Software Vulnerability Discovery

Heejo Lee

Center for Software Security and Assurance

Korea University

Dec 4, 2015

Software Vulnerabilities

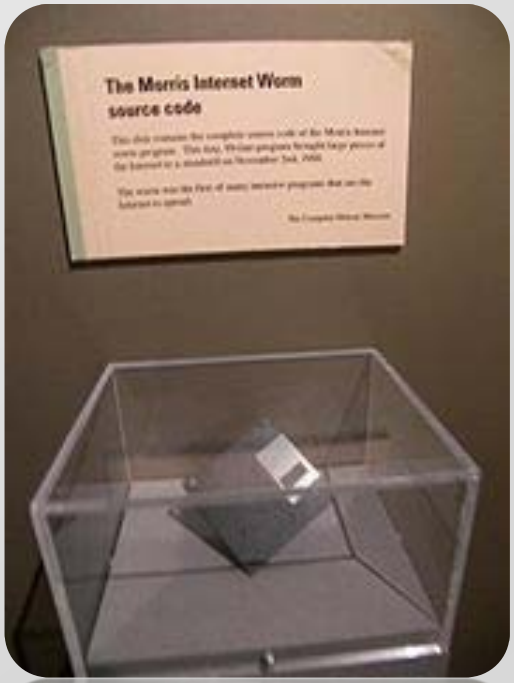
- Definition
  - An unintended flaw in software code (or a system) that leaves it open to the **potential for exploitation** in the form of unauthorized access or malicious behavior such as viruses, worms, Trojan horses and other forms of malware
    - http://www.webopedia.com/TERM/S/security_vulnerability.html
  - A vulnerability is a weakness which allows an attacker to reduce a system's information assurance. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw
    - https://en.wikipedia.org/wiki/Vulnerability_(computing)

Cases of Vulnerabilities

- **Intrusion**: Internet Worm (1988)
- **Reuse**: HeartBleed (2014)
- **Human Life**: Car & Airplain Hacking (2015)

Cases of Vulnerabilities

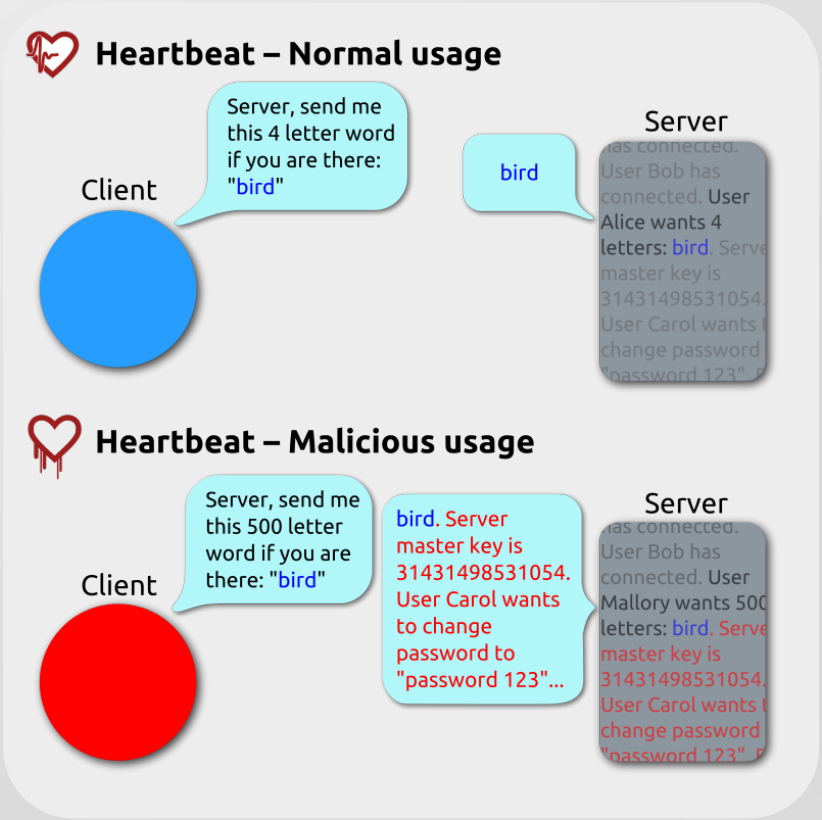
- **Intrusion**: Internet Worm (1988)
  - The Morris worm or Internet worm of November 2, 1988 was one of the first computer worms distributed via the Internet
  - It was the first to gain significant mainstream media attention. It also resulted in the first felony conviction in the US under the 1986 Computer Fraud and Abuse Act
  - It worked by exploiting known vulnerabilities in Unix sendmail, finger, and rsh/rexec, as well as weak passwords



Disk containing the source code for the Morris Worm held at the Computer History Museum, Mountain View, California, USA

Cases of Vulnerabilities

- **Reuse**: HeartBleed (2014)
  - The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library
  - This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet



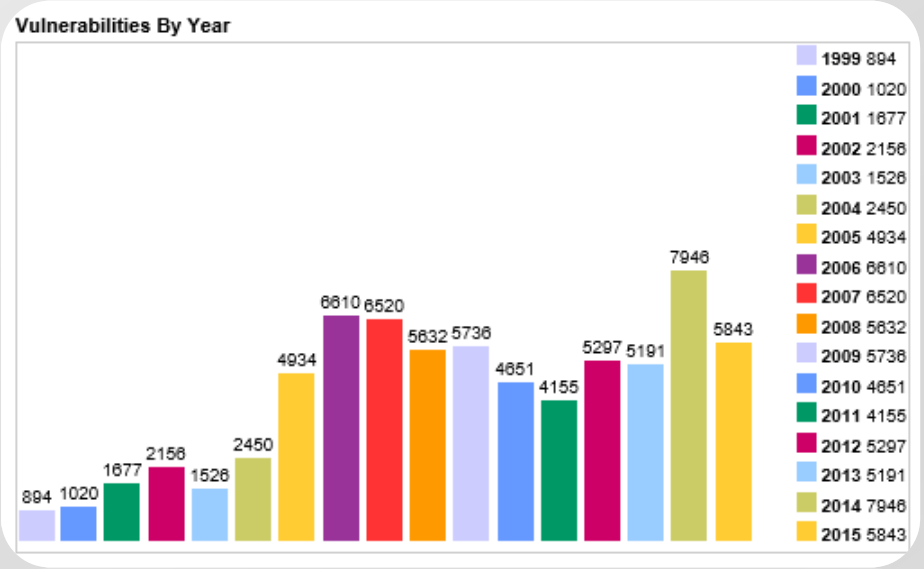
Cases of Vulnerabilities

- **Human Life**:
  Car & Airplane Hacking (2015)
  - Chrysler announced that it's issuing a formal recall for 1.4 million vehicles that may be affected by a hackable software vulnerability in Chrysler's Uconnect dashboard
  - U.S. commercial airliners could be hacked in flight by passengers using a plane's wireless entertainment system to access its flight controls

Jeep's brakes were remotely disabled

Vulnerabilities by Date

- Common Vulnerabilities and Exposures (CVE)
  - A reference-method for publicly known information security vulnerabilities and exposures
  - MITRE Corporation maintains the system, with funding from the National Cyber Security Division of the United States Department of Homeland Security



