USS “Enterprise” - сквозь черные дыры безопасности Java Web-контейнеров

Михаил Дударев

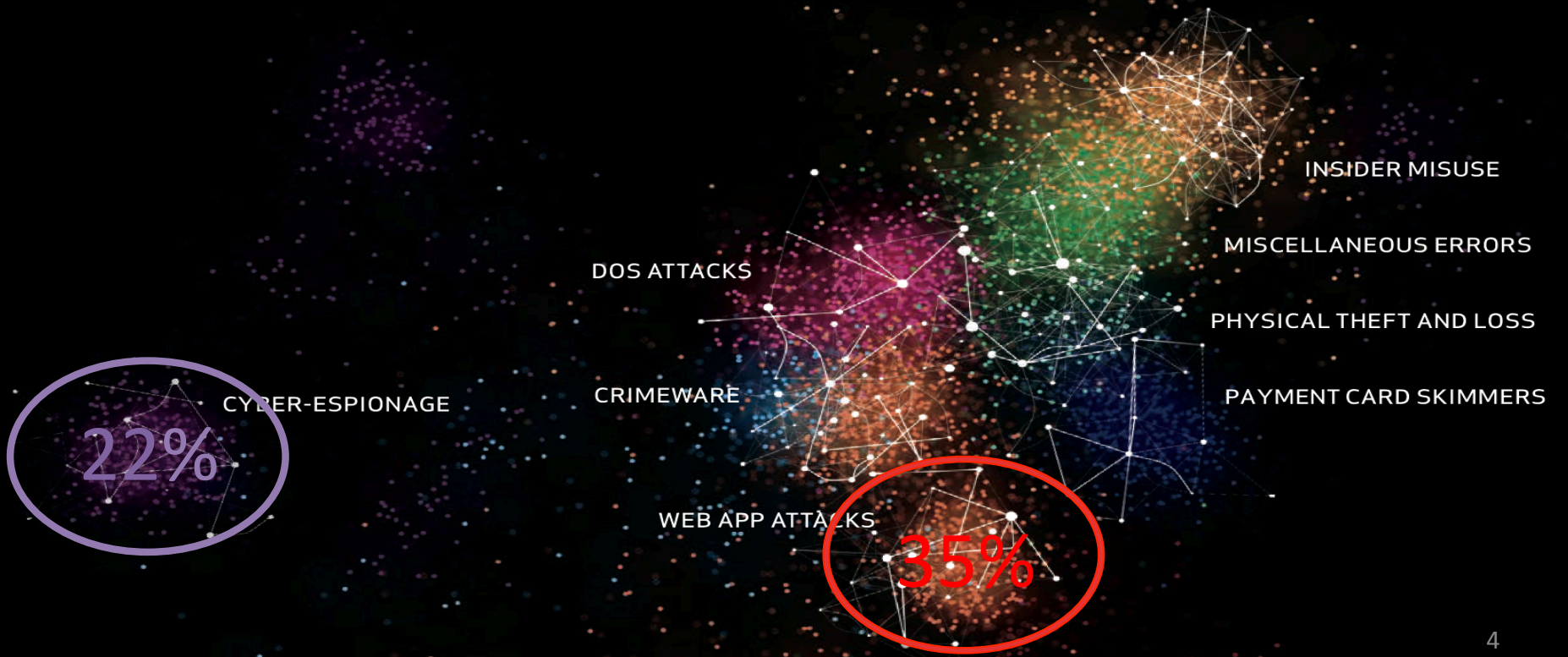
Licel, 2014

О нас

- Михаил Дударев, работает с Java Security более 15 лет, основатель проекта jCardSim, Java Card симулятора, завоевавшего награду Duke's Choice Award 2013, со-основатель компании Licel.

- Licel занимается созданием решений для защиты программного от нелегального использования и модификации третьими лицами для Java и Android платформ.

