Applying Static Analysis – Matias Madou and Daan Raman
SecAppDev, Leuven, Feb 27, 2015

Introduction
Who are we?

At NVISO, I’m responsible for the software security practice. Next to the client work, I also leads NVISO’s product development efforts.

I’ve over a decade of software security experience. I was fortunate to spend 7 years building the leading static analysis solution as well as investigate leading software security initiatives at Fortune 100 companies through BSIMM. Currently, I’m applying that knowledge to dramatically improve software security initiatives in a cost efficient way.

Matias Madou, Ph.D.
+32 (0) 495 25 49 78
mmadou@nviso.be
www.linkedin.com/in/matiasmadou/

I am a member security consultant at NVISO, and I specialize mostly in software security. I mainly use my software engineering skills during penetration tests and code reviews of mobile and desktop applications.

I am additionally responsible for NVISO’s Research & Development team, leading our technical research with a current focus on application security for mobile ecosystems and malware analysis.

Daan Raman
+32 (0) 478 65 79 36
draman@nviso.be
t.be.linkedin.com/in/daanraman

Round of introductions
Welcome!
Our goal is to let you be successful at what you’re doing

• What company do you work for, what business unit do you work in?
• What are you working on? (Any static analysis?)
• What is your security background? (Trainings, hands-on, certification)
• What would you like to get out of this session?

Secure Software Development
Why it matters
Introduction
Any Big Software Security Issues?

2009: The biggest case of identity theft in American history. 130 million credit card numbers using an SQL injection attack. Cost of breach: $140m.

2011: LulSec hacks Sony Pictures Reveals 3m passwords unguarded using SQL injection attack. Sony did not learn from this bad experience.

2012: The hacker group D33Ds stole 450,000 plain text login credentials from Yahoo! The breach was carried out by using a union-based SQL injection attack.

Introduction
Who’s job is software security?

Developers
IT Security
Nobody?

Software Security is everybody’s job! It needs to be carried out throughout the organization, from the bottom to the top.

Introduction
Problems in code, so what?

Do you want to have more or less code next year?

More code, more potential problems!

Introduction
Well, we have Security Software (same as Software Security, right?)

Security Software
Software Security

Introduction
OK, what’s the impact of such a program?

Windows LOC vs. #CVE


Introduction
Find security problems as fast as possible in the SDLC.
Bad decisions can have big consequences!

The Secure Development Life Cycle (SDLC)

On Saturday, Nov. 30, the hackers had set their traps and had just one thing to do before starting the attack: plan the data’s escape route.

As they uploaded exfiltration malware to move stolen credit card numbers—first to staging points spread around the U.S. to cover their tracks, then into their computers in Russia—FireEye spotted them. Bangalore got an alert and flagged the security team in Minneapolis. And then …

Nothing happened.

The SDLC
Secure Development Life Cycle

1. Req. & Design
2. Develop & review
3. Test
4. Deploy
5. Maintain

In an ideal scenario, security is an integrated part in each phase of the Software Development Life Cycle.

The SDLC
Secure Development Life Cycle

... the reality (Functionality over Security)

A realistic goal: developers & architects find a happy medium between functional and security requirements

The SDLC
Implementation

Comprehensive Lightweight Application Security Process (CLASP, OWASP Project)

Touch-Point Model
(Gary McGraw)

Microsoft Secure Software Development Life Cycle (SDL)

The SDLC
Implementation

Implement and use Secure Coding Standards
• Ban "dangerous" functions
  exec, passthru, shell_exec, system, proc_open, popen, curl_exec, curl_multi_exec, parse_ini_file, show_source

Perform reviews
• Manual source code reviews (ie. peer reviews)
• Automated static analysis

The consequences of a security breach can be very expensive! Can you come up with examples?

Bug vs. Flaw

The earlier security issues are identified, the lower the remediation cost.

Source: The economics of testing (Rice Consulting)

The relative cost of fixing defects in various stages of the SDLC.

The consequences of a security breach can be very expensive! Can you come up with examples?

The earlier (security) issues are identified, the lower the remediation cost.
Defects: Bugs vs. Flaws

What do the terms “bug” and “flaw” mean to you?

Bugs

Simple mistake. An error.

Flaws

A design problem.

The first computer bug

1947
Mark II Relay Calculator
Moth trapped in a relay
Moth was removed and a note was made in the log
"First actual case of bug being found"

Ariane 5 launch failure

The launch, which took place on Tuesday, 4 June 1996, ended in failure due to an error in the software caused by assertions having been turned off, which in turn caused inadequate protection from integer overflow.

This resulted in the rocket veering off its flight path 37 seconds after launch, beginning to disintegrate under high aerodynamic forces, and finally self-destructing by its automated flight termination system. The failure has become known as one of the most infamous and expensive software bugs in history.

The failure resulted in a loss of more than US $370 million.
Bugs vs. Flaws

(in) Famous examples

Where do we find mainly bugs? Where do we find mainly flaws?

