Suggesting Secure Implementation to Vulnerable Code Snippets on Stackoverflow.

1 INTRODUCTION

```
private byte[] encrypt(byte[] raw, byte[] clear) {
   ...
   Cipher cipher = Cipher.getInstance("AES");}
   // Cipher cipher = Cipher.getInstance("AES/CBC/
     PKCS5Padding");
   ....
   return encrypted
}

```

Listing 1: A real code snippet taken from Stackoverflow. I want to build a tool which after analyzing the code snippet will highlight the part of the code that is insecure and suggest an alternative secure implementation as showed in the figure.

In this project I want build a static analysis tool which will achieve the following.

- Analyze the code snippets from Stackoverflow for identifying out which part of the code is vulnerable and show warning signs to highlight that part of the code to the developer.
- When developer clicks on the warning sign a secure implementation while be shown to the developers. In case of failure of building generating a secure implementation, the tool will show insightful/helpful messages explaing why this part of the code is flaged as insecure.

2 WHY THE PROBLEM IS INTERESTING

The problem is interesting for two reasons.

Difficulty of writing crypto code securely. Writing/implementing cypto code securely is a diffculut task for programmers. Any potential bug in crypto code can lead to serious vulnerablities open for attackers. Even so unlike other code, crypto code can be insecure even if it works perfectly on traditional test-suite's input/output which is used only to prove the implementation correctness of the program.

Online platforms roles in spreading insecure code. Online programming discussion platforms such as Stack Overflow have a rich source of ready to use code snippets for software developers. It is the defac-to place where developers go to find solutions of their problems and turn to the community for answers to their problems. Insecure code snippets found on Stackoverflow itself is not a serious problem. However, Fischer et al. has shown that developers have a tendency to directly copy paste code form Stack Overflow [2]. Therefore there are chances that any insecure code snippets posted on Stackoverflow can potentially find it way into production level code. To make matters worse, Meng et al. [3] has showed that many accepted answers on Stackoverflow have seriously insecure code and often-times given by users having high reputation. This adds to the problem copy pasting vulnerable code from online platform

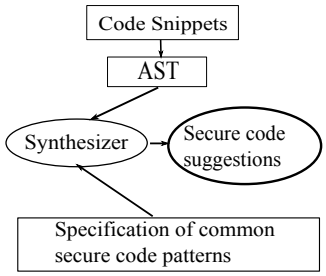

Figure 1: Proposed Methodology (Redraw the figure)

and furthermore increases the chances of the insecure code snippet being trickled down to production level code. Unfortunately there is not state-of-the-art tool to analyzie if a code posted by developer on Stackoverflow is secure or not. In absense of such tool, Stackoverflow is potentially contributing as a major source of vulnerability in production level code.

A static tool which can identify which part of the code snippet is insecure and suggest secure alternatives can help stopping the flow of insecure code from Stackoverflow to production level code.

3 PROPOSED METHODOLOGY/MILESTONES

As of now I am not really sure how to achieve the proposed goals described in §1. Especially how state-of-the-art synthesis techniques (e.g., program sketching) can be useful. However I am confortable analyzing insecurity of crypto code and therefore proposing the following methodology.

My main idea is to convert the insecure and secure code snippets to AST and then apply synthesis to convert the insecure AST to secure AST *somehow*. Again I am not sure about this synthesis part.

3.1 Data Collection (completed)

I have collected code snippets of Stackoverflow from two existing sources. They are analyzed in [2, 3] and publicy avaialble. They contain insecure code snippets of Java and Andrioid.

I have compiled and arranged the coee snippets from two sources here [1]. In future stages of the project I will add insecure codes from other languages as well using Stackoverflow's API.

3.2 Identify Insecure Code Patterns (completed)

I have manually analyzed the insecure code in [2, 3]. I have identified around 10 common insecure patterns which are most frequently occuring in them to an idea how secure code should look like. These 8 insecure partterns are briefly presented in Table. 2.

