International Journal of Engineering Research ISSN:2319-6890)(online),2347-5013(print)
Volume No.5 Issue: Special 4, pp: 790-991 20 May 2016

A Tool for Extirpateing Defenseless Software Vulnerability Superintend in Programs*

Ravinder kumar Joshi[1],Prajna Bhavi[2], Dr. Chandramouli H[3],Dr. B.R.Prasad Babu[4]
MTech Student, Department of CSE ,R&D Centre, East Point College of Engg & Technology1
MTech Student, Department of CSE ,R&D Centre, East Point College of Engg & Technology[2]
Professor, Department of CSE ,R & D Centre, East Point College of Engg & Technology[3]
Professor & Head of Department of CSE, East Point College of Engg & Technology[4]
[1]ravu3718@gmail.com,[2]prajna.rb@gmail.com,[3]hcmcool123@gmail.com ,[4]brprasadbabu@gmail.com
*Project funded by the Government of Karnataka under VGST Scheme (2015-2016)

Abstract

Research has revealed that Software can inherently have weaknesses which can be exploited by hackers. The global IT community has formed a group who are trying to detect constructs in a program which makes the software weak. These programming constructs which make software weak are grouped as in CWE [i].

Keywords: Vulnerability, SecCheck, CWE, Degree of Insecurity, ISM

I. INTRODUCTION

The security of the software menace at various stages in different ways, involuntarily due to mistakes made by developers or by deliberate piracy. Design and implementation of software of the poor are the main causes of most security vulnerabilities [i] [ii] threatened .security can also be produced from websites and web applications (web applications). They need to be protected from all threats and other data center assets used to host Web sites and related systems.

We carried out aspects of the study of the recent weakness evaluated over 250 web applications, e-commerce, online banking, the collaboration of the company , and management positions in the supply chain and found that 92 % to web applications less vulnerable to such attacks by pirates.

It is doubtful whether todays information security techniques able to protect critical software systems, unless security mechanisms be an inherent part of the software. Java has emerged as the language of choice for building large complex web-based systems, partly because of the security language features that prohibit direct memory access and eliminate problems like buffer overruns.

This document discusses the vulnerabilities that are injected into the Java programs during the coding phase and describes a tool developed to detect vulnerabilities and warn the developer for these Tool developed by the authors , and called SecCheck reveals weaknesses in each program Java.
Investigate how software security can be improved though robust design and coding. While work has been done by authors on detection of vulnerabilities during Code Review, [iii][iv] more work needs to be done on detection of Security lapses during Design and Coding.

Further presently there are no metrics suitable to specify the degree of insecurity in a program. While the authors [iii][iv] have proposed metrics based on severity figures in [1], more experimental work needs to be done Develop manpower in the area of Software Security and use of Tools for detection of software vulnerabilities

Due to processing of critical information, it is important that software is protected against malicious attacks and other risks so that it continues to function correctly even under such potential threats. Threats can be identified using application threat modelling and then evaluated by vulnerability assessment. Nowadays, lot of attention is being given on building secure software and on detecting vulnerabilities by static analysis

Many types of vulnerabilities exist in software systems injected in design and in implementation phases, such as local implementation errors, inter procedural interface errors (such as a race condition between an access control check and a file operation), design-level mistakes (such as error handling and recovery systems that fail in an insecure fashion, topology of the web of links, deep linking, unspecified image dimensions), and object-sharing systems . Efforts are required during design and implementation to make the software secure and to protect the software against malicious attacks and other risks.

This paper discusses vulnerabilities that are injected in Java programs during coding phase and describes a tool which developed to detect the weaknesses and solutions

II. Material and Methodology

Common Software Vulnerabilities that occur in Java programs as discussed in CWE [i] are:

1. Absolute path traversal
2. finalize() Method Declared Public
3. Unrestricted Upload of Files with Dangerous Type
4. Manipulating Input to File System Calls

A. Absolute path traversal

Any app which uses exterior contribution to build a trail name that should be in inadequate information bank, rather it wont correctly counterbalance unconditional lane progression as 'abs/path' that can determine to a spot outside.[v]

Solution

The following code demonstrates the unrestricted upload of a file with a Java servlet and a path traversal vulnerability. The action attribute of an HTML form is sending the upload file request to the Java servlet.

```
 <form action="FileUploadServlet" method="post
 enctype="multipart/form-data">
 Choose a file to upload:
<input type="file" name="filename"/>
<br/>
<input type="submit" name="submit" value="Submit"/>
</form>
```

When submitted the Java servlet's doPost method will receive the request, extract the name of the file from the Http request header, read the file contents from the request and output the file to the local upload directory.

B. Finalize() Method Declared Public

A program should never call finalize explicitly, except to call super.finalize() inside an implementation of finalize().because it is declared with public access.[vi]

Solution

If you are using finalize() as it was designed, there is no reason to declare finalize() with anything other than protected access.

