ANZTB TEST 2013
ADVANCING EXPERTISE IN SOFTWARE TESTING

Lightning talks

Andrew Muller, Canberra
Managing Director, Ionize, Canberra

The challenges of Security Testing

Advancing Expertise in Software Testing

Security Testing

Taming the Wild West

Who is this guy?

- Andrew Muller (Ionize)
- IT Security consultant to the stars
- Chapter Leader of Canberra OWASP
- Project Leader of OWASP Testing Guide
- Member of IT-012-04 Security Techniques
- Member of IT-015-26 Software Testing

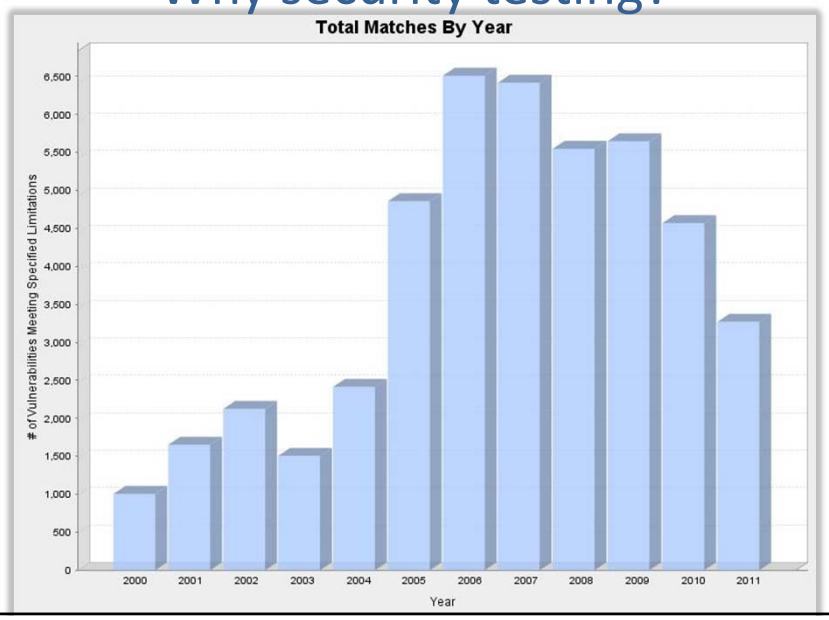
What's he talking about?

- Background
- Issues
- Methodologies
- Tools
- Common vulnerabilities

Why security testing?

Why security testing?

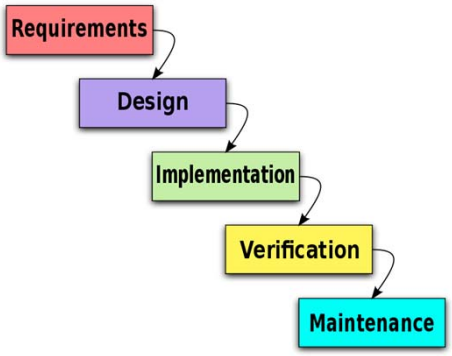
Why security testing?

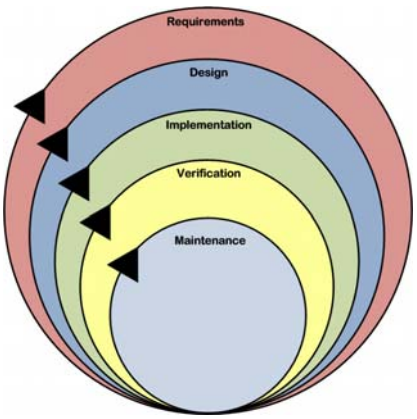


Why is this happening?

- Functional requirements are **KING**
- Security is a non-functional requirement
  – and diminishes usability
- Some of the dysfunction is cultural

How do we fix it?



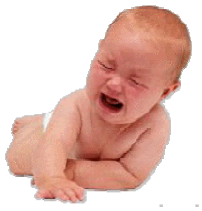


Where does security fit?