[1] https://github.com/islamazhar/CS-703-Project/tree/master/insecure-code

No	Vulnerable insecure pattern	Way to secure
1	AES default encryption mode ECB which is vulnerable to side channel attacks	Using CBC encryption mode
2	Accetping all certificates in TrustManager by implementing empty methods	Accepting only those certs which are in the trust chain
3	IV/Keys lacking randomn seed	IV/keys' seed should be generated from pesudo random functions
4	Keys having weak lengths	Using keys' having sufficient large length
5	Turing off CSRF protection	Do not turn of CSRF protection
6	Hard coded simple keys	Avoid using simple hard coded secret keys
7	Using Broken HASH MD5	Avoid using broken hash MD5
8	Using insecure HTTP	Mandating usage of HTTPS

Figure 2: Common insecure pattern and ways to secure them.

3.3 Converting Code Snippets to AST (completed)

To analyze the code snippets for finding insecure parrterns, I have to convert these code snippets to Abstratc Syntax Tree (AST). However these code snippets are partially complete (i.e., missing classes, method implementations). Hence I have used a static analysis tool for partial java programs named PPA [1] to convert the code snippet to an intermidate representation Jimple.

Jimple [5] is a 3-address intermediate representation that has been designed to simplify analysis. Jimple was inspired from SIMPLE an AST to represent C statements. We can feed the Jimple representation of the code snippet to soot – a state-of-the-art we can traverse the jimple code to generate AST.

Each node (unit) of the generated AST from jimple representation consists of around 30 types of Stmt or Inst[2] – which are in the grammer of Jimple. This Stmt or Inst are basically single/multiple line code fragment of the code snippets. Now the problem simplifies to extracting those Stmt or Inst which are insecure..

3.3.1 Experiments. For now I am currently focused on the rule no 1 (as shown on Table 2). This rules says that using default encryption mode ECB for AES encryption is vulnerable to side channel attacks. To prevent this we need to specify the a secure mode of the AES encryption (e.g., CBC, CFB, OFB, CTR). So we are searching for node in AST which has a statements such as `Cipher encipher = Cipher.getInstance("AES");` and replacing them with `Cipher cipher = Cipher.getInstance("AES/CFB/NoPadding");`. This is represented by `InvokeStmt` in Jimple. Therefore I am extracting those `InvokeStmt` and if the arguments passed to the invoke function is simply `AES`, I identify them as a insecure pattern. Currently I am justing showing a warning message with the line number saying *"Use of default encryption mode in AES is vulnerable to side channel attacks. Consider using CBC/CFB encryption mode".*

I have analyzed first 30 insecure and secure code snippets related to rule from the dataset which has a total of 1.3K code snippets. Among 12 code snippets which has the insecure parrtern, I can detect 5 of them. For others 7 which I am unable to detect, I am either failing to produce the Jimple representation of those code snipeets as PPA is throwing exceptions or the default mode is passed as a constant. My implemenetation is currently conservative it does not produce warning when it can find the exact patterns I am looking for.

[2]https://www.sable.mcgill.ca/soot/doc/soot/Unit.html

3.4 Comparing AST of secure and insecure code / Repairing insecure AST from secure AST examples (incomplete)

I am currently focused on implementing on detecting the other 7 rules. This would be challenging as they are staightforward as rule 1. For example, rule no 2 have anonymous class and as far as I know Jimple code does not support anonymous class which can be a big problem. I am also going through the dataset of insecure code snippets to understand how same insecure pattern is written in different ways.

I have read the paper which you have suggested [4]. This paper seems really relevent to the project. The difference is that we already have the quick fixes and do not need to learn them. But the modifying the AST part does seems co-related with what the project proposes. I will follow up with some updates on Monday office hour.

REFERENCES

[1] Barthélémy Dagenais and Laurie Hendren. 2008. Enabling static analysis for partial java programs. In *Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*. 313–328.

[2] Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. 2017. Stack overflow considered harmful? the impact of copy&paste on android application security. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 121–136.

[3] Na Meng, Stefan Nagy, Danfeng Yao, Wenjie Zhuang, and Gustavo Arango Argoty. 2018. Secure coding practices in java: Challenges and vulnerabilities. In *Proceedings of the 40th International Conference on Software Engineering*. 372–383.

[4] Reudismam Rolim, Gustavo Soares, Rohit Gheyi, Titus Barik, and Loris D'Antoni. 2018. Learning quick fixes from code repositories. *arXiv preprint arXiv:1803.03806* (2018).

[5] Raja Vallee-Rai and Laurie J Hendren. 1998. Jimple: Simplifying Java bytecode for analyses and transformations. (1998).