Java Web Application Security

Matt Raible
http://raibledesigns.com
@mraible

Photos by **Trish McGinity** - **http://mcginityphoto.com**

Who is **Matt Raible**?

Father, Skier, Mountain Biker, Whitewater Rafter

Web Framework Connoisseur

Founder of AppFuse

Bus Lover

Blogger on raibledesigns.com

Why am I here?

Purpose

To explore Java webapp security options and encourage you to be a security expert

Goals

Show how to implement Java webapp security

Show how to penetrate a Java webapp

Show how to fix vulnerabilities

What about **YOU**?

Why are you here?

Do you care about Security?

Have you used Java EE 7, Spring Security or Apache Shiro?

What do you want to get from this talk?

Session Agenda

Security Development

- Java EE, Spring Security, Apache Shiro
- SSL and Testing

Verifying Security

- OWASP Top 10 & Zed Attack Proxy

Tools and Services

Action!

Develop

Java EE 7

Security constraints defined in web.xml

- web resource collection - URLs and methods
- authorization constraints - role names
- user data constraint - HTTP or HTTPS

User Realm defined by App Server

Declarative or Programmatic Authentication

Annotations Support


Java EE 7 Demo

Servlet 3.0

HttpServletRequest

- authenticate(response)
- login(user, pass)
- logout()
- getRemoteUser()
- isUserInRole(name)

Servlet 3.0 and JSR 250

Annotations

- @ServletSecurity
- @HttpMethodConstraint
- @HttpConstraint
- @RolesAllowed
- @PermitAll

Servlet 3.1

Non-blocking I/O

HTTP protocol upgrade mechanism

Security

- Run-as security roles to #init and #destroy
- Session Fixation protection
- Deny HTTP methods not explicitly covered by security constraints

JSR 375: Java EE Security API

Improvements to:

- User Management
- Password Aliasing
- Role Mapping
- Authentication
- Authorization

Learn more on InfoQueue

Java EE Limitations

No error messages for failed logins

No Remember Me

Container has to be configured

Doesn’t support regular expressions for URLs

Spring Boot with Security

Basic Authentication by default

Fluent API for defining URLs, roles, etc.

Spring MVC Test with Security Annotations

Password Encoding

Remember Me

WebSocket Security

Programmatic API

spring

by Pivotal™


Spring Security Demo

Spring Security JavaConfig

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.*;
import org.springframework.security.config.annotation.authentication.builders.*;
import org.springframework.security.config.annotation.web.configuration.*;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
            .withUser("user").password("password").roles("USER");
    }
}
```

Enabling Spring Security Annotations

XML Config:

```
<global-method-security pre-post-annotations="enabled"/>
```

Java Config:

```
@EnableGlobalMethodSecurity(prePostEnabled=true)

@EnableGlobalMethodSecurity(jsr250Enabled=true)

@EnableGlobalMethodSecurity(secureEnabled=true)
```

Spring Security @PreAuthorize

```
@PreAuthorize("hasRole('ROLE_USER')")
public void create(Contact contact);

@PreAuthorize("hasPermission(#contact, 'admin')")
public void deletePermission(Contact contact, Sid recipient, Permission permission);

@PreAuthorize("#contact.name == authentication.name")
public void doSomething(Contact contact);

@PreAuthorize("hasRole('ROLE_USER')")
@PostFilter("hasPermission(filterObject, 'read') or hasPermission(filterObject, 'admin')")
public List<Contact> getAll();
```

Spring Security @Secured

```
@Secured("IS_AUTHENTICATED_ANONYMOUSLY")
public Account readAccount(Long id);

@Secured("IS_AUTHENTICATED_ANONYMOUSLY")
public Account[] findAccounts();

@Secured("ROLE_TELLER")
public Account post(Account account, double amount)}
```

Spring MVC Test with Security

```
import static org.springframework.security.test.web.servlet.setup.SecurityMockMvcConfigurers.*;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
@WebAppConfiguration
public class CsrfShowcaseTests {

    @Autowired
    private WebApplicationContext context;

    private MockMvc mvc;

    @Before
    public void setup() {
        mvc = MockMvcBuilders
                .webAppContextSetup(context)
                .apply(springSecurity())
                .build();
    }
}
```

Spring Security Test Annotations