72,240 of vulnerabilities reported at CVE in Dec 2, 2015

KOREA UNIVERSITY CSSA Center for Software Security and Assurance

CVSS Score Distribution for All Vulnerabilities

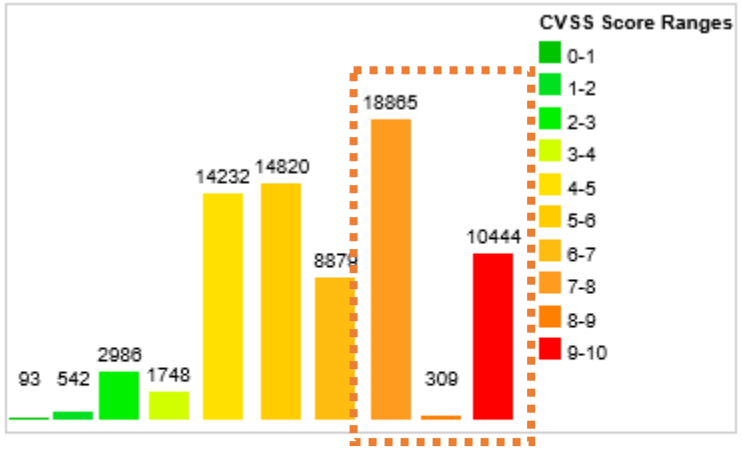
- NVD Vulnerability Severity Ratings
  - Open industry standard for assessing the severity of vulnerabilities
  - The scores range from 0 to 10

Distribution of all vulnerabilities by CVSS Scores

CVSS Score	Number Of Vulnerabilities	Percentage
0-1	93	0.10
1-2	542	0.70
2-3	2986	4.10
3-4	1748	2.40
4-5	14232	19.50
5-6	14820	20.30
6-7	8879	12.20
7-8	18865	25.90
8-9	309	0.40
9-10	10444	14.30
Total	72918	

Weighted Average CVSS Score: **6.8**

Vulnerability Distribution By CVSS Scores



High CVSS covers 41% of CVE's vulnerabilities

KOREA UNIVERSITY CSSA Center for Software Security and Assurance

CWE

- **Common Weakness Enumeration**
  - Creating a catalog of software weaknesses and vulnerabilities
  - The goal of the project is to better understand flaws in software and to create automated tools that can be used to identify, fix, and prev ent those flaws
  - The project is sponsored by Mitre Corporation

CWE-2000: Comprehensive CWE Dictionary

View Metrics

	CWEs in this view		Total CWEs
Total	**1003**	out of	1003
Views	32	out of	32
Categories	244	out of	244
Weaknesses	719	out of	719
Compound_Elements	8	out of	8

Vulnerabilities Scanning

Net Scanners

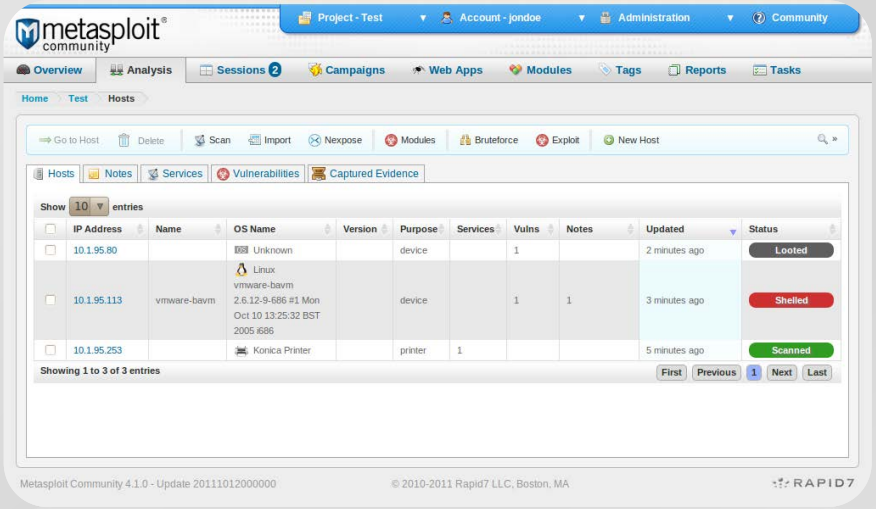
- Metasploit (http://www.metasploit.com/)
- Nessus (http://www.tenable.com/products/nessus-vulnerability-scanner)
- OpenVAS (http://www.openvas.org/)

Web Scanners

- w3af (http://w3af.org/)
- Nikto (https://cirt.net/Nikto2)
- Paros (http://www.testingsecurity.com/paros_proxy)

Net Scanners

- Metasploit Project
  - The Metasploit Project is a computer security project that provides information about security vulnerabilities and aids in penetration testing and IDS signature development
  - Its best-known sub-project is the open source Metasploit Framework, a tool for developing and executing exploit code against a remote target machine
  - It is world's most used penetration testing software

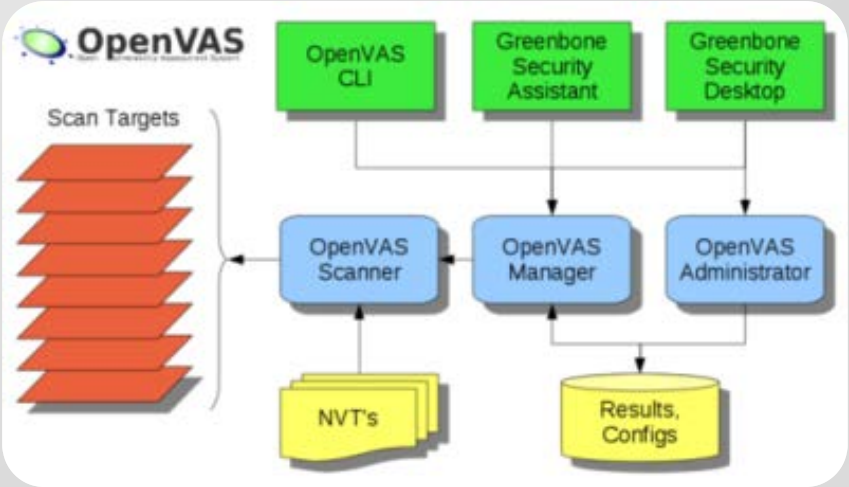


Net Scanners

- Nessus
  - Nessus is a proprietary comprehensive vulnerability scanner which is developed by Tenable Network Security. It is free of charge for personal use in a non-enterprise environment
  - According to surveys done 2009 by sectools.org, Nessus is the world's most popular vulnerability scanner, taking first place in the 2000, 2003, and 2006 security tools survey. Tenable Network Security estimated year 2005 that it was used by over 75,000 organizations worldwide

