Development of Exploitation Framework for Vulnerability Assessment of Enterprise LAN

Synopsis of the Thesis to be submitted in Partial Fulfillment of the Requirements for the Award of the Degree of

Master of Technology

In

Computer Science and Information Technology

By

Major Santosh Prasad

Under the supervision of

Dr. S.K. Ghosh

School of Information Technology

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

India

April 2008
Contents

1. Introduction.
2. Literature survey.
4. Objective and Scope.
5. Work done.
7. Summary

References
1. Introduction

The explosive growth of the Internet has brought many good things: electronic commerce, easy access to a vast store house of reference material, collaborative computing, e-mail, and new avenues for advertising and information distribution, to name a few. However, as with most technical advances, there is also a dark side: we have SPAM, cyber crimes and invasion of privacy by criminal hackers. To make things worse, while the system user needs to patch all possible vulnerabilities in his/her system, a hacker only needs to locate any one to break in.

Formal verification approach can only provide validation of software against vulnerabilities at abstract level. With increase in number of systems over the network, this approach becomes almost ineffective.

As the number of system vulnerabilities multiply in recent years, Vulnerability Assessment (VA) tools that can identify vulnerabilities in existing systems before actual exploitation takes place have become immensely important.

Vulnerability assessment is a method of evaluating the security of a computer system or the Enterprise network. The process involves an active analysis of the system for any weaknesses or technical flaws in the OS or the applications which are running on them. These are known as vulnerabilities. The vulnerability assessment cycle as shown in figure 1.1 comprise of the following stages:

![Figure 1.1](image-url)
• **Information gathering** – Background information is gathered from the target, whether the target is a process or a system. An example of this is obtaining public information from the Internet about the target organization.

• **Reconnaissance** – Positive confirmation of the target. Contact is made with the organisation to confirm that targets and security controls are as expected. An example in a physical test would be to visit a target site as a guest or bystander. In a network test, this might be sending traffic to confirm the existence of routers, web servers and email servers.

• **Enumeration** – Establishing the potential points of access being offered by a target. In a network test this will involve scanning for open services on targets or establishing the existence of possible user identification credentials.

• **Vulnerability identification** – Identifying potential vulnerabilities in a target. In a network test this will consist of using tools to test for vulnerabilities on a particular product, for example a router. In a web application test, this may involve finding an input field that does not check for malicious code in the text being entered.

• **Exploitation** – Using identified vulnerabilities to gain unauthorised access to the target. For example, in a web application test, this may involve injecting commands into the application that provide a level of control over the target.

• **Escalation** – Gaining further access on a target, once an initial level of access has been obtained. For example, in a network test, successful exploitation may allow user or guest access to a system. Escalation through additional exploitation will be required to obtain administrative privilege.

• **Advancement** – Attempting to move on from the compromised target to find other vulnerable systems. For example, in a network test this will consist of “hopping” from one system to another, potentially using the access obtained on the original target to access other systems. In a physical test, this might involve moving from one compromised building to another.

VA Tool or Penetration Testing intends to check our own network and computer systems against vulnerabilities in order to proactively find the security holes and patch them
before any attacker try to exploit them. **Exploitation Framework** is the most important part of vulnerability assessment tool, which launches attack on the vulnerable machine to gain unauthorized access.

An **exploit** is a piece of software, a chunk of data, or sequence of commands that take advantage of a bug, glitch or vulnerability in order to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerized). This frequently includes such things as gaining control of a computer system, allowing privilege escalation or a denial of service attack.

### 2. Literature Survey

In this section, we present a brief survey of literature on the topics of interest of the thesis. The scope of the survey is divided into bringing out the motivation of the thesis work: automating the entire vulnerability assessment and exploitation cycle, exploitation framework on Ruby-on-Rails, writing exploits and DoS attacks, and creating shellcodes.

**Automating the entire vulnerability assessment and exploitation cycle:** In [1], Chang Liu and Debra J. Richardson talks about ‘Automated Security Checking and Patching Using TestTalk’. They describe how manual methods are time consuming and impractical for scanning and patching vulnerable systems in the network. To solve this problem, they propose to develop a framework for automated security checking and patching. Securibot downloads security updates in machine readable format. This tool is divided in two main parts, the scanners and the patchers. The scanners portion is responsible to scan the system for known vulnerabilities and then the patchers patches them automatically. It can also detect compromised systems using attack signatures.

