
 * Code sample showing how to perform a complete logout
 */
@SuppressWarnings("serial")
@WebServlet("/Logout")
public class LogoutCodeSample extends HttpServlet {

        /**
         * {@inheritDoc}
         *
         * @see
javax.servlet.http.HttpServlet#doPost(javax.servlet.http.HttpServletR
equest,
         *      javax.servlet.http.HttpServletResponse)
         */
        @Override
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                doGet(request, response);
        }

        /**
         * {@inheritDoc}
         *
         * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
         *      response)
         */
        @Override
        protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                /*
                 * First step : Invalidate user session
                 */
                HttpSession session = request.getSession(false);
                if (session != null) {
                        session.invalidate();
                }

                /*
                 * Second step : Invalidate all cookies by, for each cookie
received,
```

```
                 * overwriting value and instructing browser to deletes
it
                 */
                Cookie[] cookies = request.getCookies();
                if (cookies != null && cookies.length > 0) {
                        for (Cookie cookie : cookies) {
                                cookie.setValue("-");
                                cookie.setMaxAge(0);
                                response.addCookie(cookie);
                        }
                }

        }

}
```

## Session 超时

https://www.owasp.org/index.php/Session_Timeout

Session timeout represents the event occuring when a user do not perform any action on a web site during a interval (defined by web server). The event, on server side, change the status of the user session to 'invalid' (ie. "not used anymore") and instruct the web server to destroy it (deleting all data contained into it).

## Define the session timeout

On JEE web application , there 2 ways to define session timeout,

- Declaratively in web deployment descriptor (file "web.xml") : This definition is applied to all session created for the application.
- Programmatically on session object : This definition apply only on current session.

### Timeout defined declaratively

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
```

```
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        id="WebApp_ID" version="3.0">

        ...

        <!-- Define a session timeout to 15 minutes -->
        <session-config>
                <session-timeout>15</session-timeout>
        </session-config>

        ...


</web-app>
```

**Timeout defined Programmatically**

```
package org.owasp.javaproject.sessiontimeout;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Code sample showing how to access to session timeout and act on it.
 */
@SuppressWarnings("serial")
@WebServlet("/SessionTimeout")
public class SessionTimeoutCodeSample extends HttpServlet {

        /**
         * {@inheritDoc}
         *
         * @see
javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRe
quest,
         *     javax.servlet.http.HttpServletResponse)
```

```
         */
        @SuppressWarnings("boxing")
        @Override
        protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
                // Get reference on session object
                HttpSession session = req.getSession();

                // Display session timeout value defined in "web.xml"
                // Value here is specified in seconds...
                System.out.printf("Session timeout defined at
application level : %s\n", session.getMaxInactiveInterval());

                // Change session timeout for this session and display
new timeout value
                // Value here is defined in seconds...
                session.setMaxInactiveInterval(60);
                System.out.printf("Session timeout defined at code
level : %s\n", session.getMaxInactiveInterval());
        }

}
Session timeout defined at application level : 900
Session timeout defined at code level       : 60
```



## Impact of the session timeout on security and best practices

Session timeout define action window time for a user thus this window represents, in the same time, the delay in which an attacker can try to steal and use a existing user session...

For this, it's best practices to :

- Set session timeout to the minimal value possible depending on the context of the application.
- Avoid "infinite" session timeout.
- Prefer declarative definition of the session timeout in order to apply global timeout for all application sessions.
- Trace session creation/destroy in order to analyse creation trend and try to detect anormal session number creation (application profiling phase in a attack).

## 预防 Session Fixation

https://www.owasp.org/index.php/Session_Fixation_in_Java

- Session ID should be regenerated after login, and switching in and out of SSL

```
session.invalidate();
session=request.getSession(true);
```

Comment: Please note that JBOSS has proven to NOT regenerate a new session id via this method as of December 2007.

- Disable URL rewriting

Comment: What threat does this mitigate?
Answer: Disabling of URL rewriting mitigates Session Hijacking/Session Sniffing via session id's being exposed in a GET parameter of your url's (as well as being stored in your browser history, proxy servers, etc). See Session hijacking attack

## Java 加密 - JCE

https://www.owasp.org/index.php/Using_the_Java_Cryptographic_Extensions



## Struts

【OWASP】Struts 安全
http://automationqa.com/forum.php?mod=viewthread&tid=2885&fromuid=21

## Hibernate 安全要点

https://www.owasp.org/index.php/Hibernate
http://software-security.sans.org/developer-how-to/fix-sql-injection-in-java-hibernate
https://www.owasp.org/index.php/Hibernate-Guidelines