```
@WithMockUser // user:password,roles="ROLE_USER"

@WithMockUser(username="admin",roles={"USER","ADMIN"})

@WithUserDetails

@WithSecurityContext
```

Spring Limitations

Authentication mechanism in WAR

Securing methods only works on Spring beans

Apache Shiro

Filter defined in WebSecurityConfig

URLs, Roles can be configured in Java

Or use shiro.ini and load from classpath

[main], [urls], [roles]

Cryptography

Session Management


Apache Shiro Demo

Shiro Limitations

Limited Documentation

Getting Roles via LDAP not supported

No out-of-box support for Kerberos

REST Support needs work

Stormpath

Authentication as a Service

Authorization as a Service

Single Sign-On as a Service

A User Management API for Developers

https://stormpath.com

Stormpath with Spring Boot

```
<dependency>
    <groupId>com.stormpath.spring</groupId>
    <artifactId>spring-boot-starter-stormpath-thymeleaf</artifactId>
    <version>1.0.RC4.5</version>
</dependency>
```

/register

/login

/logout

Includes Forgot Password

Add CORS Support

```
public class OptionsHeadersFilter implements Filter {

    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
            throws IOException, ServletException {
        HttpServletResponse response = (HttpServletResponse) res;

        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "GET,POST");
        response.setHeader("Access-Control-Max-Age", "360");
        response.setHeader("Access-Control-Allow-Headers", "x-requested-with");
        response.setHeader("Access-Control-Allow-Credentials", "true");

        chain.doFilter(req, res);
    }

    public void init(FilterConfig filterConfig) {
    }

    public void destroy() {
    }
}
```

Global CORS in Spring Boot 1.3

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
public class MyConfiguration {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/api/**");
            }
        };
    }
}
```

Securing a REST API

Use Basic or Form Authentication

Use OAuth 2

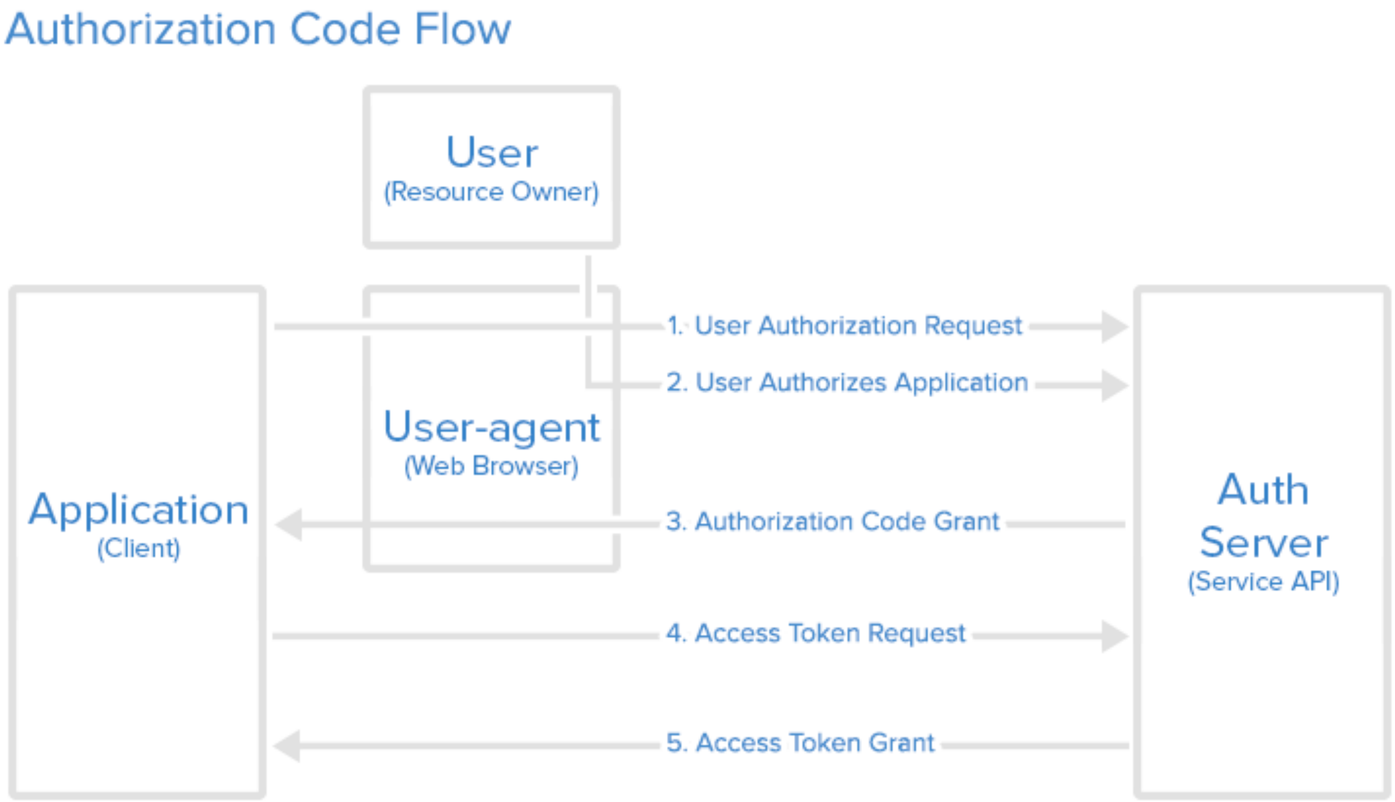
Use JSON Web Tokens (JWT)

