CYBER SECURITY DIVISION
2013 PRINCIPAL INVESTIGATORS’

Code Dx: Visual analytics for triage of source code vulnerabilities

Secure Decisions, a division of Applied Visions, Inc.
Anita D’Amico
Ken Prole

September 17, 2013
Secure Decisions

We help you **make sense of data**
- Analyze security *decision-making* processes
- Build *visual analytics* to enhance security
decisions and training

Our expertise starts where automated security
sensors and scanners leave off

We **transition** our R&D into **operational use**, in government and industry

Grounded in commercial software and product development
- Division of Applied Visions, developer of commercial software
- 40 people, most with clearances, and secure facilities
Hackers are paid bounties to find software flaws

In 2010, Google started paying hackers up to $3,133.70 ... for bugs in its Web browser.

Last month, Microsoft sharply increased the amount it was willing to pay for such flaws, raising its top offer to $150,000.
**Software Assurance**

**SwA Terminology**

**Weakness**  Source code defect that an attacker *might* exploit

**Vulnerability**  Source code defect *known* to be exploitable
  - For simplicity, we’ll use “vulnerability” in this presentation

**SAST**  Static Application Security Testing tools
  - Find vulnerabilities and poor quality in static source code
  - Rapidly growing market
    - Commercial: Fortify, AppScan, Armorize, Coverity ...
    - Open source: FindBugs, Jlint, cppcheck ...

**Other categories of tools**

**DAST**  Dynamic Application Security Testing tools
  - Penetration testing of web applications during execution

**Binary code analysis**
  - Finding vulnerabilities through analysis of compiled code
The Need
Stop shipping insecure software

On average, one SAST tool finds only **14%** of vulnerabilities; you need lots of different tools to **cover** the vulnerabilities

50,000 weaknesses in 200,000 lines of code ...

*Where do I start? What’s most important?*

“**90%** of reported security incidents result from **exploits** of application software **defects**

Build Security In Website, DHS
https://buildsecurityin.us-cert.gov/bsi/mission.html
Code Dx Approach

**Find the most important vulnerabilities**

**Challenge**
- Incomplete vulnerability coverage by single tool
- Difficult to compare tool results; different semantics
- Tens of thousands of vulnerabilities reported
- Format of results impedes communication and collaboration
- Expensive tools; hard to use for non-experts

**Code Dx Solution**
- Imports and correlates results from multiple tools
- Normalizes results; common severity scale
- Visual analytics to rapidly triage results
- Common UI with custom detail for security analysts, developers, and CISOs
- *Code Dx* will embed open source SAST tools for use with or without commercial tools
Visual Analytics
for triage, remediation, and communication

Workflows tailored to each type of user
Interactive, powerful filtering
Visualize thousands of weaknesses in a single view
Quickly and effectively triage large weakness lists
**SINGLE INTERFACE FOR CORRELATED RESULTS FROM MULTIPLE TOOLS**

- **Normalized severities**
- **Tool attribution**
- **Total from all 5 tools**
- **Overlap detection**
- **Correlated source code mappings**
- **Correlated standards mappings**

---

**WebGoat > Analysis Run 1**
- Created on 7/16/2013
- Uploaded on 7/16/2013
- **2,123 total weaknesses**

1. **Filters**
   - Weakness count: 2,123 / 2,123

2. **Tool**
   - CodedCool (9.1%)
   - FindBugs (19.1%)
   - Fortify (51.2%)
   - JUnit (7.4%)
   - PMD (13.2%)

3. **Severity**
   - Unspecified (7.4%)
   - Low (43.3%)
   - Medium (33.6%)
   - High (15.4%)

4. **Codebase Location**
   - org.owasp.webgoat (100%)
   - org.owasp.webgoat.lessons (86.3%)
   - org.owasp.webgoat.servlets (< 0.1%)
   - org.owasp.webgoat.session (10%)
   - Catcher (0.2%)
   - HammerHead (1.1%)

5. **Tool Overlaps**
   - 1 Tool (92%)
   - 2 Tools (8%)

6. **CWE**
   - CWE-497: Exposure of System Data to an Unprotected Network
   - CWE-398: Indicator of Poor Code Quality
   - CWE-296: Declaration of Catch for Generic Exception
   - CWE-404: Improper Resource Shutdown or Release
   - CWE-493: Incomplete Cleanup

7. **Bulk Operations**
   - Displaying 2,123 matching weaknesses
   - Select a status...
   - Generate Report...

---

**Secure Decisions**

**Code Dx**

Version 0.9.7 - 7/10/2013

Logged in as admin
## ACTIONABLE WORKFLOW

**Well defined triage process**

**Bulk processing**

**Powerful filtering**

**Assign for remediation**

**Communicate status**

<table>
<thead>
<tr>
<th>ID</th>
<th>Weakness Type</th>
<th>Severity</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Unreleased Resource - Database</td>
<td>High</td>
<td>Fixed</td>
</tr>
<tr>
<td>101</td>
<td>Unreleased Resource - Database</td>
<td>High</td>
<td>Ignored</td>
</tr>
<tr>
<td>102</td>
<td>Unreleased Resource - Database</td>
<td>High</td>
<td>Unresolved</td>
</tr>
<tr>
<td>103</td>
<td>Unreleased Resource - Database</td>
<td>High</td>
<td>Escalated</td>
</tr>
</tbody>
</table>

---

For the 3 selected weaknesses:

- **Unresolved**
- **Escalated**
- **Ignored**
- **False Positive**
- **Fixed**

Choose from:

- Jane
- John
DEVELOPER FOCUSED REMEDIATION

Correlated weaknesses:
- 944 (Current Weakness)
- 302 SQL line 101
- 538 CheckResultSet line 101