## iBatis

http://software-security.sans.org/developer-how-to/fix-sql-injection-in-java-mybatis

## CSRF 防御

### 检测工具 - CSRFTester

https://www.owasp.org/index.php/Category:OWASP_CSRFTester_Project

### Java 防御开发包 - CSRFGuard

https://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project

https://github.com/esheri3/OWASP-CSRFGuard

CSRF&OWASP CSRFGuard

http://automationqa.com/forum.php?mod=viewthread&tid=2919&fromuid=21



## Java 安全编程规范

OWASP 安全编码规范快速参考指南

https://www.owasp.org/images/7/73/OWASP_SCP_Quick_Reference_Guide_%28Chinese%29.pdf

\\vmware-host\Shared Folders\桌面\安全\安全防御\OWASP Secure Coding Practices**Java** 安全编码规范与案例分析 **V0.2.docx**

OWASP_Code_Review_Guide-V1_1.doc

# Java 安全库

## Java Security Libraries

| Name and link | Updated | AU | AC | CF | CR | IV | OE |
|---|---|---|---|---|---|---|---|
| AntiSami | 2011 | | | | | Y | Y |
| Apache Santuarrio | 2011 | | | | | | |
| Apache Shiro | 2011 | Y | Y | ? | Y | ? | Y |
| Bouncy Castle | 2011 | | | | Y | | |
| CSRFGuard | ? | | | Y | Y | | |
| ESAPI | 2010 | Y | Y | ? | Y | Y | Y |
| Jasypt | 2010 | | | | Y | | |
| iGuard | 2011 | Y | Y | | | | |
| Vlad | ? | | | | | Y | |

## Security Features Key

- AU Authentication
- AC Authorization / Access Control
- CF Anti CSRF
- CR Cryptography
- IV Input Validation
- OE Output encoding
- SM Session management
- XM XML security
- XS XSS protection



## AntiSamy

https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project

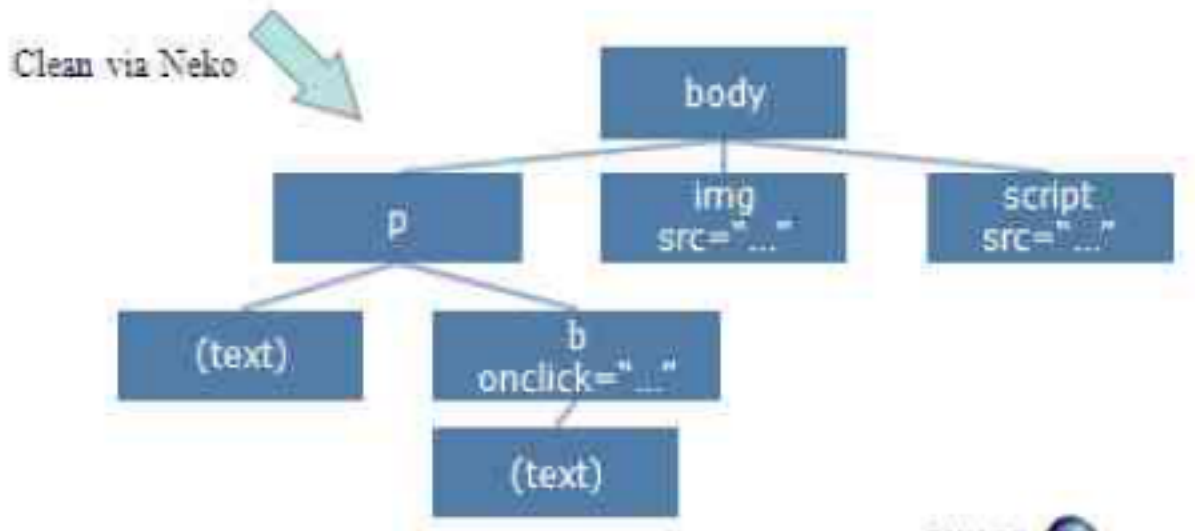
https://code.google.com/p/owaspantisamy/downloads/list

an HTML validation tool and API

主要用于防御 XSS

## How Does It Work? (cont)

```
<body>
 <p>
  This is <b onclick="alert(bang!)">so</b> cool!!
 <img src="http://example.com/logo.jpg">
 <script src="http://evil.com/attack.js">
</body>
```