Use Developer Keys

Use an API Management Platform

What have you used?

OAuth 2



https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

JHipster

http://jhipster.github.io/

Greetings, Java Hipster!

JHipster is a
Yeoman generator,

used to create a
Spring Boot + **AngularJS**
project.

JHipster Security

Improved Remember Me

Cookie theft protection

CSRF protection

Authentication

- HTTP Session
- Token-based
- OAuth2
- Social (Facebook, Google, Twitter)

JHipster HTTP Session

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true, securedEnabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Inject
    private AjaxAuthenticationSuccessHandler ajaxAuthenticationSuccessHandler;

    @Inject
    private AjaxAuthenticationFailureHandler ajaxAuthenticationFailureHandler;

    @Inject
    private AjaxLogoutSuccessHandler ajaxLogoutSuccessHandler;
```

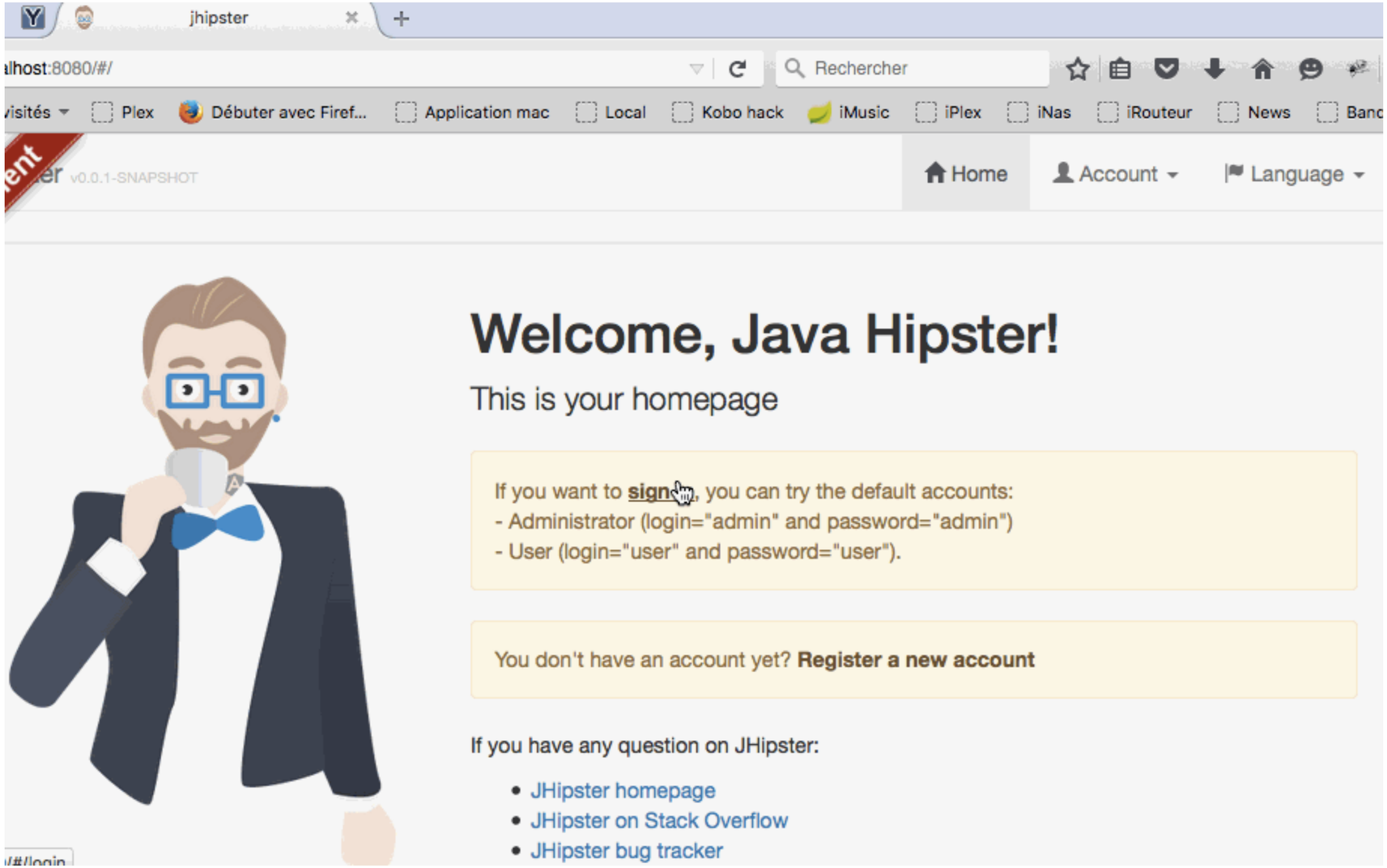
JHipster Token-based

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .exceptionHandling()
        .authenticationEntryPoint(authenticationEntryPoint)
    .and()
        .csrf().disable().headers().frameOptions().disable()
    .and()
        .sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
    .and()
        .authorizeRequests()
        .antMatchers("/api/register").permitAll()
        // additional rules for URLs
    .and()
        .apply(securityConfigurerAdapter());

}

private XAuthTokenConfigurer securityConfigurerAdapter() {
    return new XAuthTokenConfigurer(userDetailsService, tokenProvider);
}
```

JHipster OAuth2

```
@Configuration
public class OAuth2ServerConfiguration {

    @Configuration
    @EnableResourceServer
    protected static class ResourceServerConfiguration
        extends ResourceServerConfigurerAdapter {

    }

    @Configuration
    @EnableAuthorizationServer
    protected static class AuthorizationServerConfiguration
        extends AuthorizationServerConfigurerAdapter
        implements EnvironmentAware {

    }
}
```

