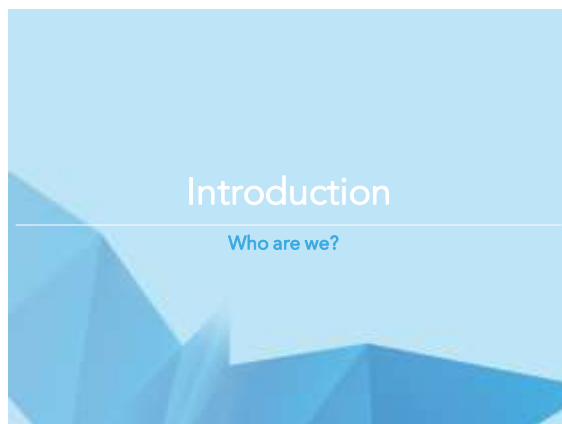




Applying Static Analysis – Matias Madou and Daan Raman
SecAppDev, Leuven, Feb 27, 2015

SecAppDev

1



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 drastically 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 a 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
be.linkedin.com/in/daanraman



SecAppDev

3



SecAppDev

4

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?



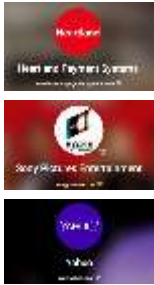
SecAppDev

5



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: LulzSec hacks Sony Pictures **Reveals 1m 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?



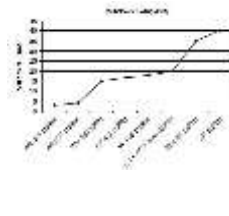
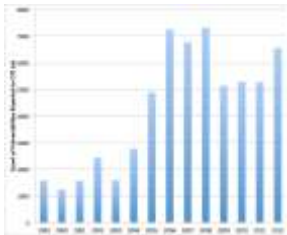
Software Security is everybody's job! It needs to be carried out throughout the organization, from the bottom to the top.

Introduction

Problems in code, so what?



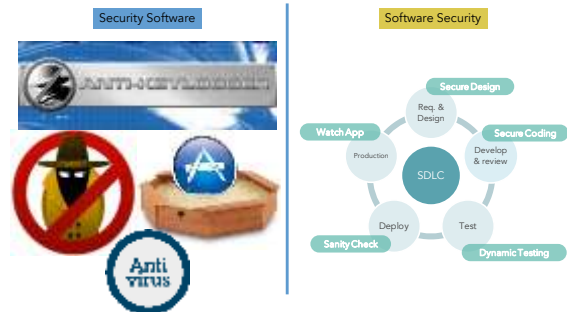
Do you want to have more or less code next year?



More code, more potential problems!

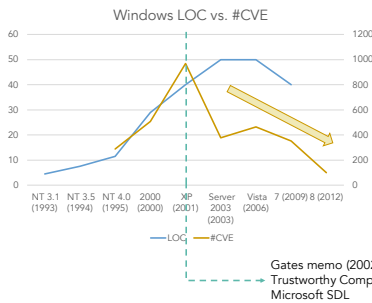
Introduction

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



Introduction

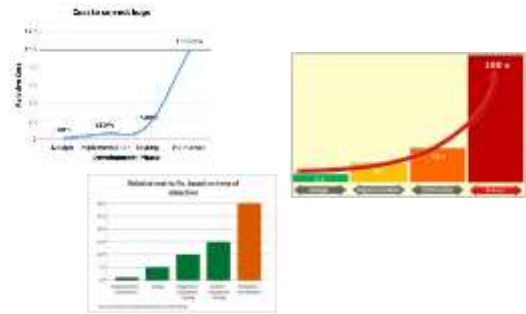
OK, what's the impact of such a program?