OWASP

整合到 ESAPI 中



## Scrubbr

https://www.owasp.org/index.php/Category:OWASP_Scrubbr

https://code.google.com/p/owaspscrubbr/

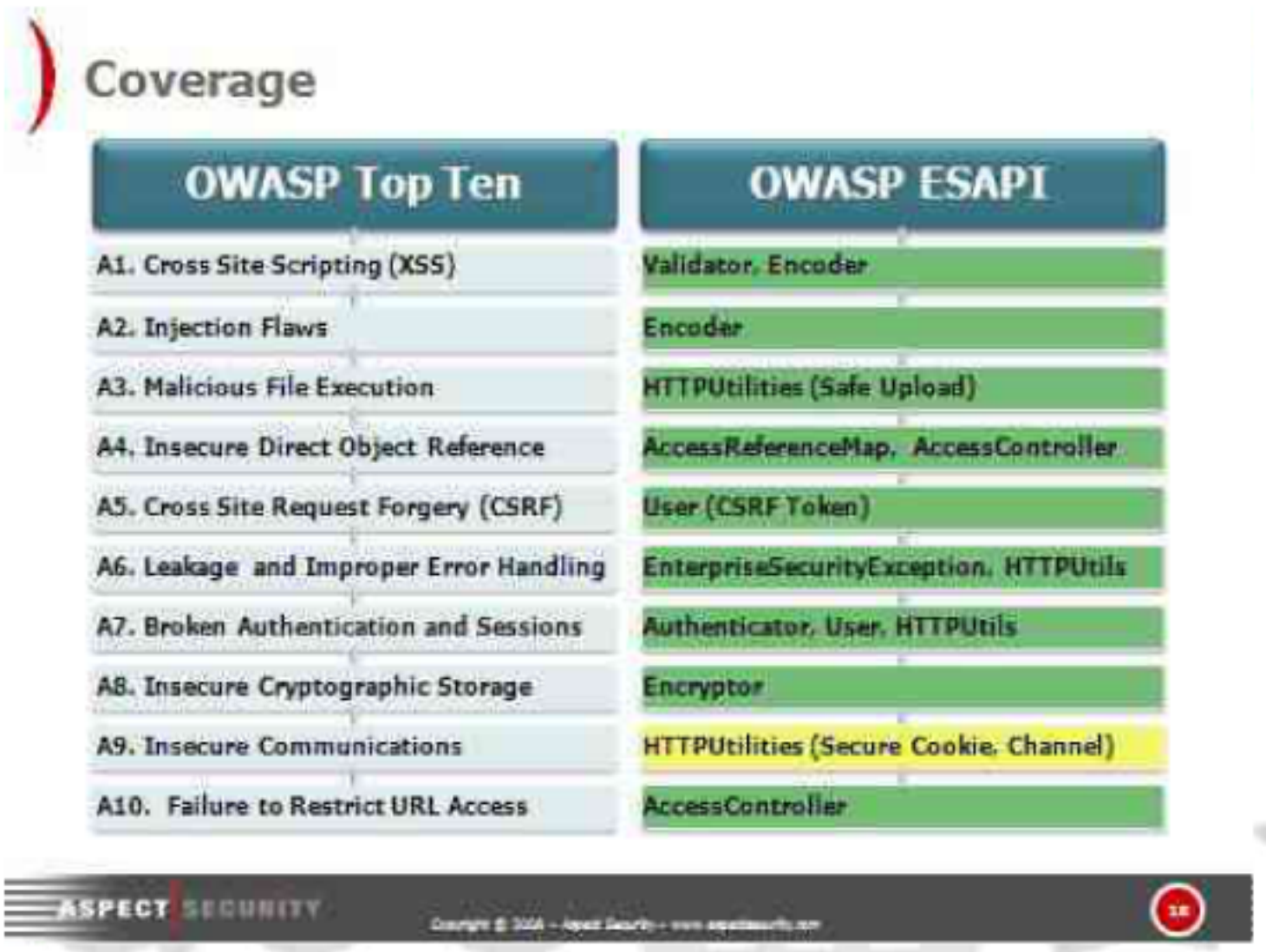
Scrubbr is a BSD-licensed database scanning tool that checks numerous database technologies for the presence of possible **stored cross-site scripting attacks**. The tool was partially inspired by "Scrawlr", a trimmed-down version of HP's WebInspect which was released for free after the so-called "asprox" mass-SQL injection bot exploited hundreds of thousands of insecure ASP sites.

In a similar vein, Aspect Security generously donated Scrubbr to OWASP to help people get some visibility into their databases and check for malicious data.