Impacted source:
```java
protected Element createContent(WebSession s) {
    return super.createStagedContent(s);
}
protected Element doStage1(WebSession s) throws Exception {
    return injectableQuery(s);
}
protected Element doStage2(WebSession s) throws Exception {
    return parameterizedQuery(s);
}
ElementContainer ec = new ElementContainer();
try {
    Connection connection = DatabaseUtilities.getConnection(s);
    ec.addElement(makeAccountLine(s));
    String query = "SELECT * FROM user_data WHERE last_name = '" + accountName + '";";
    ec.addElement(new PreparedStatement(query));
    try {
        Statement statement = connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                           ResultSet.CONCUR_READ_ONLY);
        ResultSet results = statement.executeQuery(query);
        if ((results != null) & (results.first() == true)) {
            ResultSetMetaData resultsMetaData = results.getMetaData();
            resultsMetaData.writeTable(results, resultsMetaData);
            results.last();
            // If they get back more than one user they succeeded
            if (results.getRow() >= 6) {
```
Benefits

• **Better Coverage** - Find more important vulnerabilities
  – Combine multiple tool results to find more vulnerabilities
  – Prioritize combined results to highlight most important
  – Filter out overlapping results and false positives

• **Efficiency** - Save remediation time and resources
  – Developers can remediate highest priority vulnerabilities first
    • Remediation can take 7–10 hours per vulnerability

• **Communicate** more effectively up and down the chain
  – Visual analytics and reports, based on roles and expertise

• **Easy** to get started
  – Code Dx 1.0 (Q4 2013) auto-runs open source tools
  – Affordable to small and mid-sized businesses
Current Status
Technology Readiness Level 7

• More than **10 beta testers**, incl. ITT, NSA, Raytheon, RTI, Univ. of Nebraska …

• Systematic collection of **feedback**

  “I really like the visualization. ..tying [the tools] together and being able to work with that data is very useful.”

  “…at the present state, it seems to require the user to do a lot of work and formatting that the software itself could do.”

  “After a few minutes, I was able to manipulate the filters well enough to focus on particular discoveries.”

• Currently being **evaluated** by NIST, DHS S&T CIO, TSA, McAfee, Domestic Nuclear Detection Office, Indiana Univ.

• Working with Morgridge Institute to integrate into **SWAMP**

• **Training** program being refined
# Code Dx Roadmap

<table>
<thead>
<tr>
<th>Version</th>
<th>Major Features</th>
<th>Target Users</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Dx 1.0</td>
<td>Embeds and <strong>auto-runs open source tools</strong>; produces consolidated normalized results</td>
<td>Users seeking low-cost, easy-to-use bundle of SAST tools (e.g. small biz)</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Code Dx 1.1</td>
<td>More open source tools with <strong>additional languages</strong>, e.g. .Net; <strong>Enhanced reporting</strong></td>
<td>As above</td>
<td>Q3</td>
<td>Q4</td>
<td>Q1</td>
</tr>
<tr>
<td>Enterprise 1.0</td>
<td><strong>Correlates commercial and open source SAST tools</strong>; Enhanced analytics/reporting</td>
<td>Users of commercial SAST who want to extend their vulnerability coverage</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Enterprise 1.5</td>
<td><strong>Dynamic tracing</strong> from Code Pulse; <strong>enhanced visual analytics</strong></td>
<td>As above</td>
<td>Q3</td>
<td>Q4</td>
<td>Q1</td>
</tr>
<tr>
<td>Enterprise 2.0</td>
<td>Maps weaknesses to <strong>compliance</strong> (e.g. HIPAA) and industry standards (e.g. OWASP Top Ten)</td>
<td>As above, plus industry verticals, e.g. health</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>Enterprise 3.0</td>
<td>Addition of <strong>hybrid analysis</strong> correlating SAST and DAST</td>
<td>As above, with focus on web application risk assessment</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>SWAMP 1.0</td>
<td>More tool adaptors; Modified for <strong>SWAMP beta</strong></td>
<td>SWAMP users</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>SWAMP 1.1</td>
<td>Upgrades for <strong>SWAMP IOC</strong></td>
<td>SWAMP users</td>
<td>Q3</td>
<td>Q4</td>
<td>Q1</td>
</tr>
<tr>
<td>SWAMP 2.0</td>
<td>SWAMP Yr 2 added <strong>scalability</strong> and functions</td>
<td>SWAMP users</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>SWAMP 3.0</td>
<td>Upgrades for SWAMP Yr 3 custom needs</td>
<td>SWAMP users</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>SIEM Beta</td>
<td><strong>Feed pre-correlated SAST data to SIEM</strong>; Beta to be integrated into McAfee ESM</td>
<td>SIEM vendors; SIEM users</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>ED 1.0</td>
<td><strong>Free version of Code Dx for education</strong></td>
<td>Academia; training organizations</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
</tr>
</tbody>
</table>
Next 120 days towards transition

1. Transition Code Dx into government programs
   – Integrate into SWAMP Beta version
   – Determine effectiveness in NIST SATE program
   – Have other government agencies evaluate Code Dx

2. Initiate operational pilots to reach TRL 8
   – Recovery Accountability and Transparency Board; Commonwealth of PA

3. Conduct full commercialization
   – Integrate with McAfee ESM to demonstrate value proposition to Security Information Event Management (SIEM) users
   – Determine pricing model; Set up reseller program
   – Gain active use by at least one Fortune 500 company

4. Continue collaboration with academia
What do you think?

Diagnosis and triage of source code vulnerabilities

Anita D’Amico, Ph.D.
Director, Secure Decisions
(631) 759-3909
anita.damico@securedecisions.com

Ken Prole
Principal Investigator
(631) 759-3907
ken.prole@securedecisions.com