2014 DATA BREACH INVESTIGATIONS REPORT



Популярные атаки

1. XSS
2. Encyption
3. Access Control
4. SQL Injection
5. CSRF

- **O**pen **W**eb **A**pplication **S**ecurity **P**roject
  - Некоммерческая организация, основанная в 2001 году
- Документы
  - Top 10 Security Risks
  - Guidelines
  - Cheat sheets
- Библиотеки и инструменты
  - Zed Attack Proxy (ZAP)
  - App Sensor
  - Enterprise Security API (ESAPI)
- Обучающие среды
  - WebGoat

OWASP Top 10 2013

A1 Injection

A2 Broken Authentication and Session Management

A3 Cross-Site Scripting (XSS)

A4 Insecure Direct Object Reference

A5 Security Misconfiguration

A6 Sensitive Data Exposure

A7 Missing Function Level Access Control

A8 Cross-Site Request Forgery (CSRF)

A9 Using Known Vulnerable Components

A10 Unvalidated Redirects and Forwards

Атаки на Web-приложения

- Атаки на сессию
  - A2 Broken Authentication and session management
  - A6 Sensitive Data Exposure
- Атаки на клиентской стороне
  - A3 Cross-Site Scripting (XSS)
  - A8 Cross-Site Request Forgery (CSRF)
- Получение неавторизированного доступа
  - A7 Missing Level Function Access Control
  - A4 Insecure Direct Object Reference
- Атаки на серверную логику
  - A1 Injections
  - A10 Unvalidated redirects and forwards

Манипуляции с сессией (A2)

- Кража информации о сессии
  - URL, sniffing, logs, XSS
- Предсказывание сессии
- Фиксация сессии
  - Плагины для браузеров
  - Прокси (например OWASP ZAP)

Раскрытие сессии (A6)

- Транспорт
  - Незашифрованное соединение
- Клиент
  - XSS
  - Атаки на браузер/ОС
- Сервер
  - Логи
  - Репликация сессии
  - Дамп памяти

Cross-Site Scripting XSS (A3)

- Возникает в случае отображения “вредных” данных web-приложением
  - Недоверенный код исполняется приложением
  - Выполнение “вредного” кода при отображении, без предварительной валидации
- Основные типы
  - Сохраненные (Stored)
  - Наведенные (Reflected)

CSRF (A8)

Альфа Центавра IV

2) Сайт злоумышленника

1) Аутентификация

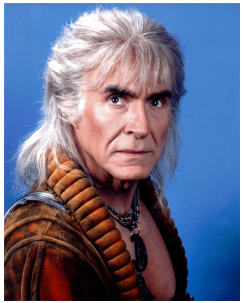
Земля

3) Страница с CSRF-кодом

```
<img src="http://earth/adminHandling.jsp?
operation=Add&status=Commander&user=Knah>
```

4) Браузер отсылает запрос

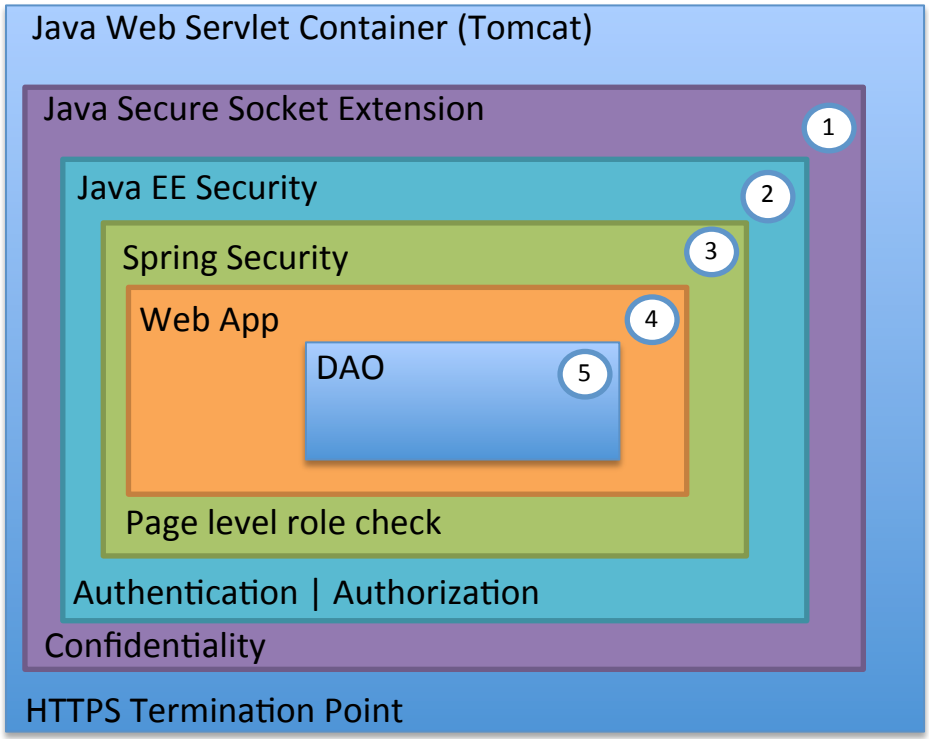
```
GET /adminHandling.jsp HTTP/1.1
Cookie: <earth authentication cookie>
```



