Hybrid 2.0 – In search of the holy grail...

A Talk for OWASP BeNeLux
by
Roger Thornton
Founder/CTO Fortify Software Inc
Before we Begin:

• Expectations

• Objectives

• Agenda
About Your Presenter

• 22 years of Engineering (“building stuff”) in the Silicon Valley
  – Semiconductors
  – Operating Systems
  – Development Tools
  – Brokerage / E-Commerce

• The Last 6 years working on Securing that Stuff
  – Founder & CTO of Fortify Software
A Simple, Reasonable, Question….

If I run software, am I putting my business, data, customers or even life on earth at risk?

If so, how serious is the threat?

Unfortunately not so simple to answer…
## Three Basic Approaches

<table>
<thead>
<tr>
<th>Hire an expert</th>
<th>Automate Hacking</th>
<th>Analyze the Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethical Hacking</td>
<td>Black Box / Penetration Test</td>
<td>Static and Dynamic Analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hire an expert</th>
<th>Automate Hacking</th>
<th>Analyze the Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exactly what the bad guy does..</td>
<td>Cheap and easy way to find the most obvious issues</td>
<td>Look for root cause issues from the “inside out” – the code</td>
</tr>
<tr>
<td>- Hard to know if your “experts” are as good as the bad guy</td>
<td>- “Badness-ometer” limitations and issues</td>
<td>- Requires intimate access to the software</td>
</tr>
<tr>
<td>- Prohibitively expensive to do on a regular basis</td>
<td>- Automated crawler and web traffic analysis can yield</td>
<td>- Requires programming knowledge and expertise</td>
</tr>
<tr>
<td>- No advantage over the bad guys</td>
<td>- Identifies the result – not the root cause</td>
<td>- Exploitability information is not present as with other two.</td>
</tr>
<tr>
<td>- Identifies the result – not the root cause</td>
<td></td>
<td>- Identifies the root cause not the result</td>
</tr>
</tbody>
</table>
Software Security Assurance (SSA)

The management & prevention of security risks in software

Operation

Assess
Software for security vulnerabilities

Test

Remediate
Vulnerabilities found in software

Prevent
Software security vulnerabilities

Construction

© 2009 All Right Reserved Fortify Software Inc.
## Software Security Maturity

### Risk Awareness
- Vulnerability Assessment
- Proving the problem or meeting a basic regulatory requirement
  - An info-sec project
  - Generates awareness & support security initiatives
  - Consulting, PenTesting & some manual code review

  *Recurring cost that does not “fix” anything*

### Risk Reduction
- Analysis & Remediation
- Fixing security issues uncovered from assessments
  - Info-sec driven project with development support
  - Forces a rework of code
  - “Inside-out” Static and Dynamic Analysis required

  *Lowering risk but costs too high*

### Prevention
- Secure SDL & Software
- Secure the development and procurement lifecycle avoiding issues altogether
  - Info-sec-sponsored Development-led project
  - Requires significant organizational buy-in
  - Requires more than a point solution

  *Minimizing business risk systematically*
The Challenge

Immediate Problem
Existing Legacy Applications

Systemic Problem
Software Procurement & Development Cycle

Compliance & Regulatory Requirements

Assessment & remediation of existing software

Prevention of the introduction of new risk

in-house
outsourced
commercial
open source
Benefits of a “Hybrid” Approach

• A seamless flow from Assessment to Prevention
  – *Facilitates growth in maturity from assessment to prevention*

• Combined benefits at Testing phase - “Remediation Gap”
  – Application Testing & Software Analysis:
    • Rapid identification of high priority issues (DAST)
    • Precise description of root cause vulnerability in code (SAST)

• **Reduced time and costs to remediate vulnerabilities**
  ✓ By mapping each security issue to root cause in source code
    • Developers understand security findings – faster fixes
    • Security findings are more accurate – less research
    • Security findings are more comprehensive – less rework

© 2009 All Right Reserved  Fortify Software Inc.
Dynamic Application Security Testing
“Black Box”

Challenges

- **Visibility to “root cause”**…
  - It is called “Black Box”
  - 1 Issue may be indicative of many
  - Multiple issues may trace back to one problem

- **Communicating to developers**
  - URLs and hacking technique vs. code errors
  - Validating behavior (FP)

© 2009 All Right Reserved  Fortify Software Inc.
File: /apl/MyCheckout.do

Scheme: http

Parameter: name

Attack Request:
POST /apl/MyCheckout.do HTTP/1.1
Accept: */*
Referer: http://zero.webappsecurity.com:8080/apl/finalCheckout.do
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: zero.webappsecurity.com:8080
Content-Length: 134
Pragma: no-cache

Memo: 229; Auditor.SendAsynchronousRequest; Attack(CID:(null)); AS: /2,EID: 9722923-8c3-49c2-901d-70b156c12b43; ST: Audit; Attack; AT: Post; ParamManipulation; APD name: t(1,0); R: False; S: /1; SID: 92866DF8737ED61D3EF5B76BDD0BC063; PSID: E5722BFACD0D5D19D13F046756C87F; Connection: Keep-Alive
Cookie: JSESSIONID=597B11C7176117C71D49C80DA49CDD253; CustomCookie=WebInspector524e10/8-6773G771D3F3412548F33364C17ACD0177F

File: WEB-INF/jar/javv/com/orders/apl/ItemService.jar

194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212

connection conn = connection.getConnection().getConnection();
if (conn != null) {
    statement = conn.createStatement();
    log.info("DOC: " + queryStr);
    //com.fortify.dev.Security.declareSafe(queryStr);
    //queryStr = Class.forName("java.lang.String").get();
    try {
        ResultSet rs = stmt.executeQuery(queryStr);
        while (rs.next()) {
            Item item = new Item[Long.valueOf(rs.getString(1)), rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5), rs.getString(6), rs.getString(7)];
            list.add(item);            }  
        conn.close();
    }
    return list;
}
Hybrid Integrated Security Testing

Security Tester

HP WebInspect

Web Application

Source Code

Fortify Source Code Analysis

Hybrid Results

Results

Results

Development

© 2009 All Right Reserved Fortify Software Inc.
FortifyHybrid Integration Demo
How did we do that?

• “Runtime Data” comes from Runtime Analysis
  ✓ Today Fortify leverages this to monitor and guard applications

• Fortify Runtime Analysis + WebInspect = Hybrid 2.0
  • Runtime Analysis is required to ensure proper mapping of SAST/DAST results
  • Runtime Analysis allows testers and programmers to see “inside” the app
  • Runtime analysis makes black box testing – white box testing
Introducing Hybrid 2.0
Hybrid 1.0 (2005 Technology – Available since 2006)

Hybrid Aggregation:
The complete set of results

Unified management & reporting
Ability to combine SAST and DAST findings for integrated prioritization and reporting.

Hybrid 2.0 (An HP/Fortify exclusive advantage)

Hybrid Correlation
The accurate results

Reduced time and cost to fix vulnerabilities
Ability to follow test findings “into” the program and the code to see the root cause.

Hybrid Insight
The actionable results
Thank you!