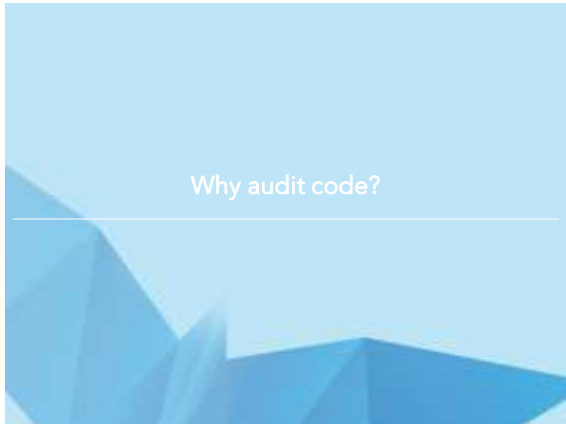


Gates memo (2002): Trustworthy Computing. Microsoft SDL

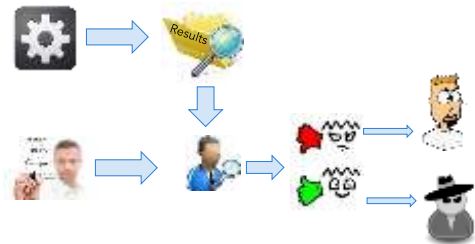
Introduction

Find security problems as fast as possible in the SDLC.

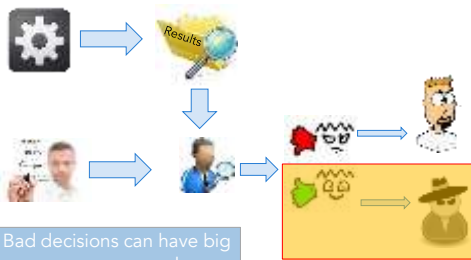




Why audit code?



Why audit code?



Bad decisions can have big consequences!

The SDLC

When things go wrong



Missed Alarms and 40 Million Stolen Credit Card Numbers: How Target Blew It

By Michael White, Ben Elger, David Johnson, and Carol Matlack | March 13, 2014

The SDLC

When things go wrong



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 ...

Missed Alarms and 40 Million Stolen Credit Card Numbers: How Target Blew It

Nothing happened.

Source: <http://www.businessweek.com/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data>



The SDLC

Secure Development Life Cycle



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



SecAppDev

19

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

A **Secure Development Life Cycle** is about building security into the software development life cycle.

Source: CISSP All-in-One Exam Guide 6th Edition, Shon Harris

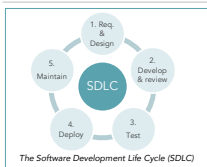


SecAppDev

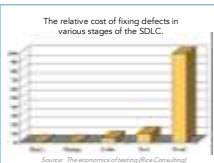
20

The SDLC

Why you need a Secure Development Life Cycle



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



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



SecAppDev

21

The SDLC

Implementation

Comprehensive Lightweight Application Security Process (CLASP, OWASP Project)

Touch-Point Model (Gary McGraw)

Microsoft Secure Software Development Life Cycle (SDL)