Injections (A1)

- Возникает в случае отправки приложением, неверифицированных данных интерпретатору
  - SQL Injection
  - XSS
  - ORM Injection
  - NoSql Injection
  - Xpath Injection
  - JSON Injection
  - Cmd Injection

Java Web Application Security



Tomcat

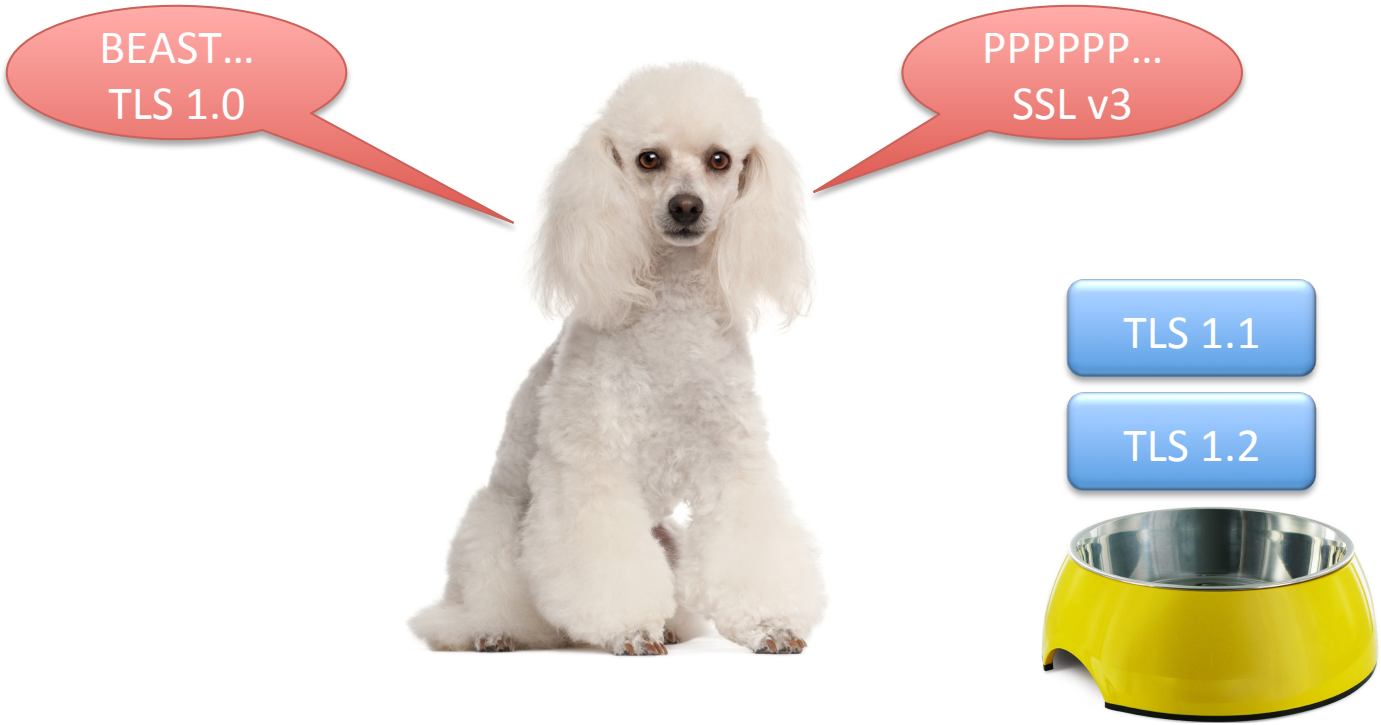
- Security Misconfiguration (A5)
  - Запуск с правами Администратора
  - Запуск Tomcat без Security Manager
  - Публичный доступ к Manager Web App и внутренним Web App
  - Публичный доступ к shutdown-порту
  - Отображение ошибок компиляции

