AN EFFICIENT APPROACH TO DETECT SUSCEPTIBLE COMPONENTS LOADING

Siva Ramavarapu[1], B Swanth[2], Betam Suresh[3]

[1]Pursuing M.Tech(CSE), [2]Asst. Professor in Department of CSE , [3]HOD

Vikas Group of Institutions, Nunna, Vijayawada. Affiliated to JNTU- Kakinada, A.P, (India)

ABSTRACT

In *dynamic loading of the software components like as libraries or the modules are the widely used for the mechanism to improve the system modularity and the flexibility. The accurate component resolutions are the critical for the reliable and the safely software execution. Hence the programming mistaken may lead to the unintentional or the even malicious components being loaded. In this scenario we consider the dynamic loading which can be hijacked by placing the arbitrary file with the identified name in the directory searched where a real ready target component is resolving. Even though this issue has become popular, that is why everything is known for the quite some time, they had not taken the serious reason which is that exploiting needs to access the local file system on the vulnerable host. Newly the vulnerabilities has been started to achieve the desirable attention as their remote exploitation has become the realistic. This is currently more important to detect and fix the vulnerabilities. This scenario says that, we are presently in static analysis that depends on the automated technique to focus on the vulnerable and the unsafe dynamic component loadings.*

Index Terms: -The unsafe component load and the dynamic analysis.

I INTRODUCTION

The dynamic components are loading the important mechanism for the software development. It gives the permission to the application for the flexibility to the dynamically link the component and used it to exported the functionalities. These benefits are including the modularity and the generic interfaces for the third party software plug in are used such as. The reasons of these are the advantages, and the dynamic loading is more probably used in the designing and the implementing software. In dynamic loading key steps are the components resolution, i.e., locating the correct component for at runtime it will use. The operating systems are gradually provided two the resolution methods, either it will specify the complete path neither the filename of a target component. Total path with the operating systems simply locate target from given complete path. Filename with the operating systems resolving the target by the searching the sequence of the directories and determined by runtime directory search order, to detecting the first occurrence of component. While the possible, this strategy means common component resolution has the inherent security problem. Then only the file names they given, the unintentional or same malicious files with the even file name can be rectified the instead. So long for this issue hasn't been effectively addressed. In the particular way that we show that insecure component loading represents the common class of the security vulnerabilities platforms on Windows and

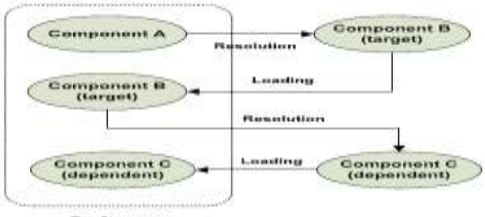
Linux. The operating systems may be provided the mechanisms to protect the system resources. For the Microsoft Windows supports are the Windows Resource Protection (WRP) to the prevented system files from it being replaced. Still those don't prevent the loading of the malicious component situated in the directory searched first directory whenever the intended component are resides. The main problem of the unsafe dynamic loading has been known for a while, but it hasn't been taking the serious threat reason is that its exploitation the required local file system to access on the victim host. Started has been problem to receive many attentions using recently the discovered remote code the execution attacks. Now is the example attack the scenario. The attacker sends are suppose that the victim archive file containing the document for the vulnerable program which means as word press program and the malicious DLL. In the sense is, whether the victims are opens the document then extracting archive file, vulnerable program shall load malicious DLL, sense is leads to the remote code execution. The current break through with the availability of open the source code directories like as source fake peoples are download the sources and the use them as the dynamic components integrated with their code. However trust of that code isn't for configure and it may have been to severe helpfulness. The motivated this is to us for current paper the work. These scenarios are considered as we recommend the static code for the analysis technique to detect the component is safe to the load or unload. The Library users are who intends to use the dynamic component necessary to provide a source code of a component and his/her the source code are which uses dynamic component to the tools. These analyses are the use and the point whether any suspicious vulnerability are present. We will also check for the security violations in the dynamic component code are using CVC rules.

I RELATED WORK

Software components mostly utilize functionalities exported by another components are such as the shared libraries at the runtime. This is the operation generally composed of the three phases: first are resolutions, second are loading, and final are usage. By the way the application resolves the needed target components, loads them, and the utilizes are the desired functions provided by them. Component interoperation could be achieved through the dynamic loading provided by the operating systems or the runtime environment Load library and open systems are calls to use for dynamic loading on the Microsoft windows and UNIX like as operating systems (os), individually. The dynamic loading is generally done in to the two steps: one is component resolution second are chained component loading. By the way that the resolve the target component they are needs to require it is correctly. To the end of the operating systems provide two types of the target component specifications are: full path and filename. In the full path specification and the operating systems solved the target component based on the provided full path.

