Detecting Web Application Vulnerability using Dynamic Analysis with Penetration Testing

Dr. T.Pandikumar [1] Tseday Eshetu [2]

[11] *Phd, Department of Computer and Information Technology, Defence University, College of Engineering, Debre Zeyit, Ethiopia*

[2] *M-tech, Department of Computer and Information Technology, Defence University, College of Engineering, Debre Zeyit, Ethiopia*

---***---

Abstract - *Web application is becoming so popular and significant part of our daily lives. Due to the use of web applications increasing day by day, the web application security is becoming vital for user's secret data. In parallel to this, the number of reported web application vulnerabilities is increasing dramatically. Most of the vulnerabilities are the result of improper input validation. This paper discuss the Tainted Mode Model (TMM) which allows inter module vulnerabilities detection. Besides, the seminar presents a new approach to vulnerability analysis which incorporates advantages of penetration testing and dynamic analysis. This approach effectively utilizes the extended Tainted Mode Model.*

Key Words: *Web Applications, Vulnerability Analysis, Penetration Testing, Dynamic Analysis, Taint Analysis*

1. INTRODUCTION

Web application is a software program that runs on a web server, it became more and more popular and important part of our daily life than before. Web application contains security vulnerability, attackers have an over growing list of vulnerability to exploit in order to maliciously gain access to web application. Thus the task of securing web applications is one of the most urgent for now.

According to, the most efficient way of finding security vulnerabilities in web applications is manual code review. This technique is very time-consuming, requires expert skills, and is prone to overlooked errors. Therefore security society actively develops automated approach to finding security vulnerability. This approach can be divided into two wide categories Black box and White box testing. The first approach is based on web application analysis from the user side, assuming that source code of an application is not available. The idea is to submit various malicious patterns (implementing for example SQL injection or cross-site scripting attacks) into web application forms and to analyze its output thereafter. If any application errors are observed an assumption of possible vulnerability is made. This approach does not guarantee neither accuracy nor completeness of the obtained results. The second approach is based on web application analysis from the server side, with assumption that source code of the application is available. In this case dynamic or static analysis techniques can be applied. A comprehensive survey of these techniques was made by vigna et al., according to this survey several statements could be made. [1]

- ✓ The most common model of input validation vulnerabilities is the Tainted Mode model. This model was implemented both by means of static or dynamic analysis.
- ✓ Another approach to model input validation vulnerabilities is to model syntactic structure for sensitive operations arguments. The idea behind this is that the web application is susceptible to an injection attack, if syntactic structure for sensitive operation arguments depends on the user input. This approach was implemented by means of string analysis in static and it was applied to detect SQLI and XSS vulnerabilities in PHP.
- ✓ One of the main drawbacks of static analysis in general is its susceptibility to false positives caused by inevitable analysis imprecisions. This is made worse by dynamic nature of scripting languages. However, static analysis techniques normally perform conservative analysis that considers every possible control path.

1.1 TAINTED MODE MODEL

Taint mode is a way of making your code more secure. It means that your program will be fussier about data it receives from an external source. External sources include users, the file system, the environment, locale information, other programs and some system calls.

Dynamic and static analysis uses Tainted Mode model for finding security vulnerabilities that cause improper input validation.

According to, following assumptions were made within Tainted Mode Model:
1. All data received from the client via HTTP-requests is untrustworthy (or tainted).
2. All data being local to the web application is trustworthy (or untainted).
3. Any untrustworthy data can be made trustworthy by special kinds of processing, named sanitization with these assumptions made; security vulnerability is defined as a violation of any of the following rules:
1. Untrustworthy (tainted) data should not be used in construction of HTTP responses.
This prevents cross site scripting attacks.
2. Untrustworthy (tainted) data should not be saved to local storages. This prevents possible construction of HTTP responses from these sources in future.
3. Untrustworthy (tainted) data should not be used in system calls and in construction of commands to external services such as database, mail, etc. This prevents most of injection attacks.
4. Untrustworthy (tainted) data should not be used in construction of commands that would be passed as input to interpreter. This prevents script code injection attacks. [2]

2. Dynamic Analysis Testing

A Dynamic analysis test communicates with a web application through the web front-end in order to identify potential security vulnerabilities and architectural weaknesses in the web application. Unlike source code scanners, a dynamic analysis program doesn't have access to the source code and therefore detects vulnerabilities by actually performing attacks.[2]
A dynamic analysis security scanner can facilitate the automated detection of security vulnerabilities within a web application. A dynamic analysis test is often required to comply with various regulatory requirements. Dynamic analysis scanners can look for a wide variety of vulnerabilities, including:
 Input/output validation: (Cross-Site Scripting, SQL Injection, etc.)