SecAppDev

22

The SDLC

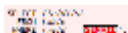
Implementation



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
```



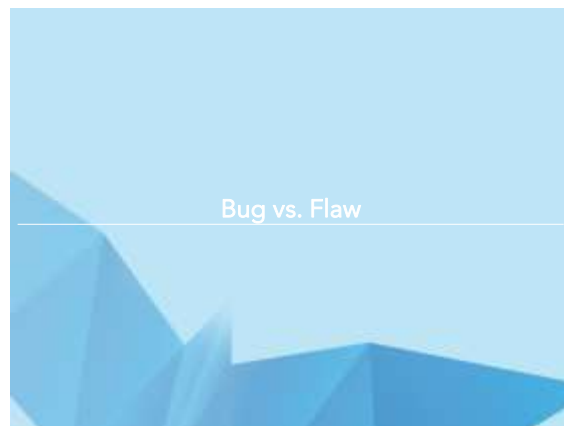
Perform reviews

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



SecAppDev

23



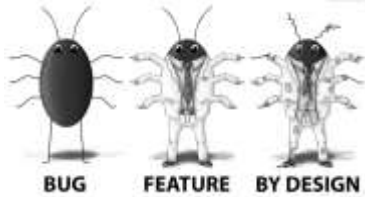
Bug vs. Flaw

Defects: Bugs vs. Flaws

Definition



What do the terms "bug" and "flaw" mean to you?



Bugs and Features by Iris-hime



SecAppDev

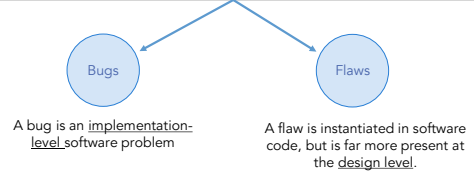
25

Defects: Bugs vs. Flaws

Definition



Defect
Can be both implementation vulnerabilities and design vulnerabilities



SecAppDev

26

Bugs vs. Flaws

Applied to code



Bugs

Simple mistake. An error.



Flaws

A design problem.



SecAppDev

27

Bugs vs. Flaws

(in) Famous examples



Bugs

The Heartbleed bug:



Flaws

Microsoft Bob:

Pardon me: Have you forgotten your password?
Type your password here: ...



SecAppDev

28

The first computer bug

(in) Famous examples



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"



SecAppDev

29

Bugs vs. Flaws

(in) Famous examples



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.



SecAppDev

30

Bugs vs. Flaws

(in) Famous examples



Why stability matters—the Airline IT crash

- Velocity is not represented on a 0-100 scale
- A conversion table of 10-99 is required (0-99 is not)
- The incorrect velocity of 100 is represented as a 10-99 value
- This rounding error represented the performance measure

```

- Vertical velocity has as measured by sensor
1. 00. 00. 00
...
- Check if measured vertical velocity has been
converted to a 10-99 int. If so, then convert
0.1, 0. 00. 00 = 10/99 float
P. M. 000000. 00. 0. 00 = 10/00000
...
- Horizontal velocity has as measured by sensor
- is converted to a 10 for the vertical checking
P. M. 000000. 00. 0. 00 =
...

```

\$370,000,000

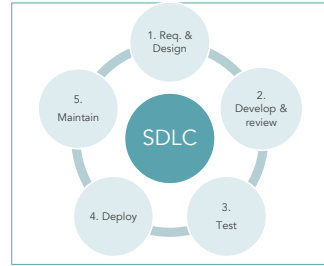
*Source: http://news.cnet.com/81-1069700270-38880000.html

SecAppDev

31

Bugs vs. Flaws

Applied to Secure Development Life Cycle



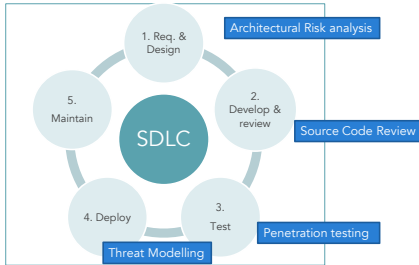
Where do we find mainly bugs? Where do we find mainly flaws?

SecAppDev

32

Bugs vs Flaws

Applied to Secure Development Life Cycle



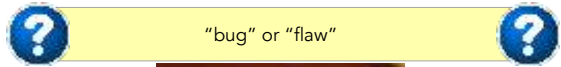
Where do we find mainly bugs? Where do we find mainly flaws?

SecAppDev

33

Bugs vs Flaws

Quiz...

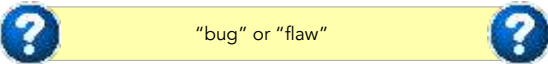


SecAppDev

34

Bugs vs Flaws

Quiz...

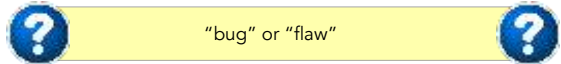


SecAppDev

35

Bugs vs Flaws

Quiz...



SecAppDev

36

Bugs vs Flaws

Quiz...

? "bug" or "flaw" ?