JHipster Social



API Security Projects

Spring Security OAuth - version 2.0.8

Spring Social - version 1.1.4

- Facebook, Twitter, LinkedIn, TripIt,
- and GitHub Bindings

Penetrate

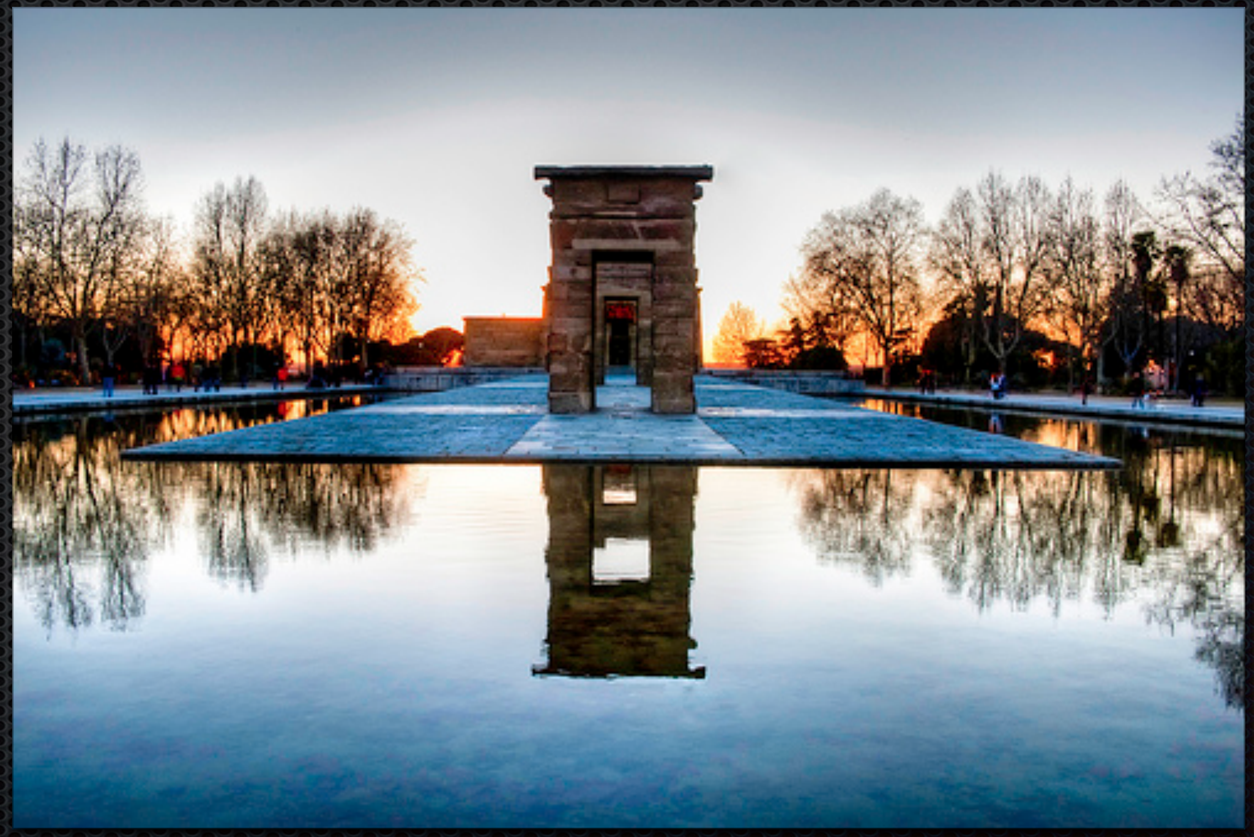
OWASP Testing Guide and Code Review Guide

OWASP Top 10

OWASP Zed Attack Proxy

Burp Suite

OWASP WebGoat



OWASP

The Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software.