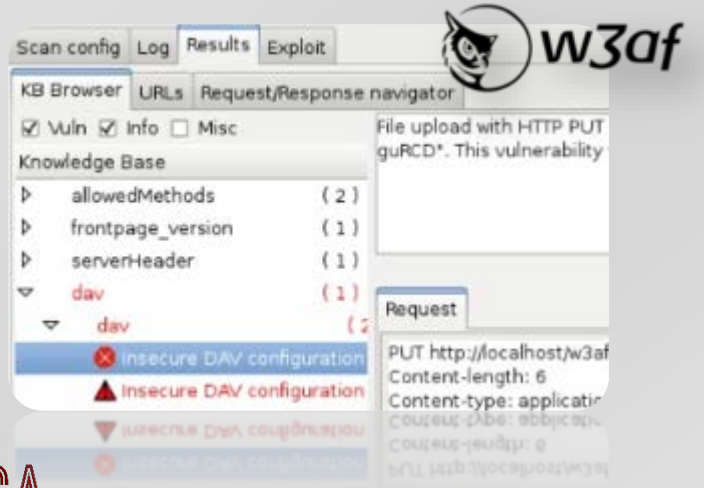
Net Scanners

- OpenVAS
  - OpenVAS is a framework of several services and tools offering a vulnerability scanning and vulnerability management solution
  - OpenVas began, under the name of GNessUs, as a fork of the previously open source Nessus scanning tool after Tenable Network Security changed it to a proprietary (closed source) license in October 2005
  - All OpenVAS products are Free Software
    - › Most components are licensed under the GPL



Web Scanners

- w3af
  - w3af (web application attack and audit framework) is an open-source web application security scanner
  - The project provides a vulnerability scanner and exploitation tool for Web applications
  - After identification, vulnerabilities like (blind) SQL injections, OS commanding, remote file inclusions (PHP), cross-site scripting (XSS), and unsafe file uploads, can be exploited in order to gain different types of access to the remote system



Web Scanners

- Paros
  - Paros is a valuable testing tool for your security and vulnerability testing
  - Paros can be used to spider/crawl your entire site, and then execute canned vulnerability scanner tests
  - It was used to attack KT, 2014
- Nikto Web Scanner
  - It is a Web server scanner that tests Web servers for dangerous files/CGIs, outdated server software and other problems
  - It performs generic and server type specific checks
  - It also captures and prints any cookies received

Finding Zero-day Vulnerabilities

- Zero-day Vulnerabilities
  - An undisclosed computer application vulnerability that could be exploited to adversely affect the computer programs, data, additional computers or a network
  - Known as a "zero-day" because once the flaw becomes known, the application author has zero days in which to plan and advise any mitigation against its exploitation (by, for example, advising workarounds or issuing patches)
- Methods of finding zero-day vulnerabilities
  - Dynamic analysis of binaries and network protocols
    - Done by automated **Black-box testing**
  - Static analysis of source codes
    - Done by automated **White-box testing**

Black-box Test Research

> *"**Black-box testing** is a method of software testing that examines the functionality of an application without peering into its internal structures or workings"* - Wikipedia

- The tester is aware of *what* the software is supposed to do but is **not** aware of *how* it does it

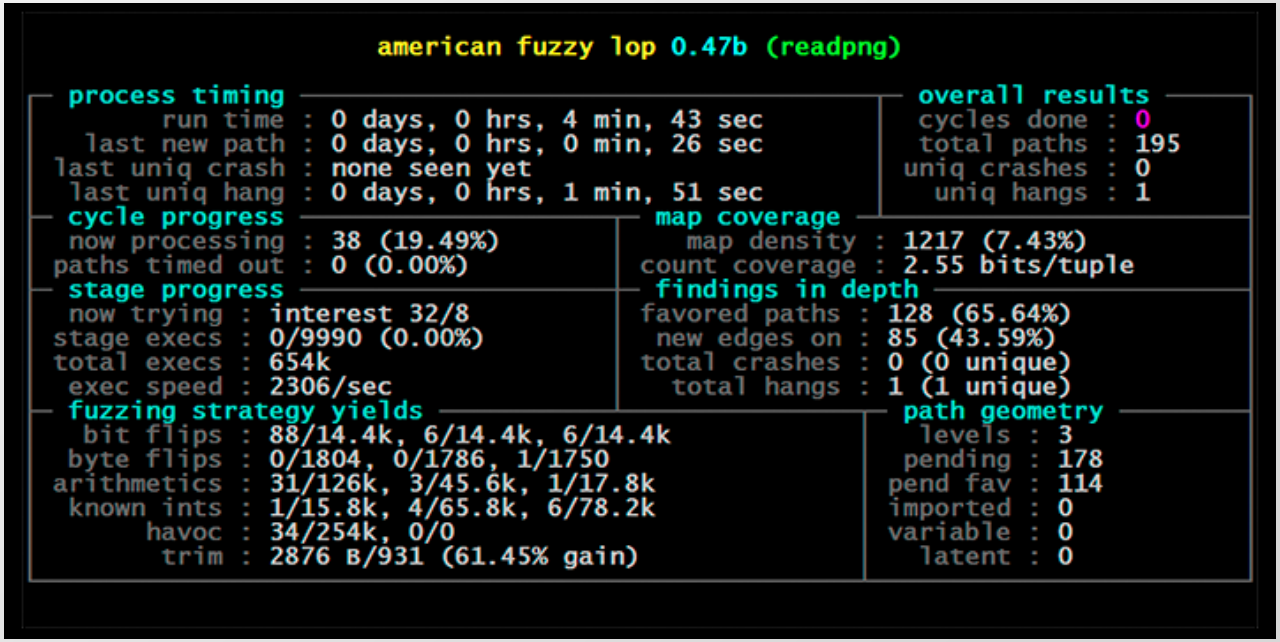
- Test cases generation
  - generally derived from external descriptions of the software
    - specifications
    - requirements
    - design parameters

Black-box Test Research

- Test designer selects both valid and invalid inputs and determines the correct output
- In penetration testing, the goal of a black-box penetration test is to simulate an external hacking or cyber warfare attack
- Open source fuzzing tools
  - Binary Fuzzing : AFL, Peach, Sulley, etc.
  - Network Fuzzing : Protos, Pwntooth, etc.

American Fuzzy Lop (AFL)

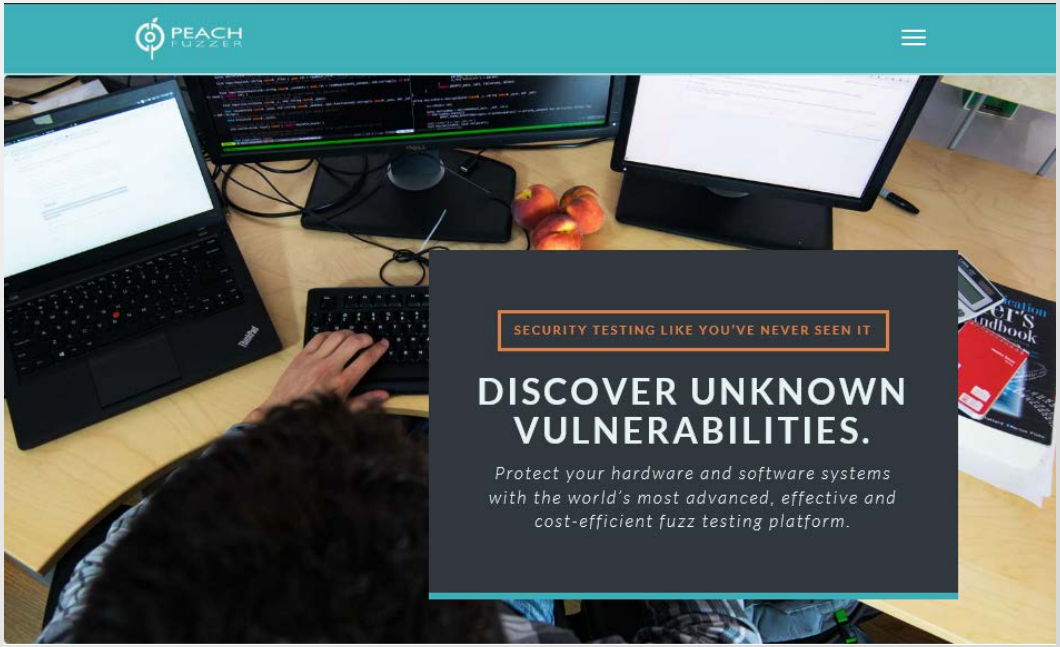
- **Compile-time dynamic fuzzer**
- Security-oriented fuzzer that employes a novel type of compile-time instrumentation
  - Automatically discover clean, interesting test cases that trigger new internal states in the targeted binary