 Specific application problems

 Server configuration mistakes/errors

3. PENTETERATION TESTING

A penetration test, occasionally called as pen-test, is a method of evaluating the security of a computer system or network by simulating an attack from malicious outsiders (who do not have an authorized means of accessing the organization's systems) and malicious insiders (who have some level of authorized access). The process involves an active analysis of the system for any potential vulnerabilities that could result from poor or improper system configuration, either known and unknown hardware or software flaws, or operational weaknesses in process or technical countermeasures. This analysis is carried out from the position of a potential attacker and can involve active exploitation of security vulnerabilities.
Penetration testing approach is based on simulation of attacks against web applications. Currently, penetration testing is implemented as black box testing. A vulnerability assessment simply identifies and reports noted vulnerabilities, whereas a penetration test attempts to exploit the vulnerabilities to determine whether unauthorized access or other malicious activity is possible. Penetration testing typically includes application security testing as well as controls and processes around the applications.

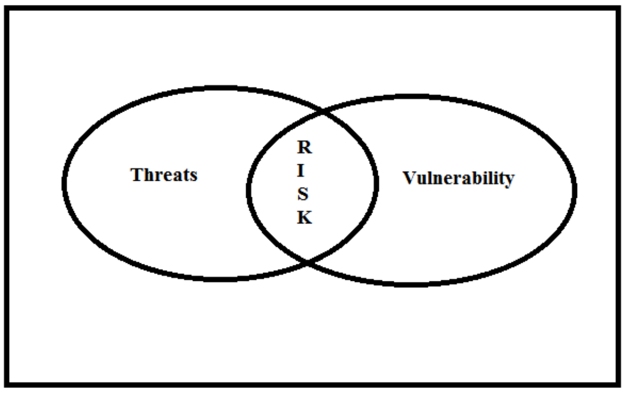

Figure 1: Visualization of vulnerability in the application [2]

3.1 Conducting Penetration Testing

Penetration testing is not merely the serial execution of automated tools and generation of technical reports as it is frequently viewed. It should provide a clear and concise direction on how to secure an organization's information and information systems from real world attacks.
One critical factor in the success of penetration testing is its underlying methodology. A systematic and scientific approach should be used to successfully document a test and create reports that are aimed at different levels of management within an organization. It should not be restrictive to enable the tester to fully explore his intuitions.
Generally, penetration testing has three phases: test preparation, test, and test analysis as shown in Figure 2.

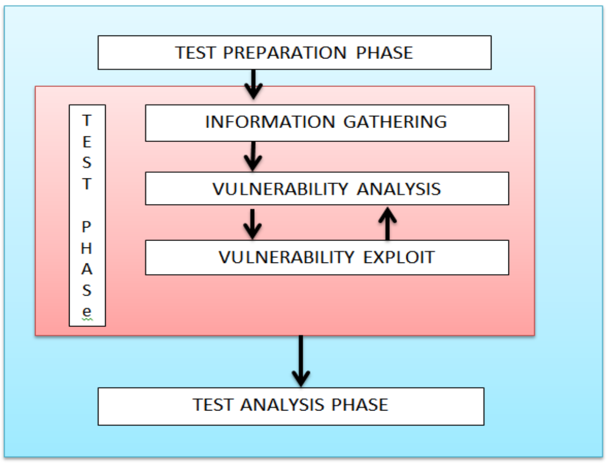

Figure 2: Penetration Testing Methodology [6]

All the necessary documents for the test are organized and finalized during the test preparation phase. The testers and the organization meet to decide the scope, objectives, timing, and duration of the test. Issues such as information leakages and downtime are resolved and put into legal agreement document. Other legal agreements that are deemed necessary are concluded and signed during this phase. The bulk of the penetration testing process is done during the test 12
phase. This phase involves the following steps: information gathering, vulnerability analysis, and vulnerability exploits.