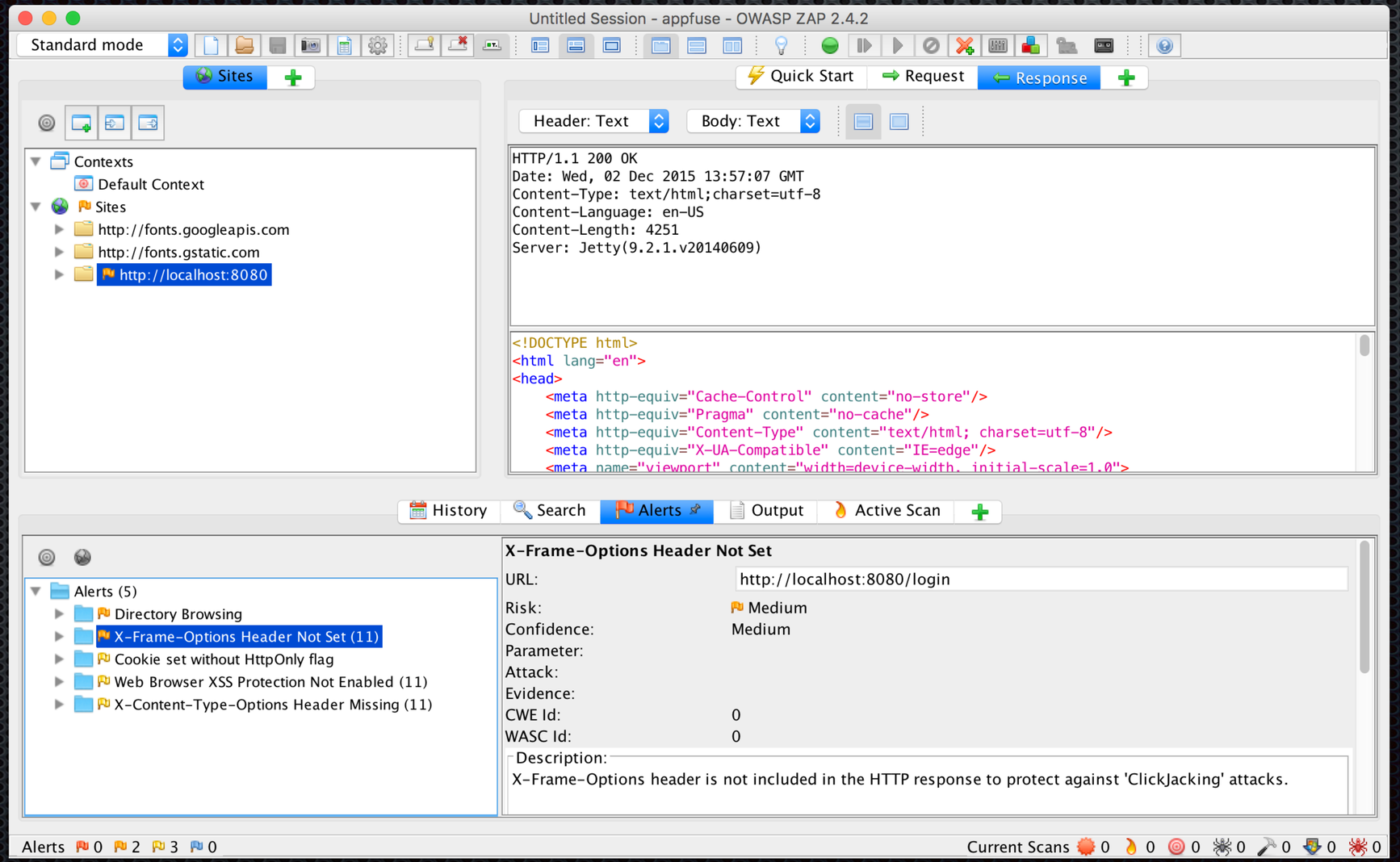
At OWASP you'll find free and open ...

Application security tools, complete books, standard security controls and libraries, cutting edge research

http://www.owasp.org

OWASP
The Open Web Application Security Project

Penetration Testing with ZAP



Fixing ZAP Vulnerabilities

```
<session-config>
    <session-timeout>15</session-timeout>
    <cookie-config>
        <http-only>true</http-only>
        <secure>true</secure>
    </cookie-config>
    <tracking-mode>COOKIE</tracking-mode>
</session-config>
```

```
<form action="${ctx}/j_security_check" id="loginForm"
      method="post" autocomplete="off">
```

7 Security (Mis)Configurations in web.xml

1. Error pages not configured
2. Authentication & Authorization Bypass
3. SSL Not Configured
4. Not Using the Secure Flag
5. Not Using the HttpOnly Flag
6. Using URL Parameters for Session Tracking
7. Not Setting a Session Timeout

http://software-security.sans.org/blog/2010/08/11/security-misconfigurations-java-webxml-files

OWASP Top 10 for 2013

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration

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

OWASP Top 10 for 2013

6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forwards

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

Protect

[SWAT] Checklist

Firewalls

IDS and IPS

Audits

Penetration Tests

Code Reviews with Static Analysis Tools

[SWAT] Checklist

http://software-security.sans.org/resources/swat

Securing Web Application Technologies
[SWAT] Checklist

The SWAT Checklist provides an easy to reference set of best practices that raise awareness and help development teams create more secure applications. It's a first step toward building a base of security knowledge around web application security. Use this checklist to identify the minimum standard that is required to neutralize vulnerabilities in your critical applications.