```
void main()
{
    char bufferA[50];
    char bufferB[163];

    printf("What is your name?\n");
    gets(bufferA);
    strcpy(bufferB, bufferA);

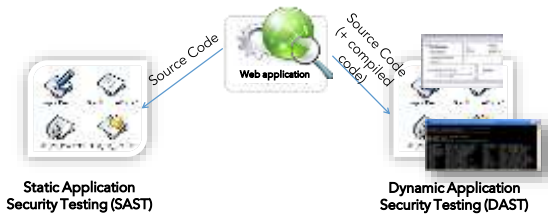
    return;
}
```



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.



Application Security Testing

Static Application Security Testing

```
if ( (SEQ==strcpy(SFID, "")) != false ) $fname
= sanitize_title_with_dashes(substr( SFID,
SEQ+1, strlen(SFID)-SEQ-1 ));
```

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.



Application Security Testing

Comparison of techniques

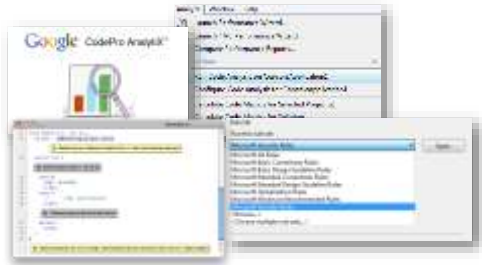
	Static Analysis	Dynamic Analysis
Automated techniques		
Manual techniques	<p>Read through and review code manually</p>	<p>Step through code using a debugger</p>

Application Security Testing

Support in your favorite IDE



Most modern IDE's support built-in static code analysis



Application Security Testing

Support in your favorite IDE



Demonstration

Application Security Testing



Stand-alone Static Application Security Testing tools



Application Security Testing

Manual and Automated techniques




Bugs	Flaws
Simple mistake. An error.	Design problems.
	

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

Application Security Testing

Manual and Automated techniques



Bugs	Flaws
Simple mistake. An error.	Design problems.
	
Mainly automated source code review techniques but also manual	Mainly manual architectural risk analysis

Application Security Testing

Choice in solutions: What does Gartner say?



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



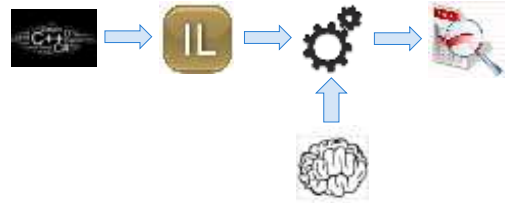
Static Analysis Theory

Static Analysis Theory

10,000 feet view



A 10,000 feet view of static analysis



SecAppDev

50

Static Analysis Theory

Model



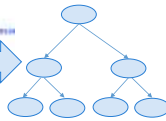
Step 1: Translate phase: Translate source code to some language independent intermediate language

```

Transaction (txn = null)
Session session = null;