C. Unrestricted Upload of Files with Dangerous Type

The "unrestricted file upload" term is used in vulnerability databases and elsewhere, but it is insufficiently precise. The phrase could be interpreted as the lack of restrictions on the size or number of uploaded files, which is a resource consumption issue.[vii]

Solution

The following code intends to allow a user to upload a picture to the web server. The HTML code that drives the form on the user end has an input field of type "file".

Once submitted, the form above sends the file to upload_picture.php on the web server. PHP stores the file in a temporary location until it is retrieved (or discarded) by the server side code.

D. Manipulating Input to File System Calls

This attack deals with an attacker who manipulates inputs to the target software which the target software passes to file system calls in the OS.[viii]

Solution

Utilize an appropriate mix of white-list and black-list parsing to filter equivalent special element syntax from all input.

Table 1: Severity of Vulnerabilities

Type of Vulnerability = i	**Severity= wi**
finalize() Method Declared Public	04
Absolute Path Traversal	16
Unrestricted File Upload with DangerousType	10
Manipulating Inputs to File System Calls	03

E. Degree of Insecurity

Each of the weaknesses discussed in this paper has been assigned a severity level defined in CWE as shown in Table 1. We use a metric for calculating the Degree of Insecurity (referred to ISM)

ISM= $\sum_{i=1}^{m} Wi * N i$ where,

ISM stands for the Degree of Insecurity,
i is the Type of Vulnerability 1,2,..m
W_i is the Severity of Vulnerability in the software
Ni is the frequency of occurrence of vulnerability i.

III. RESULTS AND TABLES

We used the tool SecCheck for detection of vulnerabilities on two classes of Java programs. Table 2 shows the programs written by the professionals taken from different vulnerability tracking sites including few from Common Weakness Enumeration (CWE) site. Table 3 shows the programs written by students of an engineering college. The results of measurements in these programs are given in Tables 2 and 3 with the vulnerabilities detected and the *Degree of Insecurity*

Table 2: Degree of Insecurity calculated in Programs written by Java professionals

Name Of Program	Source	Size	ISM
Concurrency	http://stackoverflow.com/questions/6084722/java-threads-concurrency-delta-off-by-one-error	3KB	48
Countdivisor	http://math.hws.edu/javanotes/c3/s4.html	3KB	13
Bouncingline	http://www.java-examples.com/generate-bouncing-lines-using-applet-example	5KB	81
Javapyramid	http://www.java-examples.com/java-pyramid-3-example http://www.java-examples.com/prime-numbers-java-example	2KB	36

Comparison of the *Degree of Insecurity* in the two classes reveal that the Java programs written by students have a higher *Degree of Insecurity* than those written by professional programmers and therefore are more prone to security attacks.

Table 3: Degree of Insecurity Calculated in

Program Name	Size	ISM
ClientA	13 KB	313
Router	30 KB	615
Server	36 KB	756
Dist	31 KB	525
Logger	19 KB	213
Arithmetic	32KB	659
Average Value of ISM Calculated from these Programs		**513.5**

Programs written by Students

A. Working Of SecCheck

The tool takes as input any Java program and scans to identify the vulnerabilities. If any vulnerability is detected then it displays warning message and suggests steps for its mitigation.

The steps followed are :

1. Select the input Java program
2. Select from the drop down list all types of vulnerabilities intended to be detected

As shown in Figure 1, for a Java program given as an input to the Tool, it displays type of vulnerability found and the place of its occurrence. It also gives the Degree of Insecurity in the input program The functional modules in the Tool are shown in Figure 1

Scanner*:* This module scans each line of source code one by one.

Pattern Matching Module**:** After scanning, the tool compares each line to find out if it contains a set of keywords which makes the program vulnerable to security threats. This is done by matching each line with the list of strings stored in a database.

Display Module: If there is a string match then a warning message is flagged to the user.
After the entire program is scanned, the *Degree of Insecurity* is calculated and displayed.

Mitigation Module: After the program is scanned, and Degree of Insecurity is found out, the next step is to mitigate the vulnerabilities detected. The tool provides solution for each of the vulnerabilities.