查找数据库中潜在的存储型 XSS 漏洞

# ESAPI

https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API#tab=Downloads




**Banned Java APIs:**

System.out.println() -> Logger.*

Throwable.printStackTrace() -> Logger.*

Runtime.exec() -> Executor.safeExec()

Reader.readLine() -> Validator.safeReadLine()

Session.getId() -> Randomizer.getRandomString() (better not to use at all)

ServletRequest.getUserPrincipal() -> Authenticator.getCurrentUser()

ServletRequest.isUserInRole() -> AccessController.isAuthorized*()

Session.invalidate() -> Authenticator.logout()

Math.Random.* -> Randomizer.*

File.createTempFile() -> Randomizer.getRandomFilename()

ServletResponse.setContentType() -> HTTPUtilities.setContentType()

ServletResponse.sendRedirect() -> HTTPUtilities.sendSafeRedirect()

RequestDispatcher.forward() -> HTTPUtilities.sendSafeForward()

ServletResponse.addHeader() -> HTTPUtilities.addSafeHeader()

ServletResponse.addCookie() -> HTTPUtilities.addSafeCookie()

ServletRequest.isSecure() -> HTTPUtilties.isSecureChannel()

Properties.* -> EncryptedProperties.*

ServletContext.log() -> Logger.*

java.security and javax.crypto -> Encryptor.*

java.net.URLEncoder/Decoder -> Encoder.encodeForURL/decodeForURL

java.sql.Statement.execute -> PreparedStatement.execute

ServletResponse.encodeURL -> HTTPUtilities.safeEncodeURL (better not to use at all)

ServletResponse.encodeRedirectURL -> HTTPUtilities.safeEncodeRedirectURL (better not to use at all)

