ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

# SECCHECK: A Tool for Software Vulnerability Management in Java Programs

Dr. Chandramouli H, Dr. Josephine Premkumar, Kesavan MV, Abhiman J R,

Prof. and HOD (R&D), Department of CSE, East Point College of Engineering& Technology, Bangalore, India

Prof. and Head, Department of CSE, East Point College of Engineering & Technology, Bangalore, India

Assistant Professor, Department of CSE, East Point College of Engineering & Technology, Bangalore, India

M.Tech Scholar, Department of CSE, East Point College of Engineering & Technology, Bangalore, India

**Project funded by the Government of Karnataka under VGST Scheme (2014-2018)***

**ABSTRACT:**Vulnerability is a weakness in software that enables an attacker to compromise the integrity, availability, and confidentiality of the program and data that it processes.Software applications or programs are implemented in different languages and most of them contain serious vulnerabilities which can be exploited to cause security breaches. Many security vulnerabilities may be present in programs [1] and a number of techniques have been developed to detect these [2]. However, it is essential that these vulnerabilities are not only detected but corrected. Vulnerability managementis the cyclicalpractice of identifying, classifying, remediating, and mitigating vulnerabilities[3]. The paper discusses a tool developed by the authors which not only detects software vulnerabilities but provides solutions for correcting them.The tool also calculates the Degree of Insecurity in a Java program first defined in [2]. It applies the proposed mitigating methods and recalculates the Degree of Insecurity. Experimental results as discussed in the paper indicate that the mitigation methods suggested in the tool are very effective.

**KEYWORDS:** Vulnerability, SecCheck, CWE, Degree of Insecurity,Mitigation.

## I. INTRODUCTION

It is essential that software is engineered so that it continues to function correctly under malicious attacks. Software security is the process of designing, building, and testing software for security: it needs to identify and mitigate weaknesses so that software can withstand attacks. Many recent security breaches have been found to be due to inadequate design and improper coding. The vulnerabilities have unwanted consequences such as hijacking of session information, deletion or alteration of sensitive data, and execution of arbitrary code supplied by hackers[1].It is essential that software vulnerabilities are not only detected but techniques used for correction of such vulnerabilities in order to prevent attacks and minimize operational disruption due to these. WhileVulnerability Management [3][4][5] is being discussed by researchers, there are very few tools which actually does this.

The tool SecCheck developed by the authors can detect a number of vulnerabilities and has been described in [2] while the tool discussed here not only detects vulnerabilities in Java programs but also guides the developers by offering possible solutions for mitigating these. The tool detects vulnerabilities in any Java program caused by Null Point Dereference, Reachable Assertion, Allocation Of Resources Without Limits Or Throttling, Improper Validation of Integrity Check Value, Serializable Class Containing Sensitive Data, cleartext transmission of sensitive information, Improper Validation of Certificate With Host Mismatch, Improper Encoding or Escaping of Output, Improper Neutralization of CRLF Sequences in HTTP Headers, Use of Non-Canonical URL Paths for Authorization Decisions. It also offers solutions for mitigating these.In Section 2 the consequences of the presence of weaknesses are discussed along with ways of mitigating these.Section 3 discusses Degree of Insecurity in a program.Section 4 discusses about the working of the tool while Section 5 describes results of experiments conducted to analyze the effectiveness of the tool.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

## II. COMMON VULNERABILITIES IN SOFTWARE

Common Software Vulnerabilities that occur in Java programs as discussed in CWE [1] are:

1. Reachable Assertion
2. Throttling
3. Null Pointer Dereference
4. Use of a One-Way Hash without a Salt
5. Use of Insufficiently Random Values
6. Missing Support for Integrity Check
7. Improper Neutralization of CRLF Sequences in HTTP Headers
8. Use of Non-Canonical URL Paths for Authorization Decisions
9. Reliance on Cookies without Validation and Integrity Checking in a Security Decision

### 2.1REACHABLE ASSERTION

Reachable assertion occurs when assert is triggered by an attacker causing crash of application or cause a denial of service [6].While assertion is good for catching logicerrors and reducing the chances of reaching more serious vulnerability conditions.

**Pros and Cons:**

1. Chat client allows remote attackers to cause a denial of service (crash) via a long message string when connecting to a server, which causes an assertion failure.
2. Product allows remote attackers to cause a denial of service (crash) via certain queries, which cause an assertion failure.

**Mitigations:**

1. Perform input validation on user data.
2. Make sensitive open/close operation non reachable by directly user-controlled data.

### 2.2 THROTTLING

The software allocates a reusable resource or group of resources on behalf of an actor without imposing any restrictions on how many resources can be allocated.When allocating resources without limits, an attacker could prevent other systems, applications, or processes from accessing the same type of resources [7].

**Pros and Cons:**

1. Driver Large integer value for a length property in an object causes a large amount of memory allocation.
2. CMS does not restrict the number of searches that can occur simultaneously, leading to resource exhaustion.
3. Product allows attackers to cause a denial of service via a large number of directives, each of which opens a separate window.

**Mitigations:**

Limit the amount of resources to users. Set per-user limits for resources (use thread pool concept) allow the system administrator to define these limits.