try {
    session = HibernateUtil.getSessionFactory().getSession(sessionId);
    txn = session.beginTransaction();
    // get all transaction records for the given username and password
    String sql = "from Transaction t where t.username = username and t.pwd =
    query = session.createQuery(sql);
    query.setParameter("username", username);
    query.setParameter("password", password);
    List<Transaction> results = query.list();
    if(results.size() == 0) { // No login result is found
        status = "fail";
        log.info("fail" + username + "failed login!");
    } else { // More or multiple entries
        status = "fail";
        log.info("fail" + username + "failed login!");
    }
} catch (Exception e) { // Correct include other attributes to be
    log.info("fail" + username + "failed login!");
}
}

```



SecAppDev

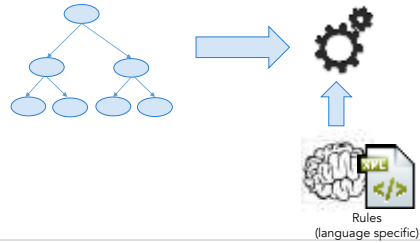
51

Static Analysis Theory

Scanning



Step 2: Scan phase: Apply analysis on top of the generic intermediate language



SecAppDev

52

Static Analysis Theory

Theory: translation



Build up model. Do analysis on model



SecAppDev

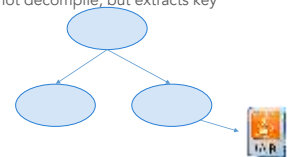
53

Static Analysis Theory

Theory: translation



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



Duplicate (but different code)



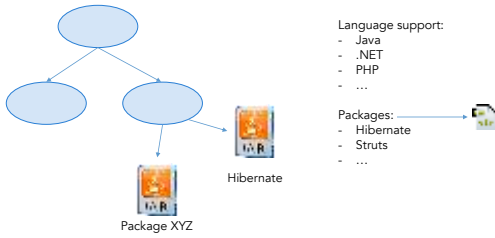
SecAppDev

54

Static Analysis Theory

Theory: phase 2: scanning

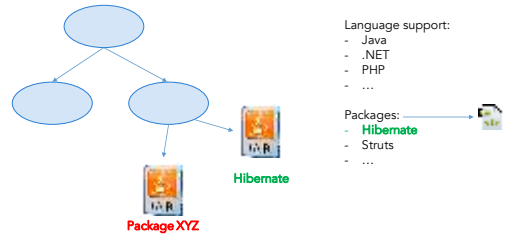
Build up model. Do analysis on model



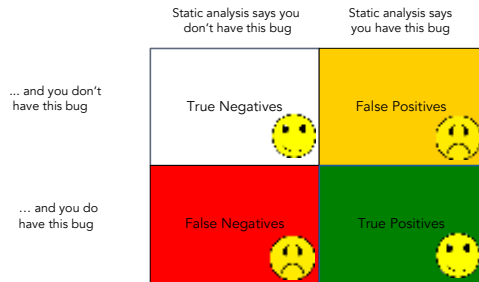
Static Analysis Theory

Theory: phase 2: scanning

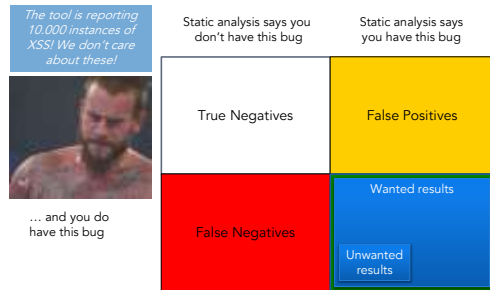
Build up model. Do analysis on model



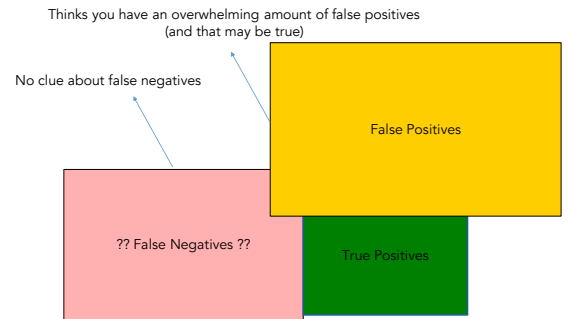
False positives vs. False negatives



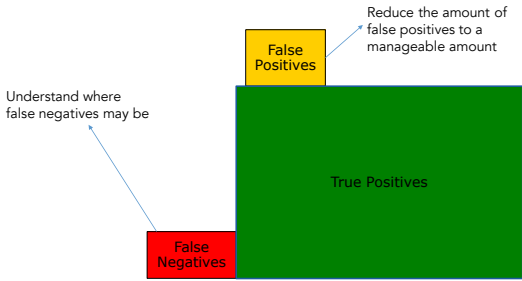
False positives vs. False negatives



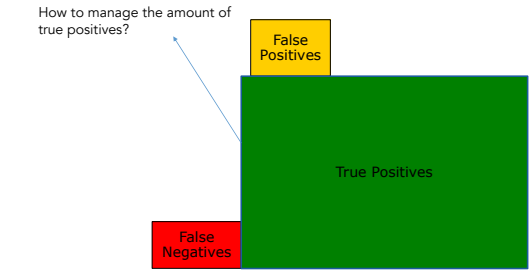
When starting with static analysis...



What you would like to achieve...



... but what about the true positives?



Where do false positives and negatives come from?

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