- ERROR HANDLING AND LOGGING
- DATA PROTECTION
- CONFIGURATION AND OPERATIONS
- AUTHENTICATION
- SESSION MANAGEMENT
- INPUT AND OUTPUT HANDLING
- ACCESS CONTROL

Firewalls

Stateless Firewalls

Stateful Firewalls

- Invented by Nir Zuk at Check Point in the mid-90s

Web App Firewalls

- Inspired by the 1996 PHF CGI exploit

Gartner on Firewalls

Content Security Policy

An HTTP Header with whitelist of trusted content

Bans inline <script> tags, inline event handlers and javascript: URLs

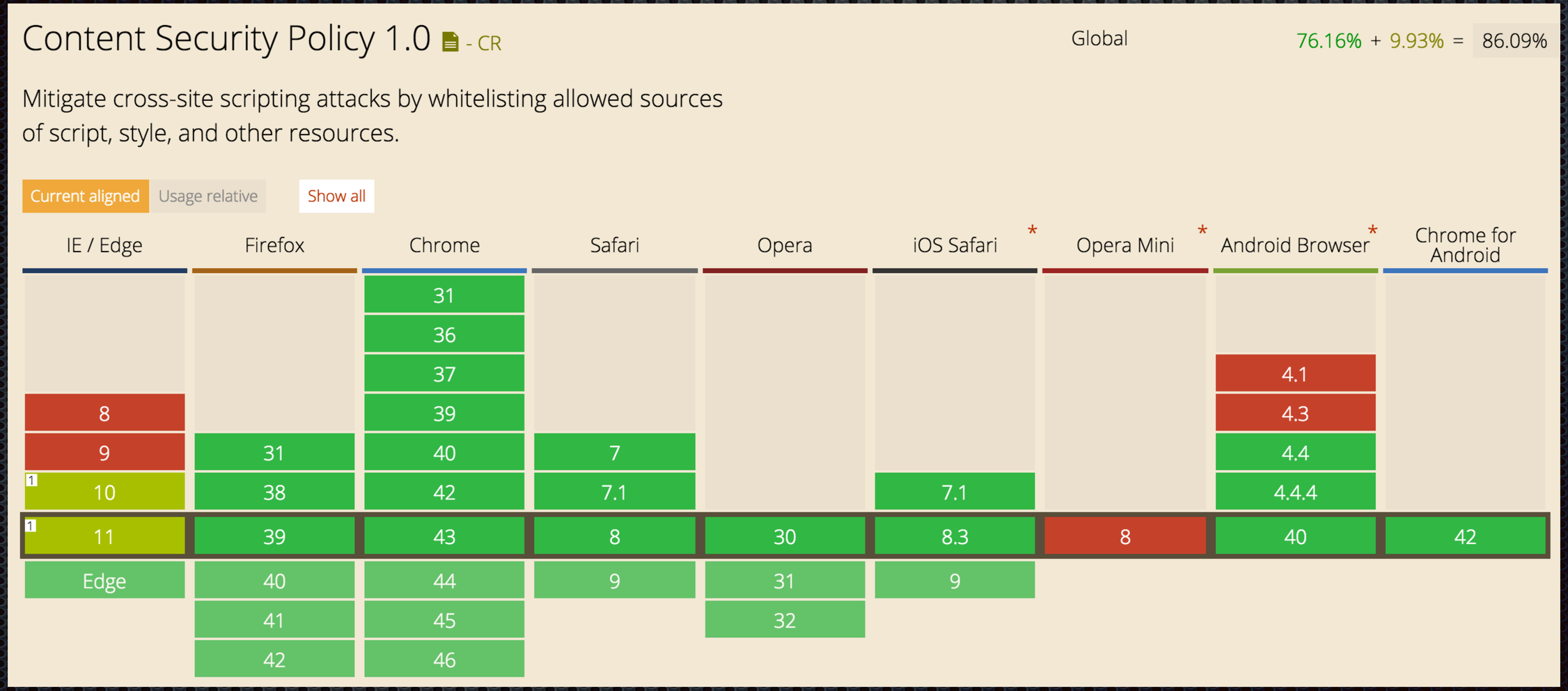
No eval(), new Function(), setTimeout or setInterval

Supported in Chrome 16+, Safari 6+, and Firefox 4+, and (very) limited in IE 10

Content Security Policy

```
Content-Security-Policy: script-src 'self' https://apis.google.com
```

Content Security Policy: Can I use?



