Android Open Source Project Java Secure Code Audit

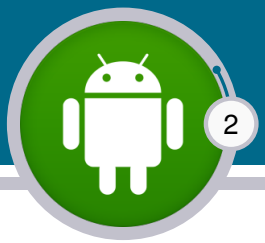
M.Tech Major Project Presentation

Rahul.B
CB.EN.P2CYS14010
Guide: Dr. PRABHAKER MATETI
Wright State University, USA

June 22, 2016

The key points are

- Analysis of source code to discover bugs, security breaches or violations of programming conventions.
- An integral part of the defensive programming paradigm.
- It attempts to reduce errors before the software is released.
- There are coding standards, auditors will use these.
- Automated tools could be used to save time.



The key points are

- 64% of the vulnerabilities in the National Vulnerability Database in 2013 were due to programming errors
- 51% of those were due to classic errors like buffer overflows, cross-site scripting, injection flaws
- Poor implementation of software interfaces (input validation, error and exception handling)
- Inadequate knowledge of secure coding practices

- Android operating system.
- Android Open Source Project(AOSP).
- Languages Used in AOSP.
- AOSP total Lines Of Code.

AOSP Source Code
Lines of Code

Language	files	blank	comment	code
C++	44684	1985251	1695899	11232371
C	18014	1129390	1542888	7176099
XML	17102	115738	1202217	6876603
C/C++ Header	51947	1416061	2398732	6565432
Java	39669	1183108	2471951	6031537
HTML	10061	606983	99627	2508613
Javascript	5235	122116	260245	1241356
Python	6149	223702	296572	923091
Bourne Shell	1452	140010	134564	877990
Assembly	3937	105294	106096	794627
Expect	829	13736	7781	284717
m4	516	23359	6415	215578
Objective C++	1401	39755	50101	176134
Perl	439	22451	23598	141361
CSS	607	16240	7431	94310
Objective C	1432	22644	68845	83826
IDL	1089	8578	0	74960
Maven	681	3917	5869	68039
D	2417	16802	0	63004
C#	448	8855	17755	53674
make	940	11529	11931	48835
Pascal	48	4902	26425	36224
Teamcenter def	148	3127	997	26871
Fortran 77	68	19	15419	26495
Ruby	92	5420	3533	23190
CMake	500	3916	4165	21650
Rust	605	5513	6536	21270

Figure: AOSP lines of code

Coding Standards

- **Computer Emergency Response Teams (CERT)** - Separate guidelines for C, C++, Java, Android, Perl. [2]
- **Motor Industry Software Reliability Association (MISRA)**- For developing safety-related electronic systems in road vehicles, telecom, aerospace and other embedded systems. [3]
- **Open Web Application Security Project (OWASP)** - It has coding standard, documentation and tools for web application security [4]
- **Payment Card Industry Data Security Standard (PCI DSS)**- is a proprietary information security standard for Payment Card Industry [7]

The SEI-CERT Coding Standard

- Computer Emergency Response Team
- Primary goals are to ensure appropriate technology and systems-management practices are used to resist attacks
- Coding Standards for C, C++, Java, Perl, Android
- For C 194 Rules and 77 Recommendations
- For Java 122 Rules and 180 Recommendations [8]

The SEI-CERT Coding Standard

Android Java secure coding standard

Rule Id	Name	Rules	Recommendations
00	Input Validation and Data Sanitization (IDS)	14	6
01	Declarations and Initialization (DCL)	03	-
02	Expressions (EXP)	07	02
03	Numeric Types and Operations (NUM)	14	04
04	Characters and Strings (STR)	-	-
05	Object Orientation (OBJ)	12	-
06	Methods (MET)	13	03
07	Exceptional Behavior (ERR)	10	01
08	Visibility and Atomicity (VNA)	06	-
09	Locking (LCK)	12	3
10	Thread APIs (THI)	06	05
11	Thread Pools (TPS)	05	05
12	Thread-Safety Miscellaneous (TSM)	4	03
13	Input Output (FIO)	16	07
14	Serialization (SER)	12	-
15	Platform Security (SEC)	08	04
16	Runtime Environment (ENV)	06	02
17	Java Native Interface (JNI)	04	-
49	Miscellaneous (MSC)	8	01

Figure: Categories in the CERT Android Java secure coding standard

Objective
Aim

- To identify maximum number of coding violations in AOSP.
- Make an audit report with better description.

Abstract Syntax Tree (AST)

- Tree representation of the source code written in a programming language [5]
- The result of the syntax analysis phase
- Contains many nodes like ConstantNode, VariableNode, AssignmentNode, ExpressionNode,
- Traversing AST

AST Traversing
Visitor pattern

- Perform Type checking, Code generation, Variable declaration, Syntax analysis on AST
- These actions should handle each type of node separately
- The Visitor pattern[6] was introduced to address the above problem
- Encapsulate a desired operation in a separate object, called a visitor.
- Some API's will do both traversing and parsing. (Eg: Eclipse JDT) [9]

Apply Rules On AST

- CERT rules are English sentences.
- Only examples of Non-compliant and compliant code.
- Example: MSC01-J. Do not use an empty infinite loop [1]

Apply Rules On AST

MSC01-J. Do not use an empty infinite loop [1]

An infinite loop with an empty body consumes CPU cycles but does nothing.