Tomcat

- Using Known Vulnerable Components (A9)

http://www.cvedetails.com

One more… Double Demo


BEAST…
TLS 1.0
PPPPPP…
SSL v3
TLS 1.1
TLS 1.2

JSSE

```
<Connector port="443"
   protocol="org.apache.coyote.http11.Http11NioProtocol”
  SSLEnabled="true”
  scheme="https" secure="true”
  clientAuth="false”
  sslProtocol="TLSv1.2”
  sslEnabledProtocols="TLSv1.2”
  ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
           …" />
```

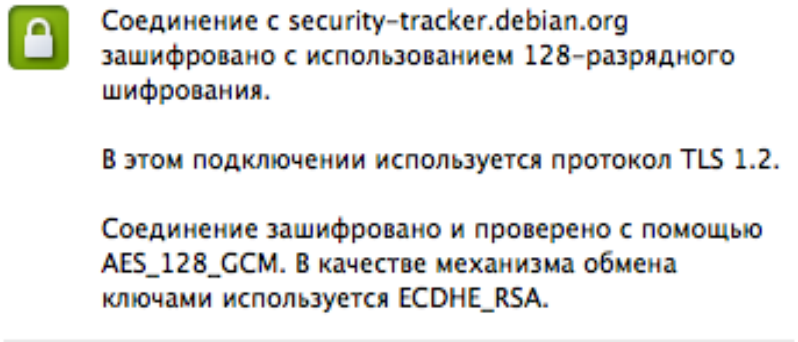
SSL, TLS

- SSL предшественник TLS
- TLS 1.0 апгрейд SSL 3.0 (1999)
- Уязвимые версии SSL v2, SSL v2, TLS 1.0
- Текущая версия TLS 1.2

IE	Firefox	Chrome	Safari	Opera	iOS Safari
11	32	38	7.1	24	8
	33	39	8	25	
	34	40		26	
	35	41			

ECDH_RSA_WITH_AES_128_GCM_SHA256 ???

- Алгоритм согласования ключей (ECDH/ECDHE)
- Ключевой алгоритм (RSA)
- WITH_(Алгоритм шифрования)_(длина в битах)_(режим шифрования)
  - Рекомендуется AES
  - Минимум 128 бит
  - GCM/CBC
- Алгоритм хеш-функции
  - Минимум SHA-256