For example, the full path specification is the /lib/libc-2,7. Hence for the libc library are in the Linux determines targets component are using a specified the full path. For the filename specification and operating systems are obtains the full path of the target components from the provided the file name and the dynamically are determined sequence of the search directories. The particular operating system iterates in through a directories until it is finds a file through the specified file name in which the resolved component. F.e, A target component suppose that is specified as the midimap.dll and directory search orders are given in$C$:\

$Program\ Files\backslash iTunes; C{:}\backslash Windows\backslash System32\ ; . . ; \$PATH$ on Microsoft Windows. If the first directory containing a file with the name midimap.dll is C:\Windows\System32, it resolved the full paths are determined by this is directory. The full path in dynamic loading of a target components are determined by it is specification through a resolution process and the component is the united into the hosted software whether this is not previously loaded. During the process of the incorporating a target components are also loaded time dependent components. Fig. shows the general procedure of the dynamic loading. Suppose the component B is loaded by the component A's and B's dependent components (e.g., component is C) are also be loaded. The information obtain usually we can take on B's dependent components are from the B's file description.



I. Dynamic component loading procedure.

This is a process of the chained components loading are repeated until all the dependent components had loaded.

II THE OVERVIEW OF THE PROPOSED SOLUTION

In this scenario we represent an overview of the solutions are the proposed software tool will analyzed the source code and the find the functionalities have expected from dynamic components. This process is called as the contract construction. The dynamic component code it will check for the full compliance among the contract and the contract violations. These processes are known as the contract violation checked. It is also check whether it is there is any security vulnerability in usage of the dynamic components. We will use a CVC security violation rules are to check for the any security violations are in the code. This is process is known as security violation checked.

2.1 The Details of the Proposed

The Security Mechanism 4.1: In the Contract Construction Contracts are nothing yet the agreement among the user of the dynamic component and a dynamic component. The contracts are in the terms of the classes and the function. Entire the classes and the functions required by a contract should be implemented in the dynamic component. The contract constructions are fully automated process. Once a source code of our application is too provided, it shall be check for dynamic components are usage in the code and extract functions are the class's usage from dynamic components and builds the contracts. The contracts construction are very more language specific and in this scenario we will do the contract construction for the java code and JSR 291 specification for the dynamic components shall be used for the constructing a parser for java code. 4.2 and the contract violation checking Based on the contract extracted from the user code and the dynamic component codes are parsed and checked for the violation. The detected of violence is whether is matching class or matching the function on the class isn't search into the dynamic components.

4.3 Security Vulnerability Detection CWE™: In the international in the scope and the free for the public use, CWE provide the unified, the measurable set of the software weaknesses is that enabling many useful discussion, the description, the selection, and the use of the software security tools are services that can be find that strengths in the source code and the operational systems are as well as the better understanding and the management of the software strengths are related to the architecture and the design. To the assist in the enhancing security are throughout software development in the lifecycle also to the support the desires of the developers, the testers and the educators Common Attack Patterned Enumeration and Classifications(CAPEC) are other of these efforts are sponsored by the DHS CS&C main part of the Software Assurance strategic are initiative. Objective of these efforts are to provide the publicly available catalog of the attack pattern are along with the comprehensive schema and the classifications are the taxonomy. The linked with CWE, CAPEC sites contains are the initial set of the content. The endure will to progress with the public participation and the contributions to form the standard mechanism for the solving, gathering, purifying, and sharing attack patterns are between software community. We can use a following Security violation checking on a code

CWE ID	Description
CVE-2002-0466	The server allows the remote attackers to brows arbitrary directories via a full pathname in the arguments to the certain dynamic pages.
CVE-2002-1483	The Remote attacker can read the arbitrary file
CWE-2001-400	Resource exhaust
CVE-2001-0255	Suspicious read write
CVE-1999-0674	System call usages

III EVALUATION

In this scenario we can the evaluated the unsafe components are loadings on the Microsoft Windows and the Linux. We detect unsafe component loadings for every platform in the diverse selection of the applications is popular. In this we structure our analysis are of the detection results to the answer following research questions are: RQ1: How are prevalent and the severe the unsafe components loadings on the Microsoft Windows. RQ2: How are prevalent and the severe are the unsafe component loadings on to the Linux. RQ3: What's the implication of our finding? RQ4: How's does our detection technique associate to the related work.

IV EXISTING SYSTEM

In the existing system to evaluate our technique, we has been implemented as the set of are tools for detecting the unsafe components are loadings on Microsoft Windows family like as XP SP3, Vista SP2, and the Windows 7 and another OS are Ubuntu 10.04 is the more popular Linux distribution. An extensive analysis of a prevalence we conducted and severity of the unsafe components are loadings in the popular software applications are: 27 for the Windows and 24 for the Linux OS. Results of our shows that unsafe components are loading prevalent on entire analyzed platforms are in this we found exactly 3,269 the unsafe loadings on the Windows and 752 on the Ubuntu.