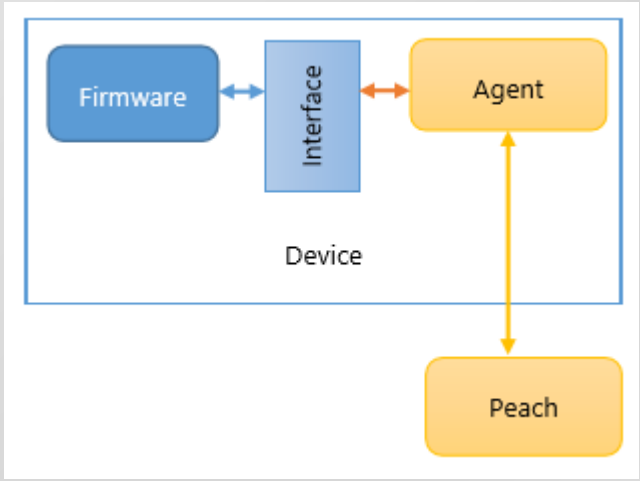


AFL fuzzing example

Peach Fuzzer

- **Easy-to-launch fuzzer** with XML libraries
- Open source fuzzing platform under MIT license
  - Provides extensive library of test definitions (Pits)
    - › For binary & network etc.





Binary(Firmware) Fuzzing

Sulley Fuzzer

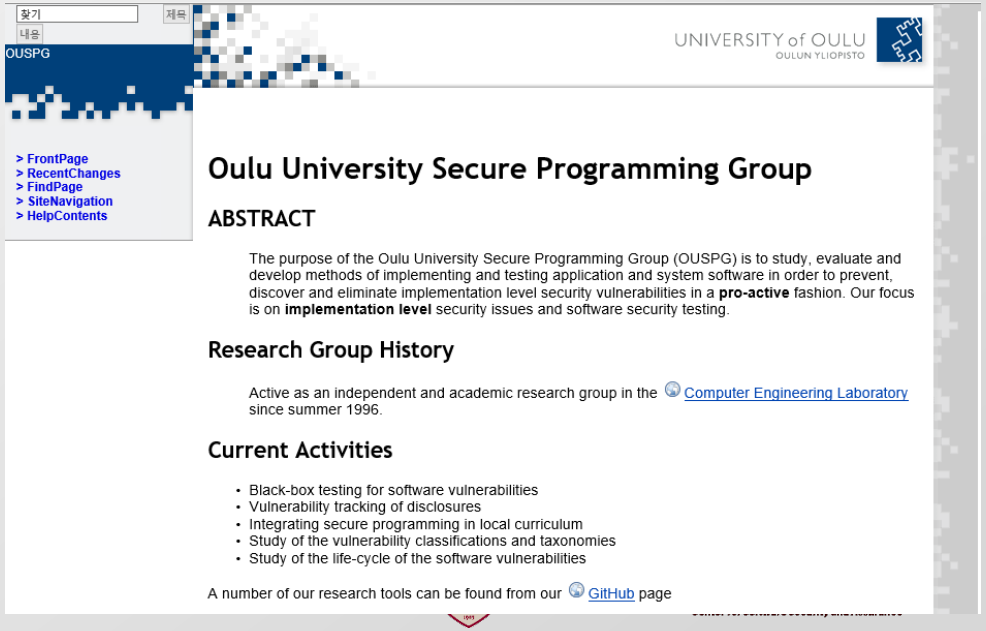
- **Easy-to-configure fuzzer** with python template
- Open source fuzzing platform under GNU license
  - Python-based customization
  - Provides library extensions
    - › For binary & network fuzzing

Web monitoring interface (example)

KOREA UNIVERSITY CSSA Center for Software Security and Assurance

Protos Project

- **Protocol implementations fuzzing** project
  - By Oulu University Secure Programming Group (OUSPG)
- To study, evaluate and develop methods of implementing and testing application and system software
  - In order to prevent, discover and eliminate implementation level security vulnerabilities

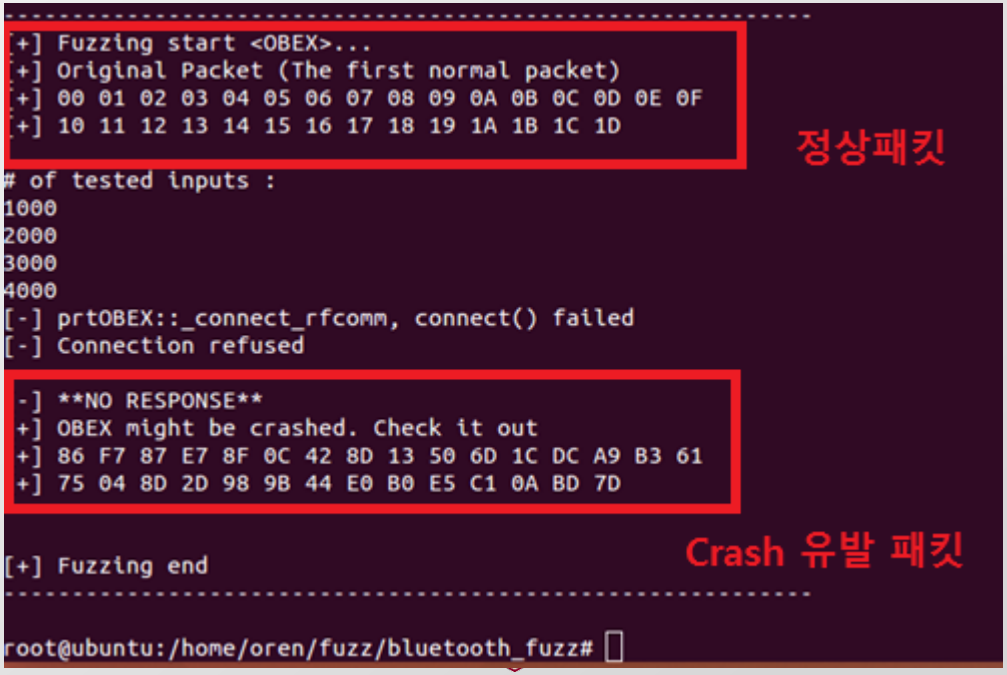

찾기
제목
내용
OUSPG
> FrontPage
> RecentChanges
> FindPage
> SiteNavigation
> HelpContents
UNIVERSITY of OULU
OULUN YLIOPISTO
Oulu University Secure Programming Group
ABSTRACT
The purpose of the Oulu University Secure Programming Group (OUSPG) is to study, evaluate and develop methods of implementing and testing application and system software in order to prevent, discover and eliminate implementation level security vulnerabilities in a **pro-active** fashion. Our focus is on **implementation level** security issues and software security testing.
Research Group History
Active as an independent and academic research group in the Computer Engineering Laboratory since summer 1996.
Current Activities
- Black-box testing for software vulnerabilities
- Vulnerability tracking of disclosures
- Integrating secure programming in local curriculum
- Study of the vulnerability classifications and taxonomies
- Study of the life-cycle of the software vulnerabilities