Perception of Security

Perception of Developers

Perception of Business



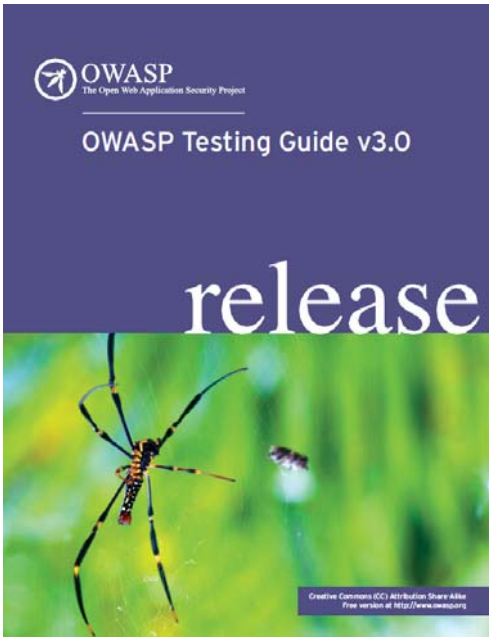
Security testing methodologies

- Most of us have a robust test cycle
- System and integration testing is methodical
- Security testing hasn't been so rigourous
- Security is just another set of test cases

Security testing methodologies

- What do we want from a methodology?

- Accountability
  - What was tested? Where are the results?
- Repeatability
  - Will the next test be the same as this one?
- Thoroughness
  - How do I know everything has been tested?



Tools

- Several HTTP intercepting proxies
  - Webscarab
  - Paros
  - BurpSuite
  - Tamperdata (Firefox plugin)

Test Automation

- There are bunch of commercial tools
  - IBM AppScan
  - HP's WebInspect
  - Aspect's Acunetix
- And free tools
  - Burp Suite (also offers commercial version)
  - Paros
  - Nikto
- So which one is the best?

Test Automation

- SAMATE - Software Assurance Metrics And Tool Evaluation (http://samate.nist.gov)
- Found that the best tools find 33% of software vulnerabilities.
- **All of the tools together could detect 50% of software vulnerabilities.**

Sorry, how much?

50%

OWASP Top 10 - 2010

Injection Flaws	Injection occurs when user-supplied data is sent to an interpreter as part of a command or query.
Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content.
Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected.
Insecure Direct Object Reference	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter.
Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker.
Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform.
Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials.
Failure to Restrict URL Access	Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users.
Insufficient Transport Layer Protection	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.
Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages

Reflected Cross Site Scripting

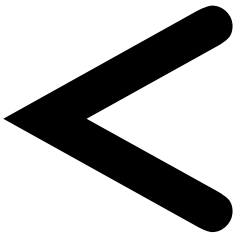
- An attack needs user interaction
- Identify parameter inputs that are reflected to the browser

```
<SCRIPT SRC=http://www.kimjong-un.com/ronery.js></SCRIPT>

<SCRIPT>alert(“xss”)</SCRIPT>

<IMG SRC=“javascript:alert(‘xss’);”>
```

Cross Site Scripting detection



Stored Cross Site Scripting

- Same as reflected but persistent
- "Watering hole" attack
  - Visit website and browser is compromised
  - Usual suspects are guestbooks and user comments

SQL injection

- Identify probable inputs to database queries
  - Login pages
  - Search pages

SQL injection detection

SQL injection detection

- Detection is as simple as ’
  - The inclusion of this query delimiter throws an exception

Standard SQL injection

Consider a typical authentication database query

```
SELECT * FROM Users WHERE Username='$username' AND
   Password='$password‘
```

Username input is `' or 1=1--`

Query becomes

```
SELECT * FROM Users WHERE Username='' or 1=1-- ' AND
   Password='$password‘
```

Query is always true, usually authenticating as the first record in the Users table

User enumeration

- Common/default usernames
  - admin, test, guest
- Guessable username structure
- Developer/tester accounts in production
- Error responses from login application
  - Invalid user
  - Invalid password

Bypass authentication schema

- Forced browsing
  - Only the login page verifies login status
- Parameter modification
  - http://server/login.asp?authenticated=no
- Session identifier prediction
- SQL injection

Bypass session management schema

- Session is typically implemented using cookies
- Cookie collection
  - What generates and consumes cookies?
- Cookie analysis
  - Session token structure and predictability
  - Cookie structure and lifetime
- Cookie manipulation

Session fixation

- Session cookie set upon accessing application
- New session cookie not set upon successful authentication
- Session cookie can be fixed using other vulnerabilities

Cross Site Request Forgery

- Trick an authenticated to execute a malicious action
  - Login as a valid user
  - Have user execute pregenerated request to perform function