Non-compliant Code Example

```
public void nop()
     {
    while (true)
              {
                  }
     }
```

Apply Rules On AST

MSC01-J. Do not use an empty infinite loop [1]

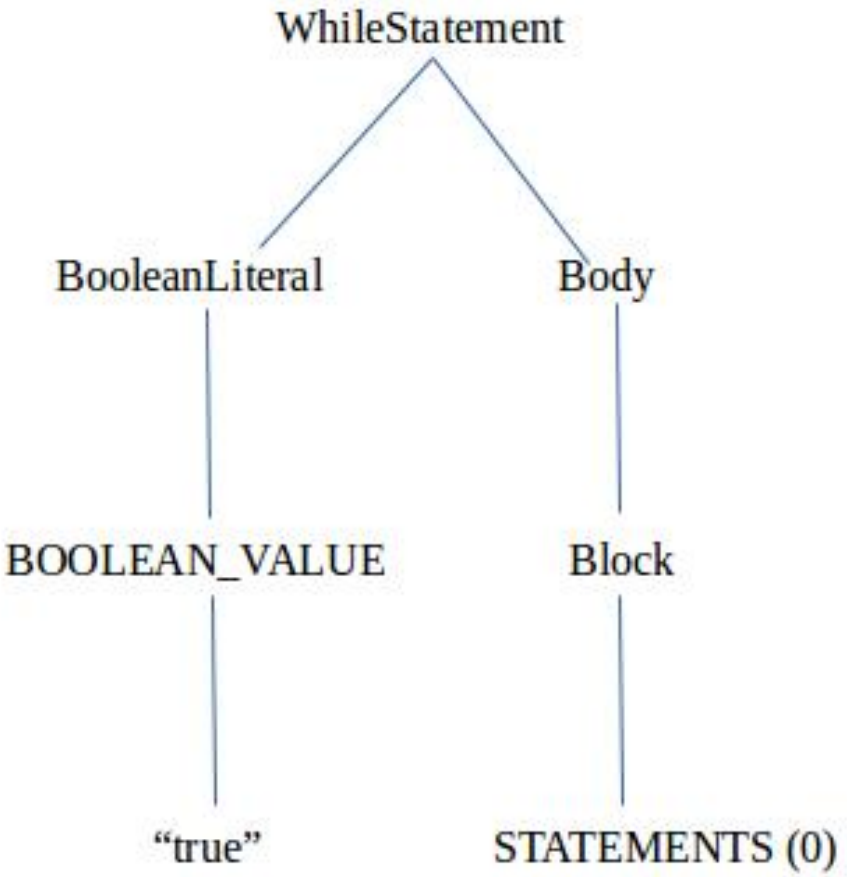


Figure: MSC01-J AST representation

Proposed System

- An algorithm for each CERT rule.
- Implement the algorithm.
- An Executable Binary with different options.
- Implement maximum number of CERT rules.
- Audit AOSP and make a well defined report.

Outline of solutions

- Convert the given source code to AST.
- Applicability of the rule.
- Semi-algorithmically explanation.
- A well-structured report.

Implementation and Result

The syntax is,

```
<rast> <argument> <filename/dir path> <optional arguments>
```

The arguments are:

```
-b For batch conversion
-s For single file conversion
-h For help page
-f For semi auto fix
<optional arguments> For Checking/Fixing a particular rule.
```

Implementation and Result
Executable Binary


```
rahul@UnknownPC:~$ rast -b /Source/framework/
```

Figure: Executable Binary with different options

```
rahul@UnknownPC:~$ rast -s HelloWorld.java
```

Figure: Executable Binary with different options

```
rahul@UnknownPC:~$ rast -r HelloWorld.java MSC01J
```

Figure: Executable Binary with different options

Conclusion

The unique things in the work

- The tool is lightweight, nearly 15mb is the file size.
- Automation of a large number of files is possible.
- It will generate a self-explanatory report for auditors.