Another important work by Michitaka Yoshimoto, Bhed Bahadur Bista and Toyoo Takata on ‘Development of Security Scanner with High Portability and Usability’ [2], talks about the need for automated Vulnerability scanners which have high usability. As per the authors a scanner which has high usability should have the following characteristics:

1. Easy installation
(2) Easy operation
(3) Intelligible scanned result.
The scanner proposed in the paper is also on similar lines and have tried to ensure high usability and easy installable. To ensure these features the tool developed is a Web based security scanner with an impressive GUI and has high user-friendliness which makes it comfortable to be used by even a layman.
Juanita Koilpillai, John B. Beavers and Paul Swinton in their paper titled ‘Recon - A Tool for Incident Detection, Tracking and Response’ [3] suggests a general framework designed to integrate many smaller available services that cooperate with each other to detect and react to security incidents in order to minimize or eliminate harm to the enterprise.
There are other works that concentrate on customised tools like the Self Port Scanning Tool by Joshua E. Koche and Dr. David P. Gilliam [4] and the NetGlean scanner by Gavin W. Manes et. al. [5]. All such tools have tried and optimized a particular phase of the vulnerability assessment cycle.

**Exploitation framework:** The framework has been developed in Ruby-on-Rails environment. For this purpose learning Ruby scripting language was essential and following books and references helped a lot in doing the same:
- Ruby-developers-guide by Robert Feldt and Lile Johnson [6].
- Ruby on Rails project [7].
- Ruby on the web [8].

**Developing Exploits and DoS attacks:** To write a specific exploit or DoS, we need to first understand the vulnerability type or the flaw present in the code. Because, our area of interest was in buffer overflow attacks, we read some literatures pertaining to buffer overflow. A book named Buffer overflow attacks by James C Foster [9] gives an in-depth understanding of buffer overflow bug and its exploitation. Buffer overflow demystified by murat@enderunix.org [10] explains the buffer overflow vulnerability in detail including various examples.
Creating Shellcode: The creation of shellcode was important, firstly to understand the working of it and then to write our own shellcode to accomplish some specific task. Sockets, Shellcode, Porting, and Coding, a book written by James C Foster [11] provides an in depth knowledge of understanding shellcodes and its creation. Understanding Windows Shellcode by M Miller [12] provides the art of writing a Windows specific shellcode.

3. Motivation

Although, a wide variety of commercial and open-source vulnerability detection and security assessment tools are available, their level of automation and process management varies considerably. Most of them require a certain amount of experience and background knowledge before they can be effectively used. Though all these tools perform one or more functionality of the VA cycle, but a fully automated tool that integrates all aspects satisfactorily is hard to find.

Thus, the primary problem is: how to transform the traditionally labor-intensive vulnerability assessment process into a scalable vulnerability management system.

The few automated tools available in the network security arena are mainly commercial products. Apart from cost being the major factor against their unabated use, they are also prone to be laced with some backdoors which may compromise the organisations interests in more than one ways. This has very serious implications in certain organisations like Defence, financial organisations and governmental bodies. These organisations deal with sensitive data and often have stringent requirements. The commercial tools being proprietary in nature are not customizable to the organization’s needs. They are pre packaged products being sold with taller claims than what they can actually fulfill. On the other hand, the open source tools are not fully automated. They require a lot of effort on the part of security personnel who has to integrate the piece meal information obtained from each freeware tool and manually leverage the goal of penetration testing.
4 Objective and scope

The objective of this project is to develop an Exploitation framework for vulnerability assessment of enterprise LAN which is the most important module of the vulnerability assessment tool. The scope of our project can be sub-divided into following heads:

- Development of exploitation framework.
- Coding exploits and DoS attacks in Ruby.
- Creating custom made payloads for bind shell and advancement.