3.2 Penetration Test Process

The Penetrator performs information discovery via a wide range of techniques that is, databases, scan utilities and more in order to gain as much information about the target system as possible. These discoveries often reveal sensitive information that can be used to perform specific attacks on a given machine [2]. Development process in such a way that findings can help improve design, implementation, and deployment practices

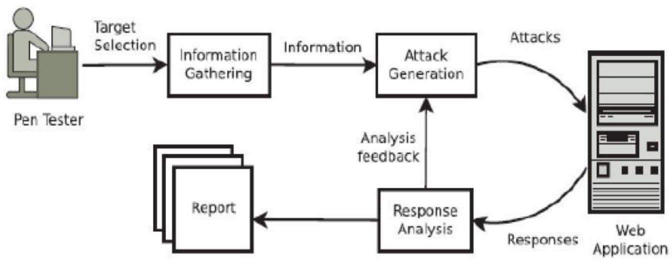

Figure 3: Penetration Testing Process [2]

4. CAUSES OF VULNERABILITIES

All vulnerabilities identified in Web applications, problems caused by unchecked input which are recognized as being the most common. To exploit unchecked input, an attacker needs to achieve two goals:

- **Inject malicious data into Web applications.**
  Common methods used include:
  - URL manipulation: use specially crafted parameters to be submitted to the Web application as part of the URL.
  - Hidden field manipulation: set hidden fields of HTML forms in Web pages to malicious values.
  - HTTP header tampering: manipulate parts of HTTP requests sent to the application.
  - Cookie poisoning: place malicious data in cookies, small files sent to Web-based applications.
- **Manipulate applications using malicious data**
  Common methods used include:
  - SQL injection: pass input containing SQL commands to a database server for execution.
  - Cross-site scripting: exploit applications that output unchecked input verbatim to trick the user into executing malicious scripts.
  - HTTP response splitting: exploit applications that output input verbatim to perform Web page defacements or Web cache poisoning attacks.
  - Path traversal: exploit unchecked user input to control which files are accessed on the server

5. CONCLUSIONS

This paper presents a new approach to automatic penetration testing by leveraging it with knowledge from dynamic analysis. Most of vulnerabilities result from improper or none input validation by the web application. Most existing approaches are based on the Tainted Mode vulnerability model which cannot handle inter-module vulnerabilities. Vulnerability management should be considered a priority given the sophisticated malware targeting client PCs inside the organization.

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. T. PANDI KUMAR for the suggestions, ideas, and advices during the development of this work. Especially, I thank him for his consistent guidance and encouragement throughout my work. The achievement of this paper work would not have been possible without his help.

REFERENCES

[1] T.Sofia and R.Kannan ―detecting security vulnerabilities in web applications using dynamic analysis with penetration testing‖ International Journal of Innovative Science and Applied Engineering Research, Volume 13, Issue 40 Ver. II (Jan. 2015)
[2] Sreenivasa Rao and Kumar N "web application vulnerability detection using dynamic analysis with

penetration testing" International Journal of Enterprise Computing and Business Systems, Vol. 2 Issue 1 January 2012
[3] Deven Gol, Nisha Shah and Priyank Bhojak —Web Application security tool to identify the different Vulnerabilities using RUP model|| International Journal of Emerging Trends in Electrical and Electronics Vol. xx Issue. xx, May. 2015
[4] Mahin Mirjalili, Alireza Nowroozi, and Mitra Alidoosti —A survey on web penetration test|| ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 6, No.12 , November 2014 ISSN : 2322-5157
[5] G. Bacudio, Xiaohong Yuan, and Bei-Tseng Bill Chu —an overview of penetration testing|| International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.6, November 2011
[6] Farkhod Alisherov A., and Feruza Sattarova Y. —Methodology for Penetration Testing|| International Journal of Grid and Distributed Computing Vol.2, No.2, June 2009
[7] Kozlov D, and Petukhov A. —Implementation of Tainted Mode approach to finding security vulnerabilities for Python|| International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.7, November 2013
[8] J. Bau, D. Gupta, and J. C. Mitchell. —Automate Black-Box Web Application Vulnerability Testing||. In Proceedings of the IEEE Symposium on Security and Privacy, 2012.

BIOGRAPHIES

My name is Tseday Eshetu Belayneh

I received my BSc. Degree in Information Communication Technology (ICT) in 2008. Currently am taking my M –tech in Computer and information technology from Defense University College of Engineering. In 2009, I joined Rift Valley University to work as An Instructor and I am still working there in this position.