A number of our research tools can be found from our GitHub page

Protos project webpage

Pwntooth

- **Automated packet mutation fuzzer**
- Designed to automate Bluetooth penetration testing
  - Scans for the devices, runs the tools
    - › Blueper / Bluesnarfer / BSS / carwhisper



Bluetooth Stack Smasher (example)

Research Related

- Program-Adaptive Mutational Fuzzing (2015, CMU)
  - Leverage white-box symbolic analysis on an execution trace for a given program-seed pair to detect dependencies among the bit positions of an input
  - Use the dependency relation to **compute optimal mutation ratio**

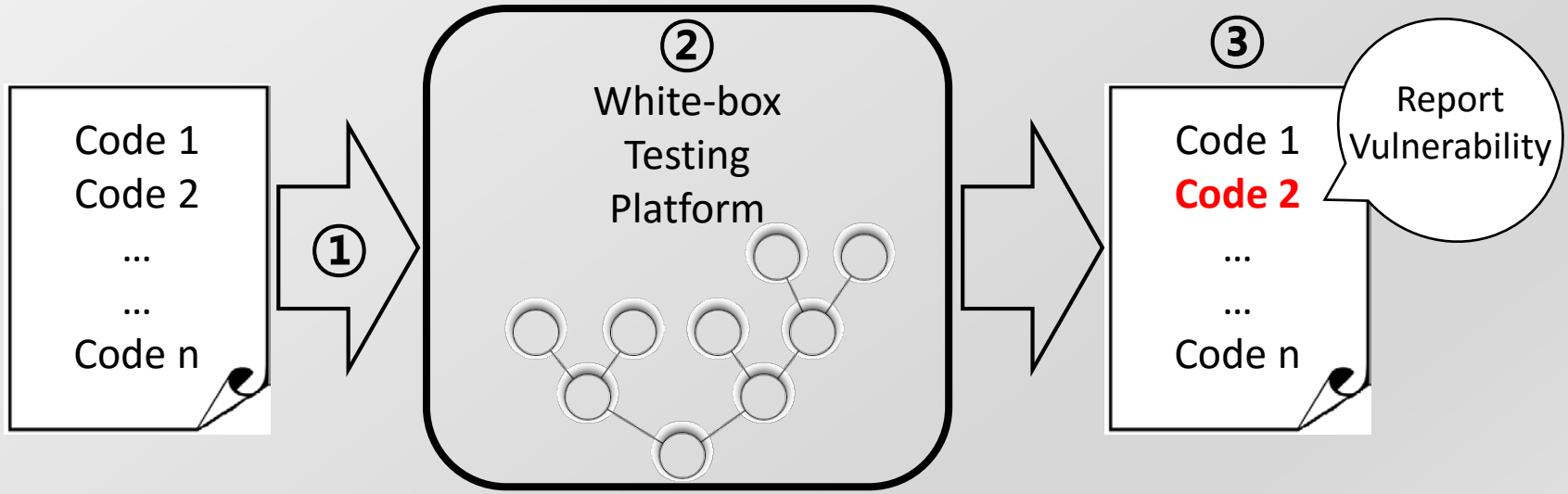
- Optimized Seed Selection (2014, CMU)
  - How to mathematically formulate and reasonabout one critical aspect in fuzzing: **how best to pick seed files** to maximize the total number of bugs found

Research Related (Cont.)

- Scheduling Black-box Mutational Fuzzing (2013, CMU)
  - How to schedule the fuzzings in order to maximize the number of unique bugs found at any point in time
  - The algorithm presented outperforms the multi-armed bandit algorithm in the current version of the CERT Basic Fuzzing Framework (BFF) by finding 1.5x more unique bugs in the same amount of time

What is White-box Testing?

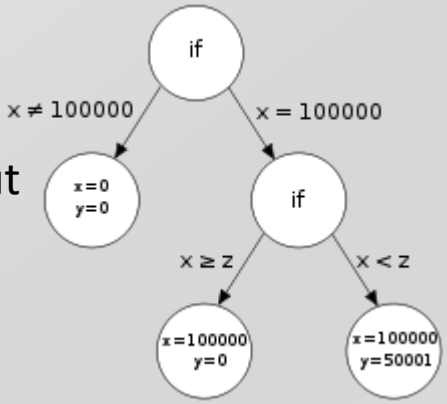
- Method of testing software with the knowledge of internal structure of application
  - ① Source code (e.g. C, C++, …) is given as an input
  - ② Detect and verify the vulnerability in the source code (Static, Dynamic)
  - ③ Report the vulnerability



How to Detect & Verify Vulnerability?

- Example of detection : Code Clone Detection
  - A set of identical or similar piece of code
  - Dangerous when vulnerable code is reused
  - Various detection methods
    - › Token, Abstract Syntax Tree, Dependency Graph based

- Example of Verification : Concolic execution
  - Concolic = CONCrete + symbOLIC
    - › Trace all source code path by using symbolic input
    - › Symbolic input is a representation of input which is used to generate random value



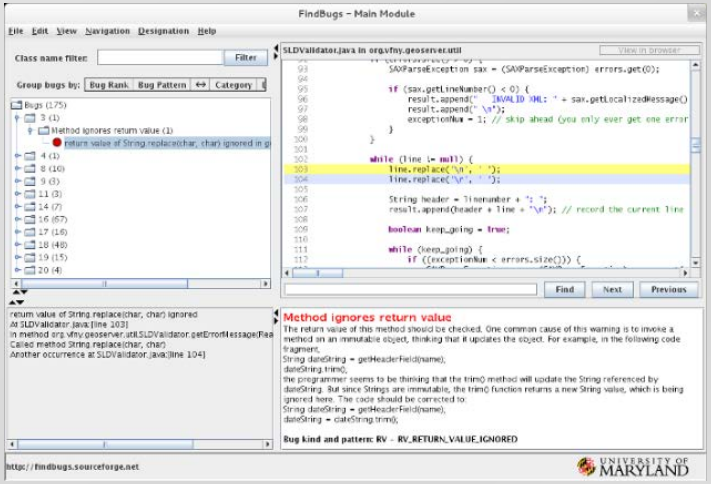
Example of Language-dependent Tools

- Junit
  - Framework for write and run repeatable unit test
  - Test whether program works as expected
  - Work on Java code

```
public void testSum() {
    Calculator calculator = new Calculator();
    assertEquals(30, calculator.sum(10, 20));
}
```