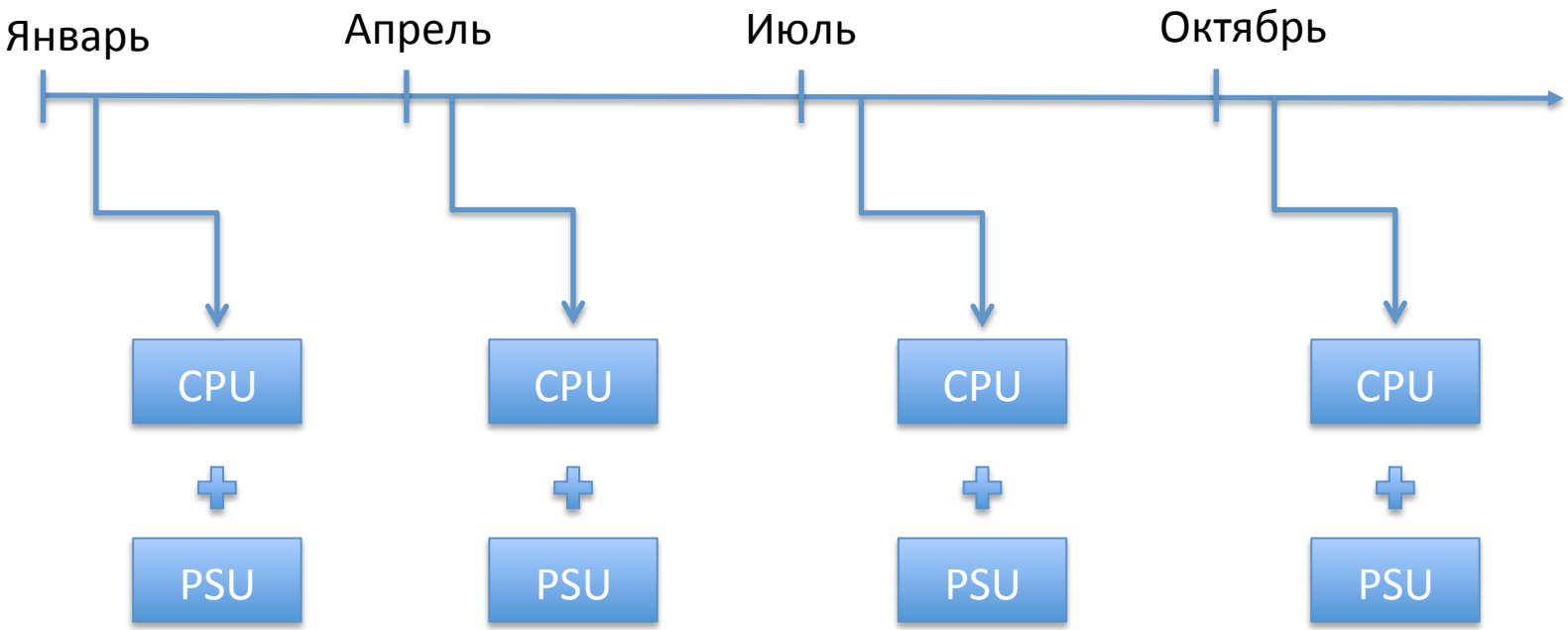


New Java 8 Security Features

- TLS Server Name Indication (SNI) Extension (виртуальные адреса)
- Поддержка Authenticated AES GCM режима шифрования
- Набор EC-ключей NSA Suite B
- PKCS#11 Crypto Provider для 64-bit Windows
- Аппаратное ускорение для AES
- Полнофункциональное PKCS#12 хранилище ключей