Relax

Web App Firewalls: Imperva, F5

Open Source: WebNight and ModSecurity

Stateful Firewalls: Palo Alto, Check Point, Juniper

IDP/IPS: Sourcefire (Cisco), TippingPoint (HP)

Open Source: Snort

Audits: ENY, PWC, Grant Thornton

Pen Testing: Metasploit, Nessus, Veracode, Burp Suite

Open Source: OWASP ZAP

Remember...

"Security is a quality, and as all other quality, it is important that we build it into our apps while we are developing them, not patching it on afterwards like many people do."

-- *Erlend Oftedal*

From a comment on raibledesigns.com: http://bit.ly/mjufjR

Action!

Use OWASP and Open Source Security Frameworks

Follow the Security Street Fighter Blog

http://software-security.sans.org/blog

Use OWASP ZAP to pentest your apps

Don’t be afraid of security!

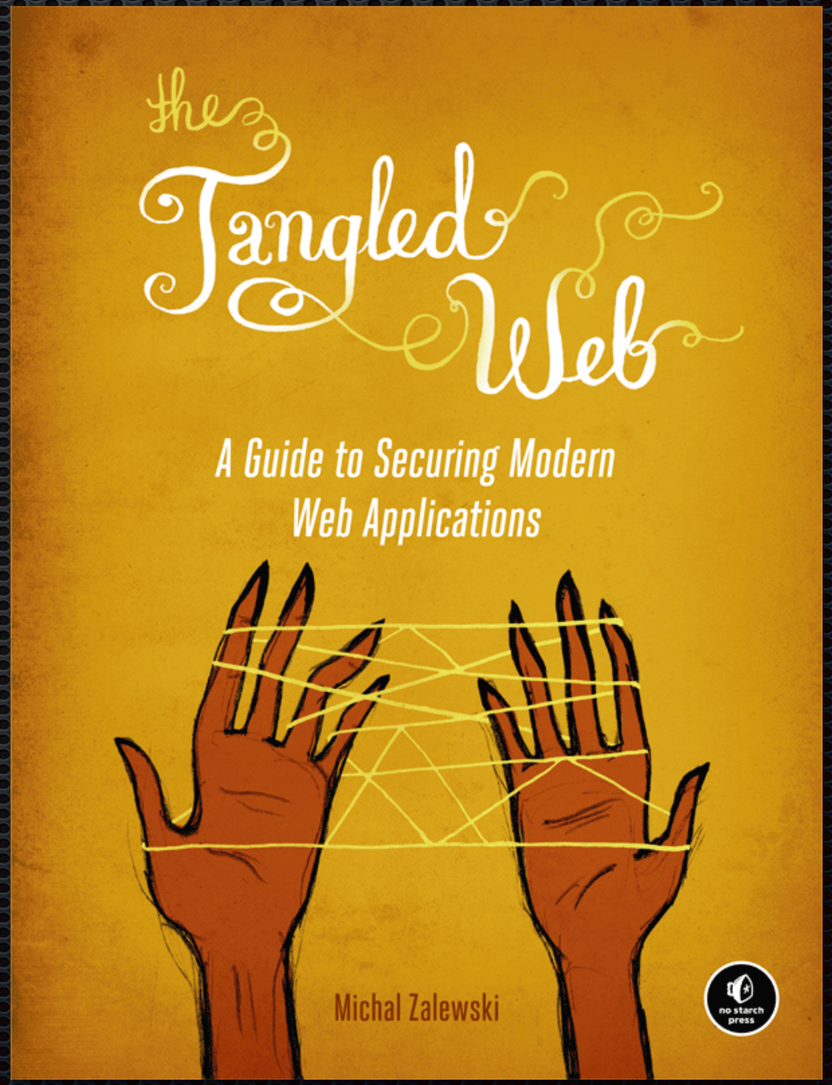
Additional Reading

Securing a JavaScript-based Web Application

http://eoftedal.github.com/WebRebels2012

Michal Zalewski's "The Tangled Web"

http://lcamtuf.coredump.cx/tangled



Questions?

Keep in touch!

- http://raibledesigns.com
- @mraible

Presentations

- http://slideshare.net/mraible

Code

- https://github.com/mraible/java-webapp-security-examples