1. Clearly specify the minimum and maximum expectations for capabilities, and dictate which behaviours are acceptable when resource allocation reaches limits.
2. Ensure that protocols have specific limits of scale placed on them.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

### 2.3 NULL POINTER DEREFERENCE

Null pointer dereferencing occurs when a variable bound to the null value is treated as if it were a valid object reference and used without checking its state. This condition results in a NullPointerException, which can in turn result in a denial of service [8].

**Consequences:**

1. NULL pointer dereferences usually result in the failure of the process unless exception handling (on some platforms) is available and implemented. Even when exception handling is being used, it can still be very difficult to return the software to a safe state of operation.
2. In very rare circumstances and environments, code execution is possible.

**Mitigations:**

1. If all pointers that could have been modified are sanity-checked (handle nullpointer exception) previous to use, nearly all null pointer dereferences can be prevented.
2. Check the results of all functions that return a value and verify that the value is non-null before acting upon it.
3. Explicitly initialize all your variables and other data stores, either during declaration or just before the first usage.

### 2.4 USE OF A ONE-WAY HASH WITHOUT A SALT

The software uses a one-way cryptographic hash against an input that should not be reversible, such as a password, but the software does not also use a salt as part of theinput [9].In cryptography, a **salt** is random data that is used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks versus a list of password hashes and against pre-computed rainbow table attacks.

**Pros and Cons:**

1. If an attacker can gain access to the hashes, then the lack of a salt makes it easier to conduct brute force attacks.
2. While it is good to avoid storing a cleartext password, the program does not provide a salt to the hashing function, thus increasing the chances of an attacker being able to reverse the hash and discover the original password if the database is compromised.

**Mitigations:**

1. Use an adaptive hash function that can be configured to change the amount of computational effort needed to compute the hash, such as the number of iterations ("stretching") or the amount of memory required.
2. Add the salt to the plaintext password before hashing it.

### 2.5USE OF INSUFFICIENTLY RANDOM VALUES

The software may use insufficiently random numbers or values in a security context that depends on unpredictable numbers.

Pseudo-random number generators can produce predictable numbers if the generator is known and the seed can be guessed[10].A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random.

**Pros and Cons:**

1. When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated.
2. And use this guess to impersonate another user or access sensitive information.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

**Mitigations:**

1. Use a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations with adequate length seeds.
2. Use automated static analysis tools that target this type of weakness.
3. Use tools and techniques that require manual (human) analysis, such as penetration testing, threat modelling, and interactive tools that allow the tester to record and modify an active session.

## 2.6MISSING SUPPORT FOR INTEGRITY CHECK

Checksumming is a well known method for performing integrity checks [11].If the computed checksum for the current data input matches the stored value of a previously computed checksum, there is a very high probability the data has not been accidentally altered or corrupted.

**Pros and Cons:**

1. Data that is parsed and used may be corrupted.
2. Without a checksum it is impossible to determine if any changes have been made to the data after it was sent.
3. Attackers can gain access to the sensitive information and can alter the data.

**Mitigations:**

1. Add an appropriately sized checksum to the protocol, ensuring that data received may be simply validated before it is parsed and used.
2. Ensure that the checksums present in the protocol design are properly implemented and added to each message before it is sent.

## 2.7 IMPROPER NEUTRALIZATION OF CRLF SEQUENCES IN HTTP HEADERS

CRLF Injection is a software application coding vulnerability that occurs when an attacker injects a CRLF character sequence where it is not expected.Exploits occur when an attacker is able to inject a CRLF sequence into an HTTP stream [12].CRLF Injection vulnerabilities result from data input that is not neutralized, incorrectly neutralized, or otherwise unsanitized.

**Consequences:**

1. CRLF Injection exploits security vulnerabilities at the application layer.
2. Attackers can modify application data compromising integrity.
3. Enables the exploitation of the following vulnerabilities:
   - XSS or Cross Site Scripting vulnerabilities,
   - Proxy and web server cache poisoning.
4. CR and LF characters in an HTTP header may give attackers control of the remaining headers and body of the response entirely under their control.

**Mitigations:**

1. Add the following import statement for URLEncoder class at top of the file with the rest of the import statements if it does not exist.
   response.sendRedirect(exitPage); to response.sendRedirect(URLEncoder.encode(exitPage, \"UTF-8\"));
2. Use input validation as a defense-in-depth measure to reduce the likelihood of output encoding errors.
3. Fully specify which encodings are required by components that will be communicating with each other.
4. When exchanging data between components, ensure that both components are using the same character encoding.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

### 2.8 USE OF NON-CANONICAL URL PATHS FOR AUTHORIZATION DECISIONS

The software defines policy namespaces and makes authorization decisions based on the assumption that a URL is canonical [13]. This can allow a non-canonical URL to bypass the authorization.Even if an application defines policy namespaces and makes authorization decisions based on the URL, but it does not convert to a canonical URL before making the authorization decision, then it opens the application to attack.

**Pros and Cons:**

1. If a non-canonical URL is used, the server chooses to return the contents of the file, instead of pre-processing the file.
2. An attacker can bypass the authorization mechanism to gain access to the otherwise-protected UR.