And One More Thing…

Начиная с октября 2014



Коммандер Спок рекомендует

1. Content Security Policy

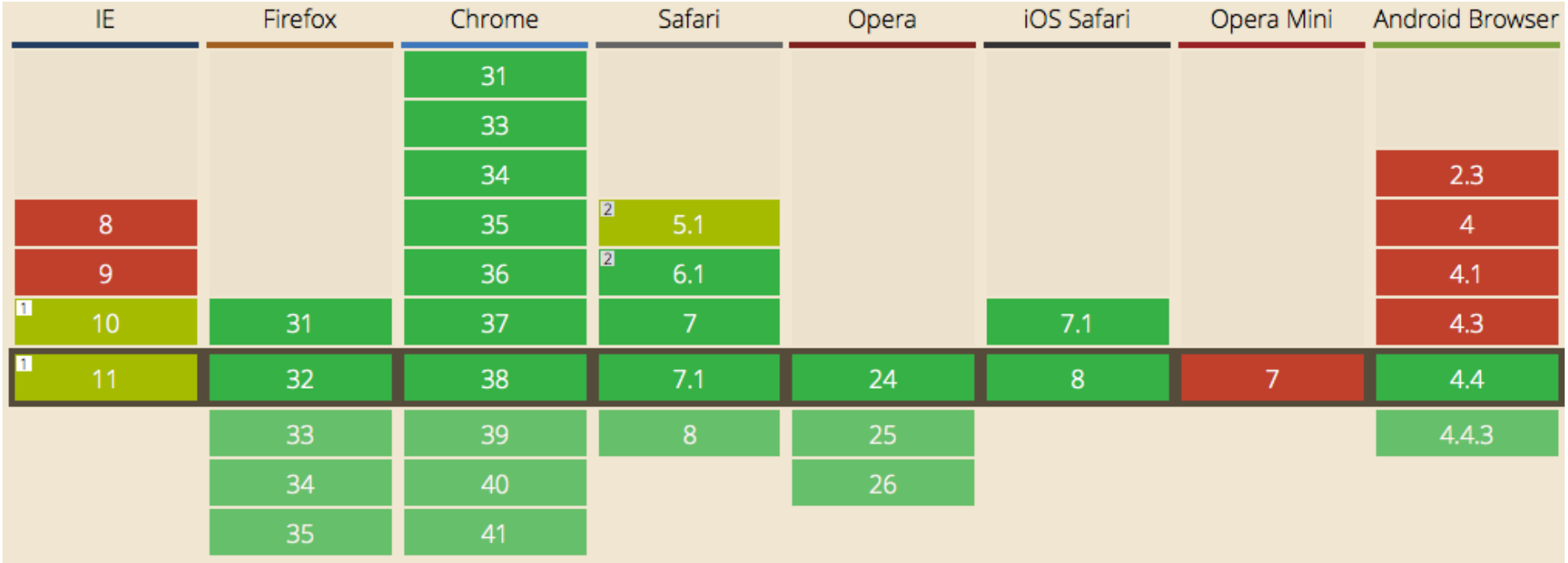
- W3C Candidate Recommendation
  - `Content-Security-Policy`
  - `Content-Security-Policy-Report-Only`
- Определяет список разрешенных источников
- Директивы

```
default-src
script-src
object-src
style-src
img-src
```

```
media-src
frame-src
font-src
connect-src
```

```
report-uri
```

1. Content Security Policy



http://caniuse.com/#feat=contentsecuritypolicy

2. Encryption

- Включение SSL в web.xml

```
<security-constraint>
  ...
  <user-data-constraint>
    <transport-guarantee>
      CONFIDENTIAL
    </transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

2. Secure cookie

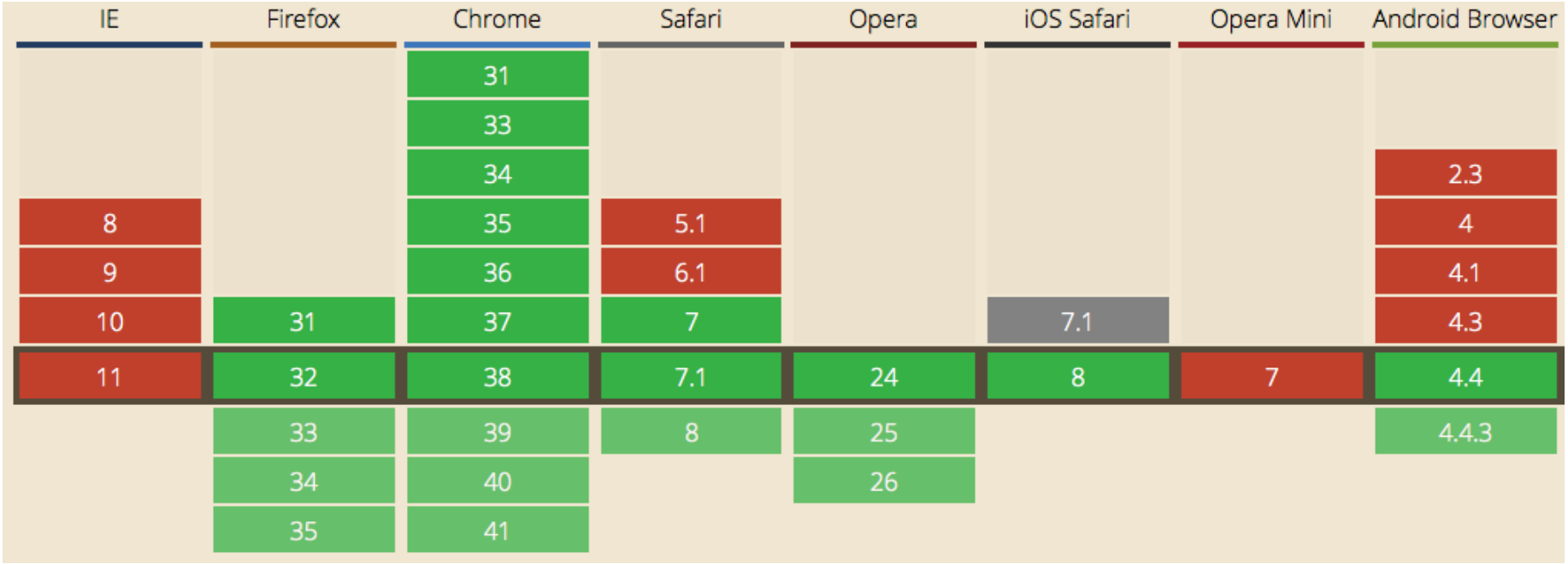
- Убедитесь в том, что Cookie’s могут передаваться/ использоваться только через SSL + HttpOnly
- Через web.xml (Servlet 3.0)