Architectural Risk analysis
Source Code Review
Threat Modelling
Penetration testing

Bugs vs. Flaws
Applied to Secure Development Life Cycle

1. Req. & Design
2. Develop & review
3. Test
4. Deploy
5. Maintain

Bugs vs Flaws
Quiz...

“bug” or “flaw”
**Application Security Testing**

**Different approaches**

During **Application Security Testing**, we are going to analyze the source code and/or the compiled version of the code in order to identify potential security defects.

**Static Application Security Testing (SAST)**

**Dynamic Application Security Testing (DAST)**

**Application Security Testing**

**Combining Static and Dynamic Analysis**

To get the most out of Application Security Testing, we will often combine both static and dynamic techniques.

**Automated techniques**

- **Static Analysis**
  - Read through and review code manually
- **Dynamic Analysis**
  - Step through code using a debugger

**Manual techniques**

- **Static Analysis**
- **Dynamic Analysis**
Most modern IDE’s support built-in static code analysis.

Demonstration

IBM AppScan

HP Fortify (audit workbench)

Which techniques are most suitable in order to detect bugs & flaws?

Manual and Automated techniques

 Leaders: HP, IBM, Veracode and WhiteHat Security:
✓ Offer at least SAST and DAST
✓ IAST (Interactive Application Security Testing) and RASP (Runtime) are a differentiator

Mainly automated source code review techniques but also manual
Mainly manual architectural risk analysis
Static Analysis Theory
10,000 feet view

A 10,000 feet view of static analysis

<table>
<thead>
<tr>
<th>Static Analysis Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1:</strong> Translate phase: Translate source code to some language independent intermediate language</td>
</tr>
<tr>
<td><strong>Step 2:</strong> Scan phase: Apply analysis on top of the generic intermediate language</td>
</tr>
</tbody>
</table>

Model

- Build up model: Do analysis on model

- Results:
  1) ...
  2) ...
  3) ...

Scanning

- Rules (language specific)

Theory: translation

- Missing library files (end points)
  (Source code scanning does not decompile, but extracts key information!!)

- Duplicate (but different code)
Static Analysis Theory
Theory: phase 2: scanning
Build up model. Do analysis on model

Language support:
- Java
- .NET
- PHP
- ...

Packages:
- Hibernate
- Struts
- ...

Limitations of static analysis

False positives vs. False negatives

Static analysis says you don't have this bug
Static analysis says you have this bug

... and you don't have this bug
... and you do have this bug

False Negatives
False Positives
True Negatives
True Positives

When starting with static analysis...

The tool is reporting 10,000 instances of XSS. We don't care about these!

... and you do have this bug

The tool is reporting 10,000 instances of XSS! We don't care about these!

Wanted results
Unwanted results

No clue about false negatives

Thinks you have an overwhelming amount of false positives (and that may be true)

False Negatives
False Positives
True Positives

Wanted results
Unwanted results

?? False Negatives ??
What you would like to achieve...

Reduce the amount of false positives to a manageable amount.

... but what about the true positives?

How to manage the amount of true positives?

Where do false positives and negatives come from?

Example: Bug/Product does not really work with the framework.

The static analysis tool does not know Struts...

It ignores the "actionerror escape" attribute – false negative!

Triaging

Good part: The bulk of the work should be done by fine-tuning the scan with custom rules.

Bad part: they weeded out a ton of trivial issues, the hard part is left for you!

Triaging

Can it be exploited?

Find exploit

Argue about the exploit, severity, ...
**Triaging**

High-level overview

1. Read the evidence
2. Understand the problem category
3. See if the evidence supports the problem category
4. Take appropriate action

---

**Static Analysis Theory**

---

**Static Analysis Theory**

Analyzers

Different analyzers find different issues:
- Structural matching
- Taint tracking
- Control flow analysis
- Configuration file analysis
- Content analyzers (html files)
- ...

---

**Structural matching**

(Grep++)

What other categories can be used to grep?
(Think about manual code review)

---

**Taint tracking**

Most important analyzer

High level overview of taint tracking (what the dataflow analyzer does)

```
Parameter1 = Request.getParameter("the_first_parameter");
Other_variable = CopyData(Parameter1);
CommandLine.execute(Other_variable);
```
Taint tracking
Most important analyzer

High level overview of taint tracking (what the dataflow analyzer does)

Source of taint
Parameter1= Request.getParameter("the_first_parameter");
Other_variable= CopyData(Parameter1);
CommandLine.execute(Other_variable)
Sink

Passthrough rule:
Q: How can data move through the solution? (White board)

Q: Where can data enter our solution? How can data enter the solution? (White board)
Inform Developers

**Bugs Flow**
Inform Developers

- How do the developers get informed?

```
Issue -> Audit

Problem
```

Q&A
Thank you!

Contact information:
- mmadou@nviso.be
- draman@nviso.be
- @mmadou / @daanraman
- @NVISO_BE
- www.nviso.be