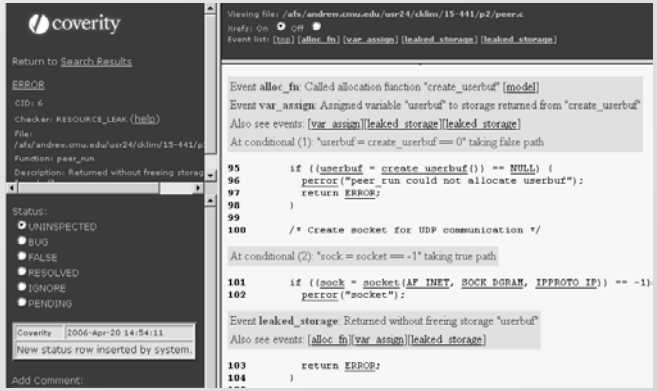
< Example of Unit Test : sum(10, 20) == 30 >

- FindBugs
  - Static code analysis tool
  - Using bug patterns, find the code that are likely to be error
  - Work on Java code

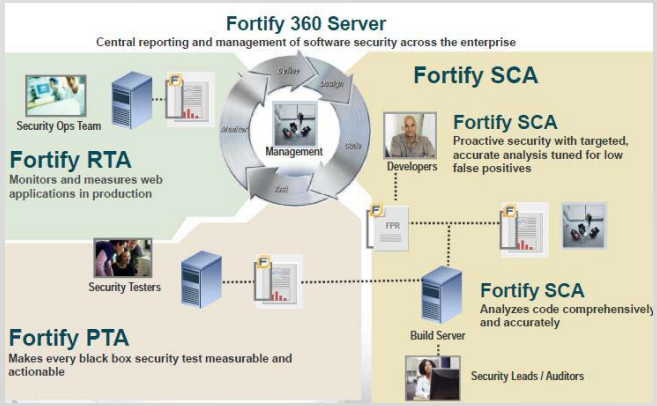


Example of Commercial Tools (1)

- Coverity
  - Perform static and dynamic analysis
  - Analyze code level vulnerability and system fault
  - Support C and C++ code



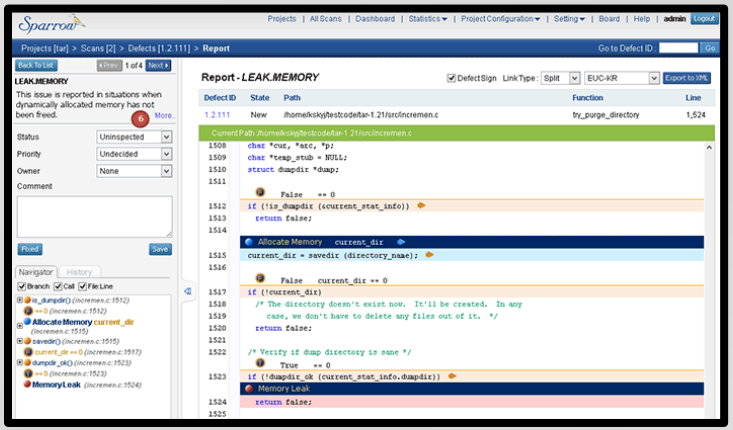
- Fortify
  - Static analysis tool
  - Support vulnerability definition
    - › OWASP, CWE, CERT, …
  - Multilanguage support



Example of Commercial Tools (2)

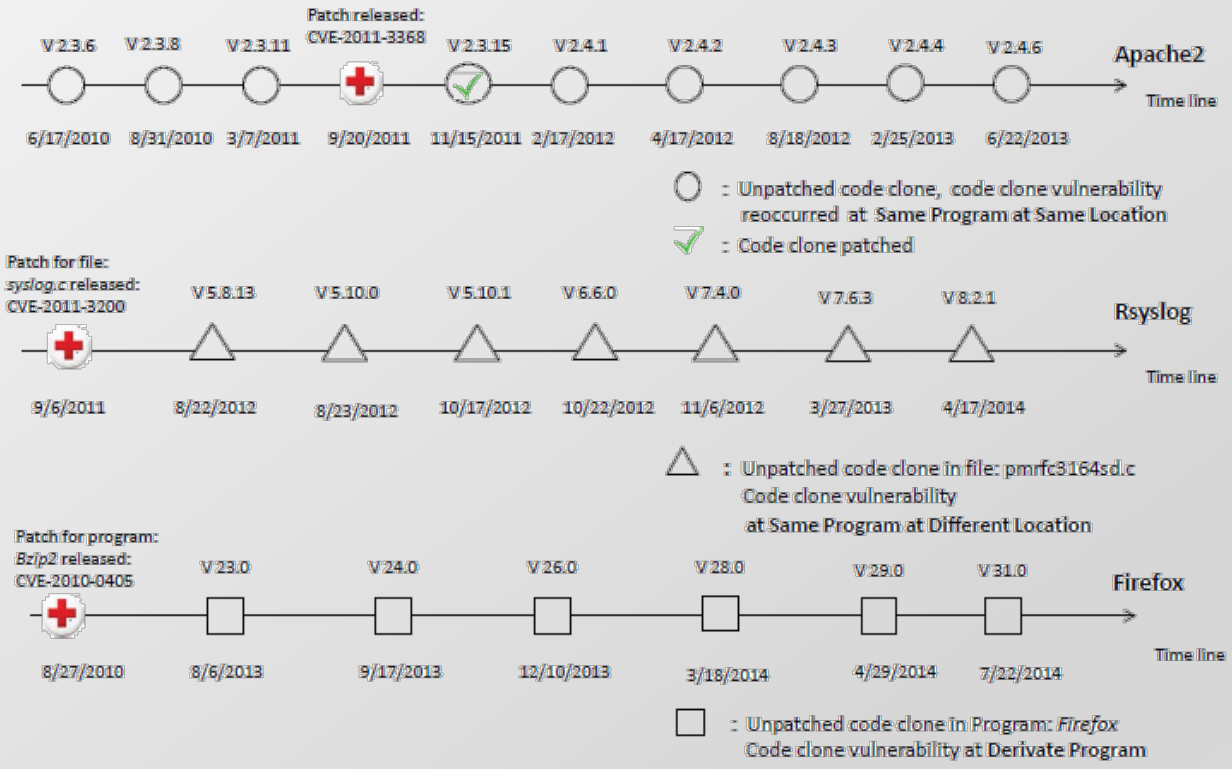
- CodeSonar
  - Static analysis tool
  - Support C, C++, Java code

- Sparrow
  - Semantic based static analysis tool
  - Multilanguage support
    - C, C++, Java, PHP, …
  - Support vulnerability definition
    - CWE, CERT, …