```

1: @param contextType: ContextType; context: Context;
2: @param url: string; url: string;
3: @param request: Request; request: Request;
4: @param response: Response; response: Response;
5: @param session: Session; session: Session;
6: @param cookie: Cookie; cookie: Cookie;
7: @param headers: Headers; headers: Headers;
8: @param body: Body; body: Body;
9: @param query: Query; query: Query;
10: @param path: Path; path: Path;

```

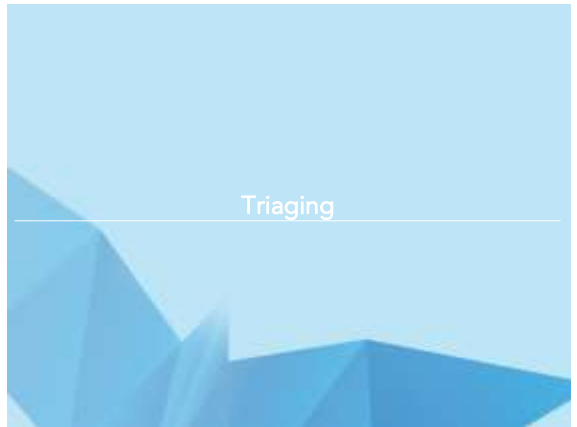
The static analysis tool does not know Struts...

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