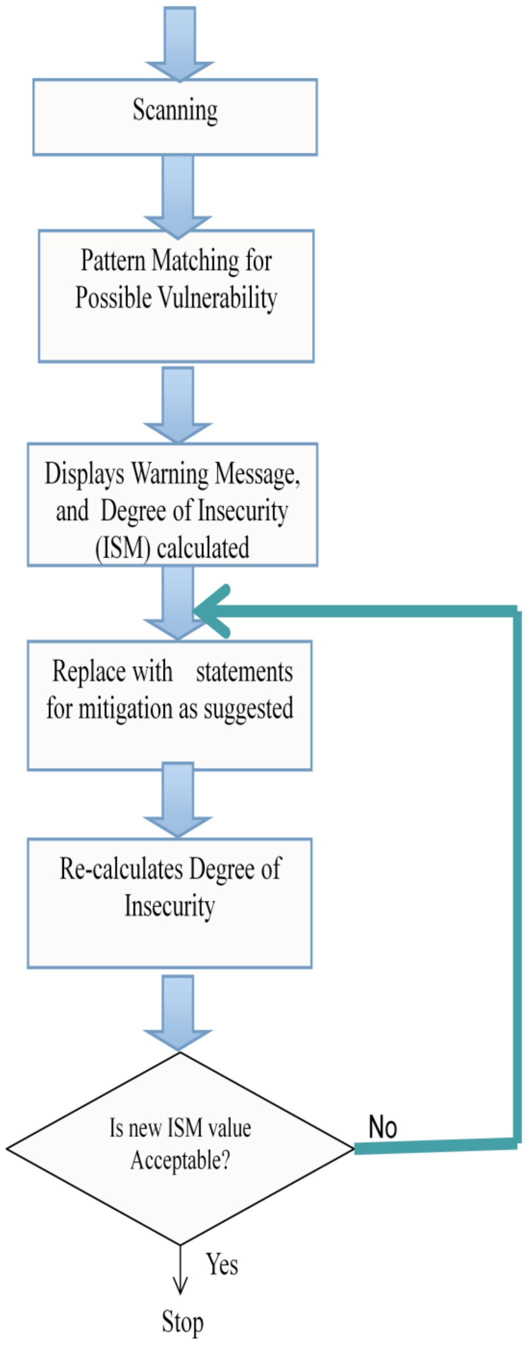
Scanning
Pattern Matching for Possible Vulnerability
Displays Warning Message, and Degree of Insecurity (ISM) calculated
Replace with statements for mitigation as suggested
Re-calculates Degree of Insecurity
Is new ISM value Acceptable?
No
Yes
Stop

Figure 1: Functioning of the Tool

IV. CONCLUSION

Vulnerability is weakness in software. The causes of "weakness" can be faults in design and code and allows an attacker to reduc system's information assurance.

Presence of vulnerabilities in software makes it necessary to have tools that can help programmers to detect and correct them during development of the code.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to *Dr. B M Satish*, Principal, East Point College of Engineering and Technology, *Dr. Prasad Babu B R*, HOD, Department of CSE, *Dr. Chandramouli H*, Professor and PG Coordinator, Department of CSE for there guidance to complete this paper work on time.

References

[1] www.cwe.mitre.org

[ii] Asoke K.Talukder,and Manish Chaitanya."Architecting Secure Software Systems", CRC Press, 2009

[iii] Priyadarshini. R, Anirban Basu and Sushma. S 'SecCheck: A Tool to Detect Vulnerabilities in Java Code' ICODC International Conference on On-Demand Computing, Bangalore, Nov 15-16, 2012

[iv] Nivedita Ghosh and Anirban Basu 'WebCheck: A Tool to Detect Weaknesses in Java Web Applications' ICICE-2013 International Conference on Information and Communication Engineering, Bangalore, June 28-29, 2013

[v]Absolute path traversal
http://cwe.mitre.org/data/de nitions/36.html

[vi] finalize() Method Declared Public:
http://cwe.mitre.org/data/definitions/583.html

[vii] Unrestricted Upload of File with Dangerous Type:
http://cwe.mitre.org/data/definitions/434.html

[viii] Manipulating Inputs to File System Calls: http://capec.mitre.org/data/definitions/76.html

[ix] Asoke K.Talukder,and Manish Chaitanya."Architecting Secure Software Systems", CRC Press, 2009

[x] http://pmd.sourceforge.net/

[xi] http://www.programmingresearch.com/

[xii] http://msquaredtechnologies.com/

PROFILE:

Mr.Ravinder kumar joshi is M.Tech Scholar in Computer Science and Engineering at East Point College of Engineering and Technology,VTU Belgaum.He Attended and presented more than 4 papers In National and International Conferences in various colleges,He attended the Handson workshop in CSI Bangalore Chapter.My Research areas are Big data,Cloud Computing, and Agile Technology.

Ms.Prajna Bhavi is a MTech Student in Software Engineering in Department of Computer Science & Engineering, East Point College Of Engineering, Visvesvaraya Technological University. She has attended 2 Conference and presented 4 papers in National and International Conferences in various colleges and companies. Her research areas are Software Engineering, Testing, Development. Attended hands on workshop on Design Pattern held by CSI Chapter, Bangalore.

Dr. Chandramouli H is working as Professor,R&D Dept of Computer Science and Engineering at East Point College of Engineering and Technology.His research areas are Wireless Sensors Networks,Mobile Adhooc Networks,IOT and Cloud Computing.He published more than 10 papers in various International Journals.Presently he is guiding for Phd Scholars in Visvesvaraya Technological Univesity (VTU) India.

Dr. B.R Prasad Babu is working as Professor and Head, Dept of Computer Science and Engineering at East Point College of Engineering and Technology. His research areas are Mobile Adhooc Networks,Mobile Communication and Software Engineering. He published more than 50 papers in various International Journals . Presently he is guiding for Phd Scholars in Visvesvaraya Technological Univesity (VTU) and Jawaharlal Nehru Technological University (JNTU) India.