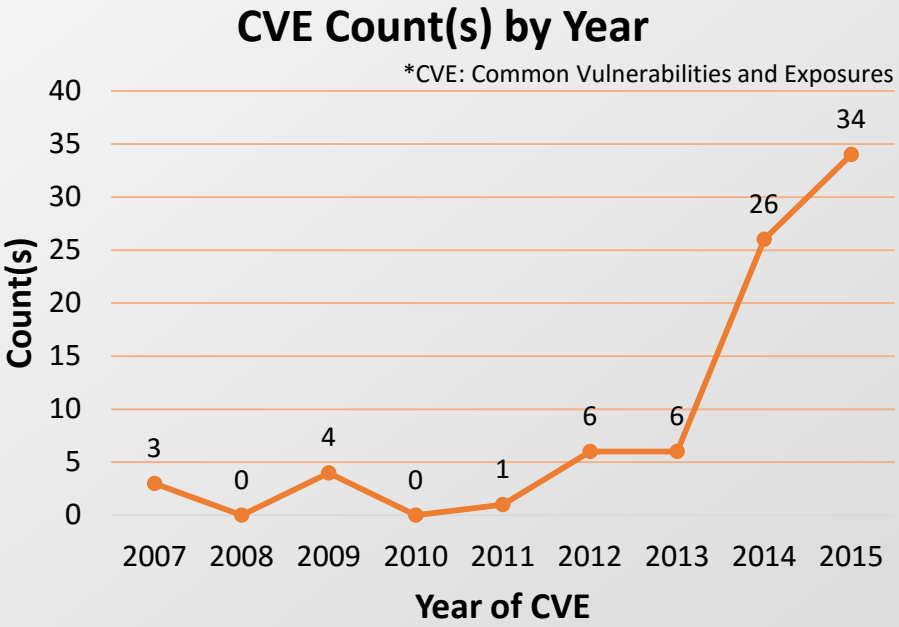
Our Research: Code Clone (1)

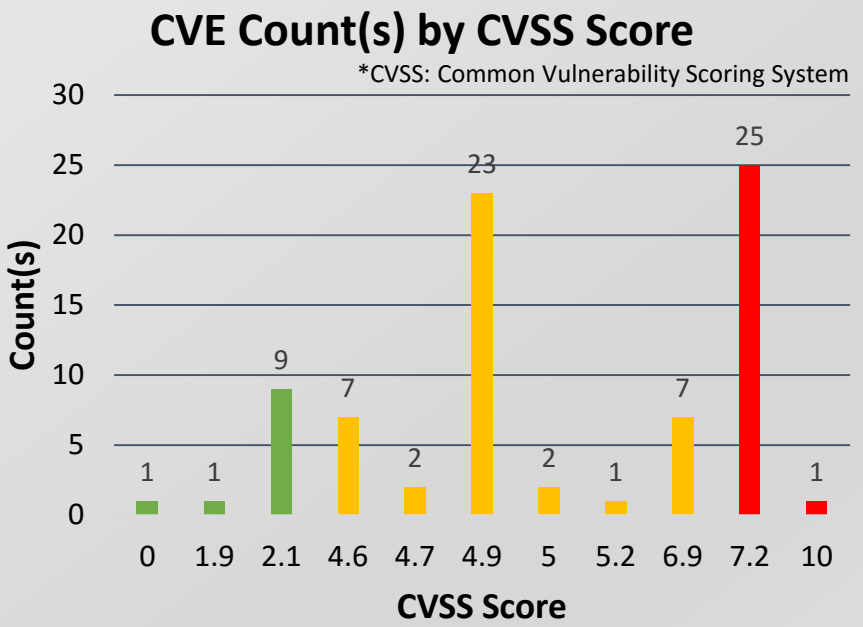
- Code clone vulnerability illustrated
  - Vulnerability still exists even after the patch has been released



Our Research: Code Clone (2)

- Code clone vulnerability is found on latest Smartphone's source code



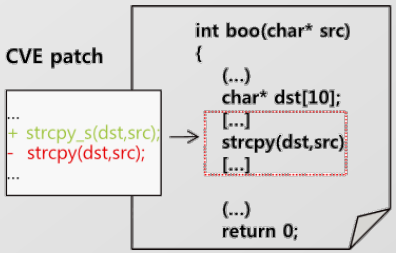


<Statistics of code clone vulnerabilities by CVE Counts>

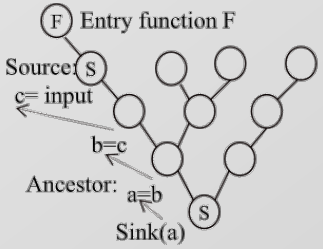
KOREA UNIVERSITY CSSA Center for Software Security and Assurance

Our Research: CLORIFI (1)

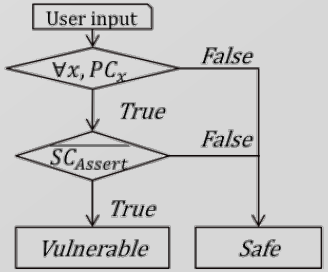
- CLORIFI: Software Vulnerability Discovery using Code Clone Verification
- CLORIFI is composed of 3 steps:
  - Step 1: Code Clone Detection
    - › Use CVE patch and source code to detect code clone vulnerabilities
  - Step 2: Backward Sensitive Data Tracing
    - › Backward data tracing from vulnerable point to the initial user input and insert the test code
  - Step 3: Vulnerability Verification
    - › Verify the vulnerability by using the randomly generated input



(1) Code clone detection



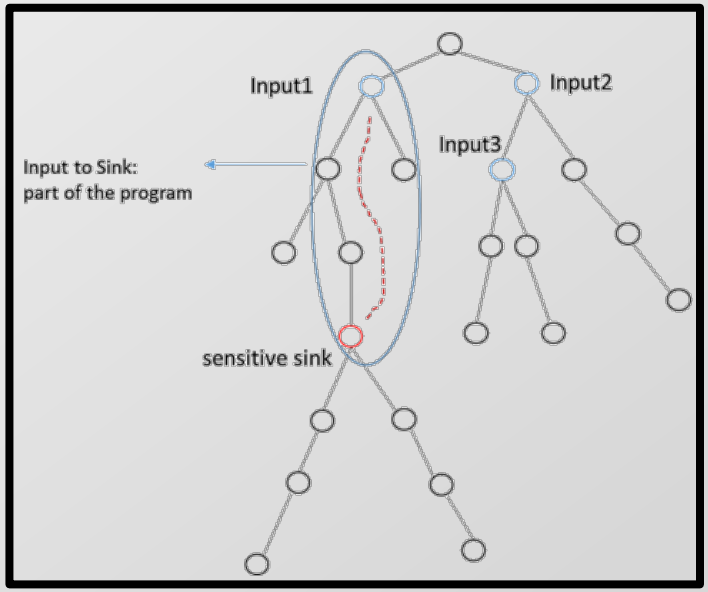
(2) Backward data tracing



(3) Vulnerability Verification

Our Research: CLORIFI (2)

- Main contribution is in Step 2: Backward Sensitive Data Tracing
  - Starting from the sensitive sink, backwardly trace to find its entry point
  - Only consider the input which is related to sensitive sink
    - In the right example, only consider Input 1 among 3 inputs



- Reduce the whole input search space
  - Mitigate path explosion problem occurred in other mechanism
  - Other mechanism analyze the program from its all entry point which cause exponential growth of searching time

Strength of Our Research

- Reduce the false positive
  - Most commercial tools have a huge amount of false positive
  - Reduce the false positive by code clone based detection and perform multiple verification phase
- Consider the priority
  - Find the vulnerability to solve first
  - ex) CVSS Score
- Publication
  - CLORIFI: software vulnerability discovery using code clone verification, Concurrency and Computation: Practice and Experience, Apr. 2015.
  - A scalable approach for vulnerability discovery based on security patches, ATIS 2014. **(Best Paper Award)**
  - Software vulnerability detection using backward trace analysis and symbolic execution, ARES 2013

Center for Software Security and Assurance

http://cssa.korea.ac.kr/

1 Overview

Center for Software Security and Assurance (CSSA) was established in 2015 for the purpose of building an Automated Analysis Platform which enables even non-experts to deal with the known or unknown vulnerabilities of IoT software professionally.