```

1: @param contextType: ContextType; context: Context;
2: @param url: string; url: string;
3: @param request: Request; request: Request;
4: @param response: Response; response: Response;
5: @param session: Session; session: Session;
6: @param cookie: Cookie; cookie: Cookie;
7: @param headers: Headers; headers: Headers;
8: @param body: Body; body: Body;
9: @param query: Query; query: Query;
10: @param path: Path; path: Path;

```



Triaging

Going through issues



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

What to avoid



Triaging

High-level overview



SecAppDev

67



Static analysis theory

Analyzers



Different analyzers find different issues:

- Structural matching
- Taint tracking
- Control flow analysis
- Configuration file analysis
- Content analyzers (html files)
- ...



SecAppDev

69

Static analysis theory

Analyzers



Why do we need to understand the different analyzers:

- Understand why the tool is generating an issue
- Discover false positives and false negatives
- Feedback loop to solution improvements (customization)



SecAppDev

70

Structural matching

Grep++



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



SecAppDev

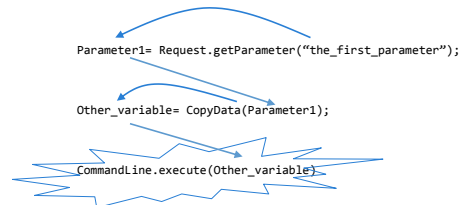
71

Taint tracking

Most important analyzer



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



SecAppDev

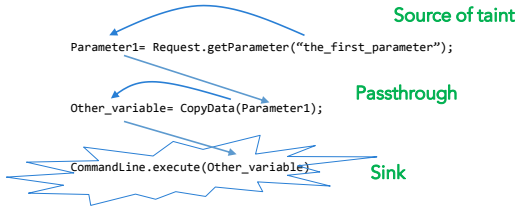
72

Taint tracking

Most important analyzer



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

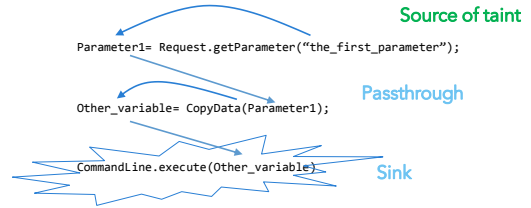


Taint tracking

Most important analyzer



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



Taint tracking

Most important analyzer



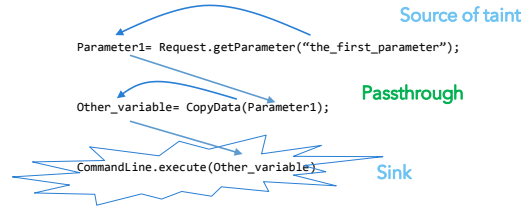
Source rule:
 Q: Where can data enter our solution? How can data enter the solution?
 (White board)

Taint tracking

Most important analyzer



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



Taint tracking

Most important analyzer



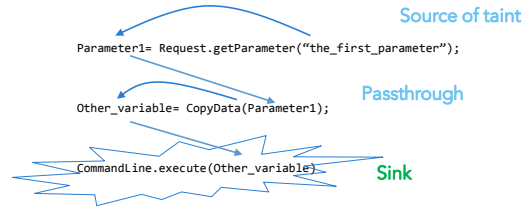
Passthrough rule:
 Q: How can data move through the solution?
 (White board)

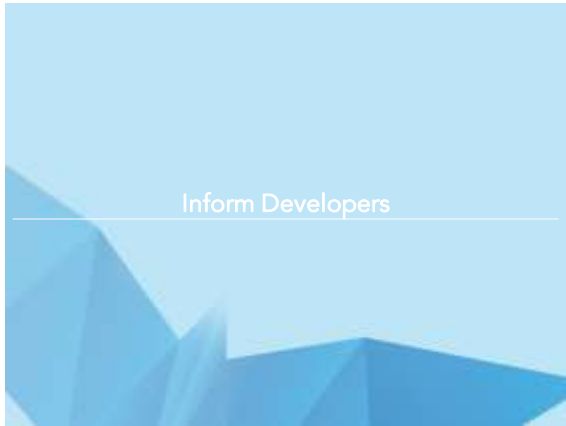
Taint tracking

Most important analyzer



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





Bugs Flow
Inform Developers



How do the developers get informed?



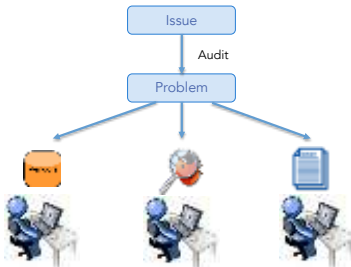
SecAppDev

80

Bugs Flow
Inform Developers

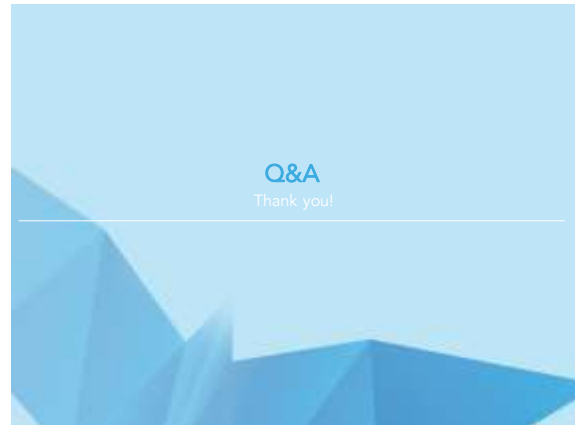







How do the developers get informed?



SecAppDev

81



-  mmadou@nviso.be
-  draman@nviso.be
-  @mmadou / @daanraman
-  @NVISO_BE
-  www.nviso.be