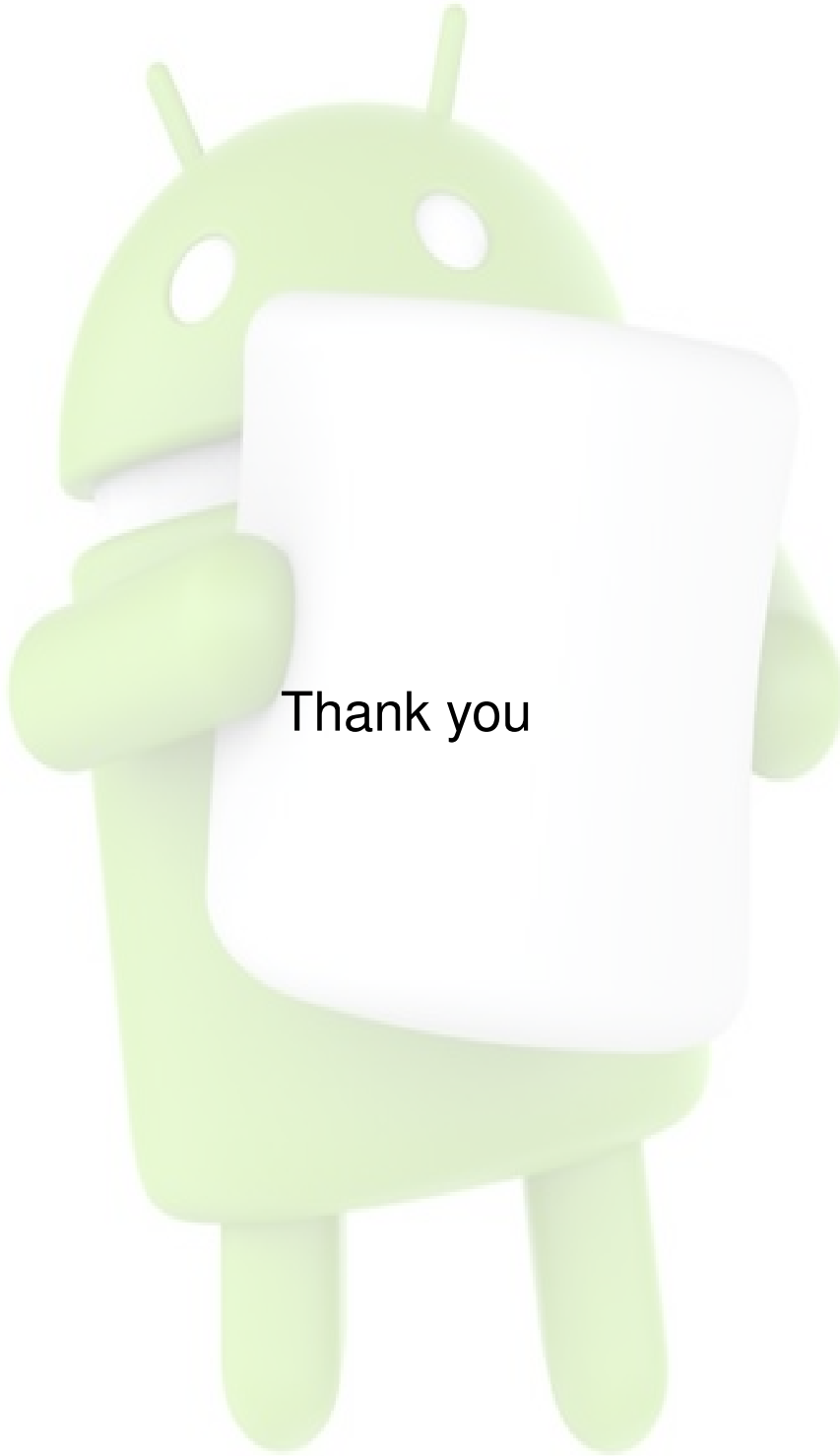
Conclusion

What is my contribution.

- More than 3500+ lines of code (HTML+Java+Test codes)
- Implemented 20+ CERT rules
- A well-formed report

Future Work

- Implement more rules.
- Correction of AOSP.


Thank you

References I

[1] Jose Sandoval Chaverri.

Msc00-j.

https://www.securecoding.cert.org/confluence/display/java/MSC00-J.+Use+SSLSocket+rather+than+Socket+for+secure+data+exchange, Feb 10, 2016.

[2] Fred Long, Dhruv Mohindra, Robert C Seacord, Dean F Sutherland, and David Svoboda.

The CERT Oracle Secure Coding Standard for Java.

Addison-Wesley Professional, 2011.

[3] MISRA.

Guidelines for the use of the c language in critical systems.

http://caxapa.ru/thumbs/468328/misra-c-2004.pdf.

[4] MITRE.

Mitre.

https://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance/secure-code-review.

References II

[5] S. Nair, R. Jetley, A. Nair, and S. Hauck-Stattelmann.
A static code analysis tool for control system software.
In *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*, pages 459–463, March 2015.

[6] oodesign.com.
Visitor pattern.
http://www.oodesign.com/visitor-pattern.html.

[7] Helen Gill Paul E. Black and W. Bradley Martin (co-chairs) Elizabeth Fong (editor).
Proceedings of the static analysis summit.
https://samate.nist.gov/docs/NIST_Special_Publication_500-262.pdf.

[8] Robert C Seacord.
The CERT C secure coding standard.
Pearson Education, 2008.

[9] IBM Ottawa Lab Thomas Kuhn, Eye Media GmbH Olivier Thomann.
Abstract syntax tree.
http://www.eclipse.org/articles/Article-JavaCodeManipulation_AST, November 20, 2006.