```
<session-config>
    <cookie-config>
        <secure>true</secure>
        <http-only>true</http-only>
    </cookie-config>
</session-config>
```

2. HTTP Strict-Transport-Security (HSTS)

- Определен в RFC 6797
  - `Strict-Transport-Security: max-age=`*`seconds`*`; includeSubdomains` (seconds – min 18 недель)
- Браузер работает по следующим правилам
  - Автоматический редирект всех HTTP-запросов на HTTPS (http://example.com -> https://example.com)
  - Если сертификат сайта ошибочен, или самозаверен, пользователю запрещается работать с сайтом

2. HTTP Strict-Transport-Security



http://caniuse.com#feat=strictransportsecurity

3. Access Control

- Избегайте указания <http-method> в <security-constraint>

```
<security-constraint>
    <web-resource-collection>
            <url-pattern>/site/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
    </web-resource-collection>
  ...
</security-constraint>
```

4. SQL/ORM Injection

- Используйте PreparedStatement с обязательным указанием всех параметров через ‘?’

```
String query = "SELECT * FROM users
  WHERE age > " + age + " AND gender = '" + gender + "'";

PreparedStatement stmt =  con.prepareStatement(query);
ResultSet rs = stmt.executeQuery();
```

5. CSRF

- Необходимо включать что-то случайное в запрос
  - anti-CSRF токен
- JSF 2.2
  - HTTP POST: **javax.faces.ViewState** hidden field со случайным токеном
  - HTTP GET: **protected-views** в WEB-INF/faces-config.xml
  - Новый **javax.faces.Token** URL параметр
- Tomcat 6/7/8 (org.apache.catalina.filters.CsrfPreventionFilter)

5. OWASP CSRF Guard

- Может включать anti-CSRF токены, используя:
  - Работу с JavaScript DOM – автоматическая защита с минимальными усилиями
  - JSP Tag library – для тонкой ручной защиты
- Фильтр для перехватки запросов и валидации токенов

5. OWASP CSRF Guard

```
XMLHttpRequest.prototype._open = XMLHttpRequest.prototype.open;
XMLHttpRequest.prototype.open = function(method, url, async, user, pass) {
  // store a copy of the target URL
  this.url = url;
  this._open.apply(this, arguments);
}

XMLHttpRequest.prototype._send = XMLHttpRequest.prototype.send;!
XMLHttpRequest.prototype.send = function(data) {
  if(this.onsend != null) {
    // call custom onsend method to modify the request
    this.onsend.apply(this, arguments);
}
  this._send.apply(this, arguments);
}
```

Ссылки

https://www.owasp.org

https://shiro.apache.org

Контакты

- Email: dudarev@licel.ru
- Twitter: @MikhailDudarev
- Web: http://licelus.com