**Mitigations:**

1. Add the port number.
   - Remove the fragment (the part after the #).
   - Remove trailing slash.
   - Sort the query string params.
   - Remove some query string params like \"utm_*\" and \"*session*\".
2. Make access control policy based on path information in canonical form. Use very restrictive regular expressions to validate that the path is in the expected form.
3. Reject all alternate path encodings that are not in the expected canonical form.

## III. DEGREE OF INSECURITY IN A PROGRAM

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 as ISM) [2].

$$ISM = \sum_{i=1}^{m} Wi * Ni \text{----------} (1)$$

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

## IV. WORKING FLOW OF THE TOOL

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, itdisplays type of vulnerability found and the place of its occurrence. It also gives the Degree of Insecurity in the input program.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

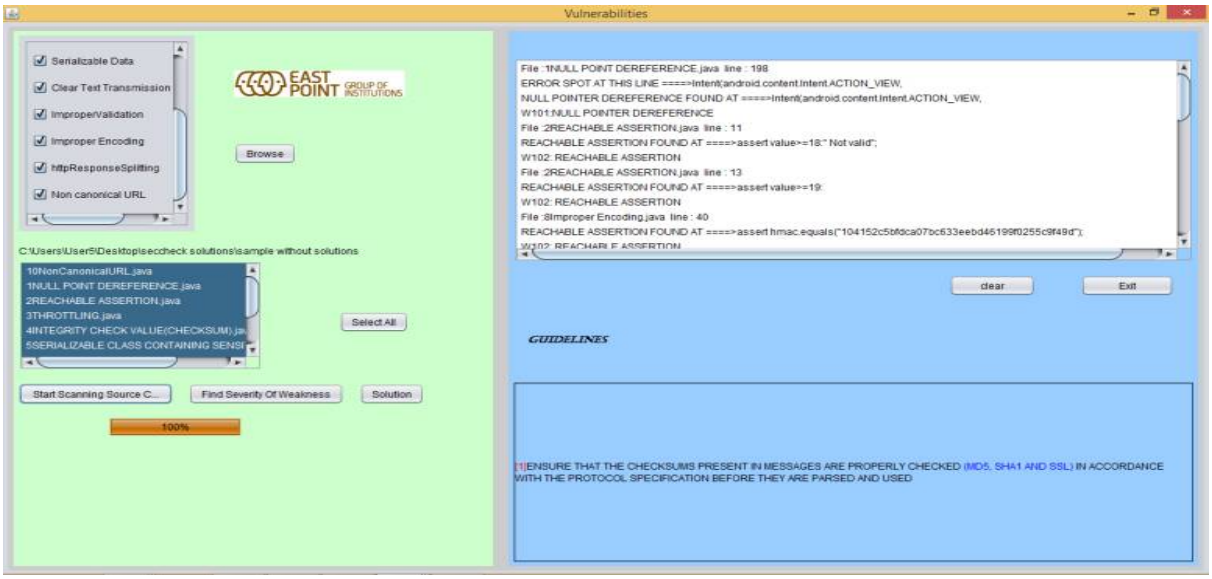


**Figure 1: Structure of Front end of the Tool**

**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.

## V. INETRPRETATION OF RESULTS

For detection of vulnerabilities by the tool we used programs written by the professionals taken from different vulnerability tracking sites and some were taken from Common Weakness Enumeration (CWE) site.

## VI. CONCLUSION

Vulnerability is a weakness in software. The causes of such "weakness" can be faults in design and in code and allows an attacker to reduce a system's information assurance. Thepresence of vulnerabilities in the software makes it necessary to have tools that can help programmers to detect and correct them during development of the code. There are many tools available only to detect the vulnerabilities present in application programs written in various programming languages but no single tool has the capability to detect and mitigate the vulnerabilities found.The tool developed by the authors and described in this paper detects and mitigates twelve vulnerabilities in Java source code.

## REFERENCES

[1]http://www.cwe.mitre.org
[2]Priyadarshini. R, NiveditaGhosh andAnirban Basu "SecCheck: A Tool for Detection of Vulnerabilities and for Measuring Insecurity in Java Programs": International Journal of Software Engineering, Vol.7, No.2, July 2014, pp.67-93
[3] ParkForeman, Vulnerability Management, Auerbach Publications, 2009
[4]http://download.microsoft.com/download/5/0/5/505646ED-5EDF-4E23-8E84 6119E4BF82E0/Mitigating_ Software_ Vulnerabilities.pdf
[5] A Agrawal and R A Khan, A Framework to Detect and Analyze Software Vulnerabilities -Development Phase Perspective", International Journal of Recent Trends in Engineering, Vol 2, No. 2, November 2009, pp 82-84
[6] Reachable Assertion:

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

"http://cwe.mitre.org/data/definitions/617.html"
[7] Allocation of resources without limits or throttling:
"http://cwe.mitre.org/data/definitions/770.html"
[8] Null Pointer Dereference:
"http://cwe.mitre.org/data/definitions/476.html"
[9] use of a one-way hash without a salt:
http://cwe.mitre.org/data/definitions/759.html