뉴스
학술·연구

고려대, IoT 소프트웨어보안 국제공동연구센터 개소

해외 유수대학과 IoT 보안 원천기술개발 국제공동연구 추진

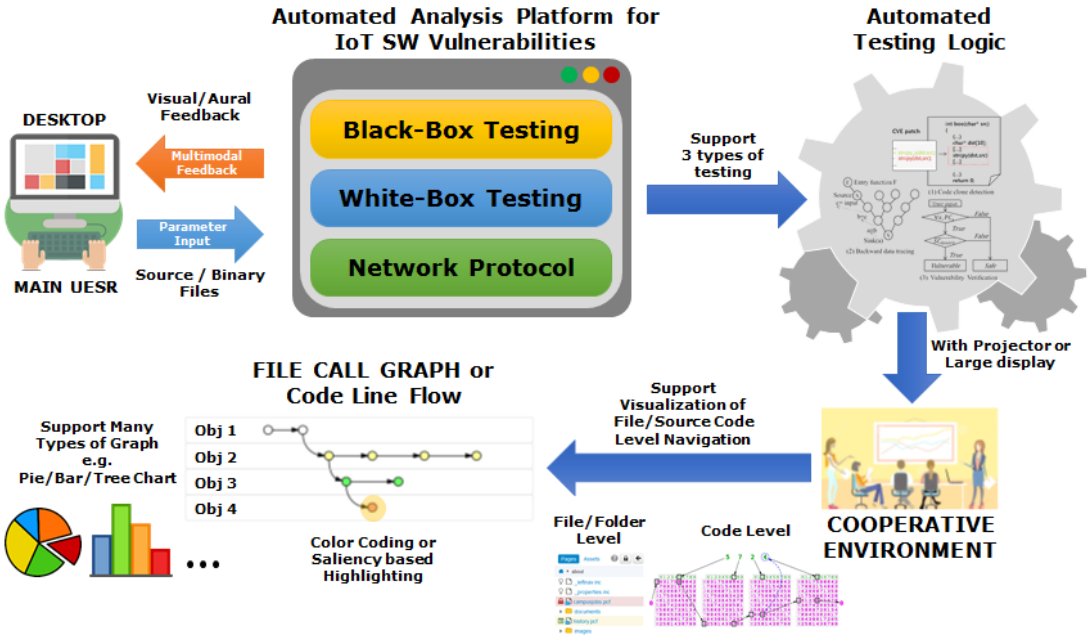
송보배 기자 | 10rim@unn.net

승인 2015.11.20 17:18:50

▲ (앞줄 왼쪽에서 두번째부터)이희조 IoT 소프트웨어보안 국제공동연구센터장, 아드리언 페리그 취리히연방공대 교수, 유혁 정보대학장, 임종인 청와대 안보특보, 송정수 정보보호정책관, 이재일 CP.

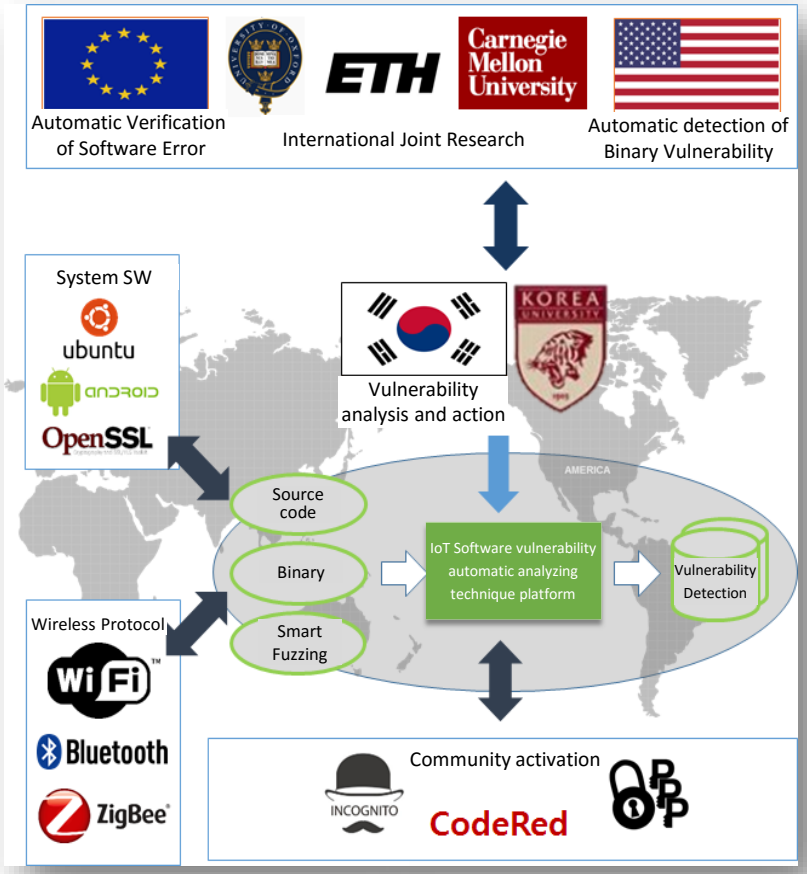
[한국대학신문 송보배 기자] 고려대(총장 염재호)는 지난 19일 IoT 소프트웨어보안 국제공동연구센터(CSSA- Center for Software Security and Assurance, 이하 국제공동연구센터)를 개소했다고 20일 밝혔다.

국제공동연구센터는 'SW컴퓨팅산업원천기술개발' 사업 과제의 일환으로 설립됐다.



< Concept of Automated Analysis Platform for IoT SW Vulnerabilities Detection >

4 International Joint Research



International Joint Research

- Cooperation with research teams with 3 universities of the world top 10 (3rd, 4th, 9th) to develop world-leading source technology
- Hold each online/offline meeting periodically
  - ✓ Found **international joint research center** in Korea University
  - ✓ 2 Months of **research abroad** (Dispatch 3~6 students to 3 overseas universities)
  - ✓ Monthly online research meeting / Quarterly overall result sharing meeting / Yearly offline result announcement meeting

Community Connection & Consultation

- World-leading hacking team CMU PPP & Domestic hacking group CodeRed
  - ✓ Hold seminars and exchange technique
- Incognito: Union of 12 university security clubs
  - ✓ Introduce automatic SW vulnerability analyzing technique and hold vulnerability detection contest
- Meeting with a consulting group
  - ✓ Hold advisory committee 2 times per year
  - ✓ Feedback and verification of usability

6 Research Mission

Center Mission

*"Here have we established the center for everyone in the world who can **secure their own IoT devices** by themselves!"*

Access http://cssa.korea.ac.kr

- Enforcing IoT SW & Network security
- Simple evaluation even for non-experts

- Enabling low cost evaluation of vulnerabilities in small-sized SW

- Secure IoT-based society infrastructure

Conclusion

- Current Status
  - The number of software vulnerabilities is increasing continuously
  - Finding known vulnerabilities is not enough to overcome current security threat

- Way to go
  - Finding/building unknown vulnerabilities discovery on development environment ahead of time
  - Even non-experts could deal with the unknown vulnerabilities of IoT software professionally