We represent as the effective dynamic analysis to detect vulnerable and the unsafe dynamic components are loadings. This is work introduces the first automated technique to detect and the analyze vulnerabilities and the errors related to dynamic components are loading to our knowledge. The problems are of the unsafe dynamic loading has been known for the while, way is that it haven't been considering the very serious threats are because these exploitation requires the local files system access on victims host. Started problem have to receive much attention due to newly discovered the remote codes

V PROPOSE SYSTEM

In the propose system future works are interested in the developing static binary analysis techniques to the detection of unsafe components are loadings. While our dynamic analyses are the effective, it may suffer from standard limitation of the dynamic analysis, namely as the code coverage problems. Exactly, our approach may be missed the unsafe components are loadings that can happened. We develop to plan sound, practical static analysis techniques are to the complement the dynamic analysis we introduced here. These are analyzes the profiles to detect two types of the unsafe components loadings are: the resolution failures are unsafe resolutions, to the evaluated our techniques, we can implement the tools are to detect unsafe components loadings on the Microsoft Windows and the Linux in this the evaluation shows that the unsafe components are loadings to prevalent on both platforms and much severe on the Windows platforms from the security perspective. In particular way is that, our tools are detected much than the 4,000 unsafe component loadings in the popular software on the total platforms. It is also discovered the 41 potential remote code execution attack on the Microsoft Windows.

VI RESULTS

The result of this implemented proposed solutions are in the JAVA. In this we tested out the solution for the using the 5 libraries are on image processing downloaded from the source forge and entire these used as the dynamic components in the image applications and we measured number of the contracted violations. The number of the security violations we can also found. In this we found our solutions are not unable to find out more than 75% of the compliance faults and the security vulnerabilities

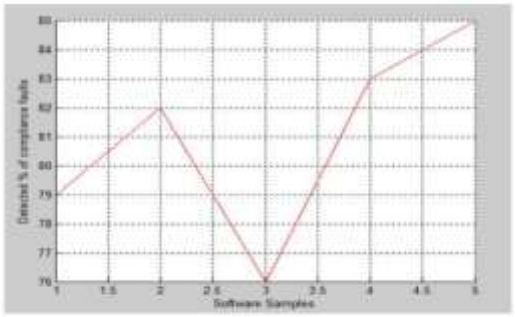


Fig. 2: Number of Faults occurred.

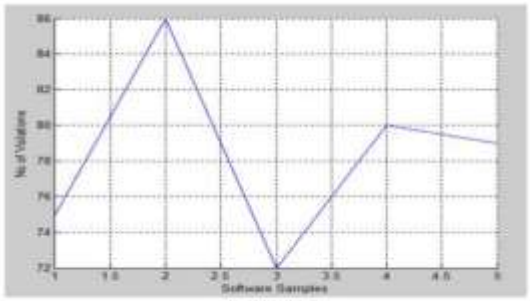


Fig. 3: Number of violations in software.

VII CONCLUSION

In this scenario we have proposed and the evaluated static analysis solutions to detect the unsafe component loadings and solve that our solutions are not unable to identify much than 75% vulnerabilities. We propose the static code analyses technique to detect the component is safe and to the load or to unload. This technique involves analyzing source code and they point out whether any vulnerability is present.

REFERENCE

[1] "About Windows Resource Protection," http://msdn.microsoft. com/en-us/library/aa382503(VS.85).aspx, 2011.

[2] "Windows DLL Exploits Boom; Hackers Post Attacks for 40-Plus Apps," http://www.computerworld.com/s/article/9181918/ Windows_DLL_exploits_boom_hackers_post_attacks_for_40_ plus_apps, 2011.

[3] "Hacking Toolkit Publishes DLL Hijacking Exploit," http://www.computerworld.com/s/article/9181513/Hacking_toolkit_ publishes_DLL_hijacking_exploit, 2011.

[4] T. Kwon and Z. Su, "Automatic Detection of Unsafe Component Loadings," Proc. 19th Int'l Symp. Software Testing and Analysis, pp. 107-118, 2010.

[5] "Zero-Day Windows Bug Problem Worse than First Thought, Says Expert," http://www.computerworld.com/s/article/9180978/ Zero_day_Windows_bug_problem_worse_than_first_thought_ says_expert, 2011.

[6] "Update: 40 Windows Apps Contain Critical Bug, Says Researcher," http://www.computerworld.com/s/article/9180901/ Update_40_Windows_apps_contain_critical_bug_says_ researcher, 2011.

[7] "Researcher Told Microsoft of Windows Apps Zero-Day Bugs 6 Months Ago," http://www.computerworld.com/s/article/ print/9181358/Researcher_told_Microsoft_of_Windows_apps_ zero_ day_bugs_6_months_ago, 2011.

[8] "Exploiting DLL Hijacking Flaws," http://blog.metasploit.com/ 2010/08/exploiting-dll-hijacking-flaws.html, 2011.

AUTHORS PROFILE

Siva Ramavarapu, pursuing M.Tech(CSE) from Vikas Group of Institutions, Nunna, Vijayawada. Affiliated to JNTU-Kakinada, A.P., India

B Swanth, working as a Asst. Professor of CSE department at Vikas Group of Institutions, Nunna, Vijayawada, Affiliated to JNTU-Kakinada, A.P., India

Betam Suresh, is working as an HOD, Department of Computer science Engineering at Vikas Group of Institutions, Nunna, Vijayawada, Affiliated to JNTU-Kakinada, A.P., India