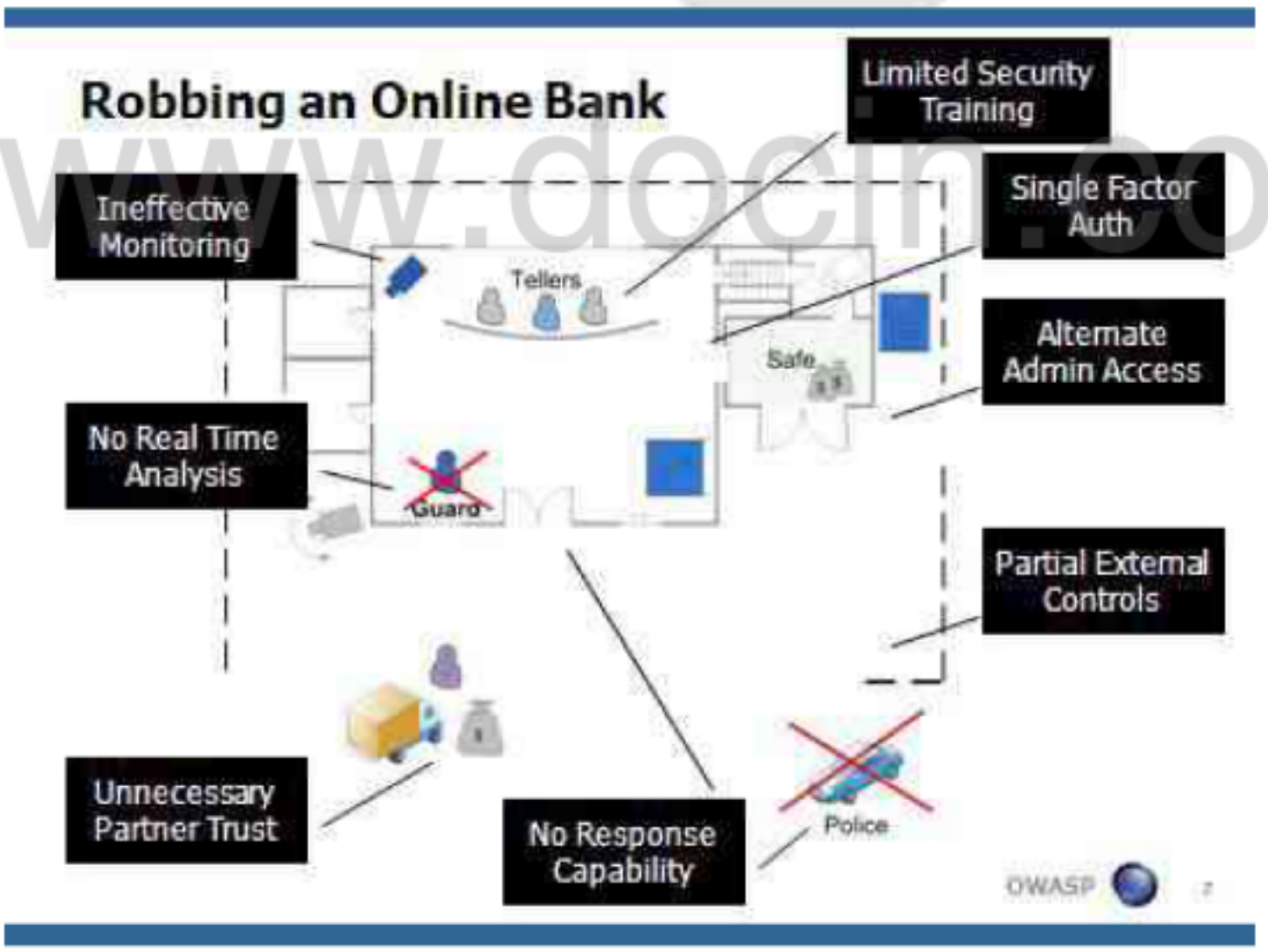
# 主动防御

## OWASP APPSENSOR

https://www.owasp.org/index.php/Category:OWASP_AppSensor_Project#tab=Overview

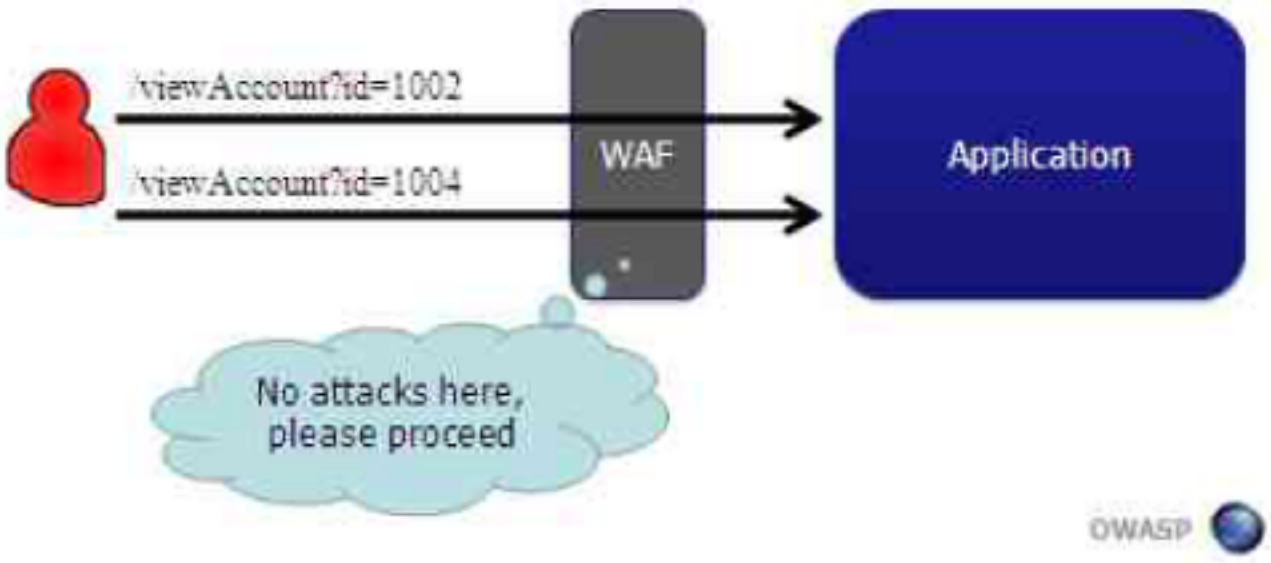
If you walk into a bank and try opening random doors, you will be identified, led out of the building and possibly arrested. However, if you log into an online banking application and start looking for vulnerabilities no one will say anything. This needs to change!

https://code.google.com/p/appsensor/



## What about a WAF?

- WAFs don't understand application context
- Custom application + Generic Solution != success






## Example Detection Point

| ACE1 | Modifying URL Arguments Within a GET For Direct Object Access Attempts |
|---|---|
| Exception Type | AccessControlException |
| Description | The application is designed to use an identifier for a particular object, such as using category ID=4 or user=guest within the URL. A user modifies this value in an attempt to access unauthorized information. This exception should be thrown anytime the identifier received from the user is not authorized due to the identifier being nonexistent or the identifier not authorized for that user. |
| Considerations | |
| Example(s) | The user modifies the following URL from site.com/viewpage?page=1&user=guest to site.com/viewpage?page=12&user=admin |

```
String action = "viewAccount";
String targetID=Utility.safeGetParam("ID", request);
if (authorized(action,targetID){
        //perform action }
else{
 AppSensor.log (ACE1,user,request);
  throw new AccessControlException();
}
```

# 参考

OWASP JAVA Project
https://www.owasp.org/index.php/Category:OWASP_Java_Project
https://www.owasp.org/index.php/Category:Java

《Web 应用安全威胁与防治--基于 OWASP Top 10 与 ESAPI》