5 Work done

Development of exploitation framework: An exploitation framework was required to be developed for automating the entire process of vulnerability assessment cycle. The framework requires: i) a database management system for storing the scan information and exploits result, ii) a knowledge base for storing various decision making information, iii) a front-end as user interface for automatically launching the exploits and, iv) a web server for hosting the toolkit. The model architecture of the VA tool is as shown in Figure 5.1.

![Figure 5.1](image-url)
For the purpose of developing exploitation framework following work has been done:  

i) MySQL has been used as a backend database and knowledge base. Various tables have been created for storing various kind of information as required by the framework, 

ii) Ruby language has been used for creating web pages for the front end,  

iii) Rails web application framework is used to build the entire architecture framework, and 

iv) WEBrick server is used for hosting the toolkit. The operating system used is Fedora Core 6.

**Coding Exploits and DoS attacks:** Following exploits and DoS attack have been written in Ruby and integrated with exploitation framework:  

i) DCOM_RPC remote buffer overflow exploit for Windows operating system,  

ii) IGMP_V3 denial of service (DoS) attack for Windows operating system,  

iii) Exploit code for advancement in which attempt is made to move on from one compromised machine to other vulnerable systems. This consists of “hopping” from one system to another using the access obtained on the original target machine. For this work, specific shellcode has been written which launches attack in two stages to accomplish this job.

**Creating custom made payloads:** Writing shellcode involves an in-depth understanding of assembly language for the target architecture in question. Shellcode is therefore often created to target one specific combination of processor, operating system and service pack, called a platform. Within shellcode, system calls or library functions are used to perform actions, for example creating a socket or spawning a shell. Therefore, shellcodes are OS-dependent because different operating systems use different system calls.

For the purpose of developing this VA tool, we have written our own shellcodes. The shellcode is specific to Windows XP, SP-1 platform. Both the shellcodes are remote shellcodes, i.e., uses standard TCP/IP socket connections.

- The forward bind shellcode is a normal bind shell which opens a port at the target machine for forward connection.
- The download and execute shellcode is written for staged attack i.e. for advancement over the network. During first stage a special payload is sent from
host machine which opens a port at the target machine and waits for an exe file to be received. During second stage the exe file containg the exploit code for attacking the third machine along with its payload is sent to the target machine. This exe is only written over the machine’s RAM and not saved on the hard disk, thereby avoiding any detection or tarce. This exe is then executed at the target machine, which launches attack on third machine and forwards the command prompt received from it to the host machine. There is no direct network connection between the host machine and the target machine. Therefore, the identity of the host machine can be protected.

6. Organisation of the thesis

Chapter 1. Introduction: This chapter contains an introduction to the vulnerability assessment, exploitation and various phases of vulnerability assessment cycle. It also contains a brief background, literature survey and, motivation and objective.

Chapter 2. Vulnerability Assessment and System Exploitation: This chapter explains the methodology and techniques behind vulnerability assessment and system exploitation. The entire penetration test cycle has also been explained.

Chapter 3. Framework and Tool Architecture: In this chapter, we explain the architectural model of the tool, explain the Model-View-Controller (MVC) architecture on Ruby-on-Rails, provide the framework methodology and, usage model of the tool.

Chapter 4. Developing Exploits: This chapter provides detail knowledge on developing exploits and wriing DoS attacks. Buffer overflow attack has also been explained in depth. The exploit and DoS attack codes which are written in Ruby have also been explained in detail.
Chapter 5. **Shellcoding and Advancement:** This chapter gives an overview of shellcoding and then describes the art of writing custom made shellcodes. Shellcoding in this chapter is specific to Windows operating system. We also describe our own payloads which have been written for bindshell and advancement.

Chapter 6. **Framework Implementation and Results:** In this chapter, we explain the working of our Exploitation Framework tool which is followed by the results.

Chapter 7. **Conclusion:** Finally we conclude our thesis by mentioning major contribution made to accomplish the objective. This is followed by some future scope to our exploitation framework.

7. **Summary**

Developing an integrated and automated vulnerability assessment tool for an enterprise LAN is very essential especially for **Defence, financial organisations and governmental bodies**. The few automated tools available in the network security arena are mainly commercial products. Apart from cost being the major factor against their unabated use, they are also prone to be laced with some backdoors which may compromise the organisations interests in more than one ways. The open source tools are not fully automated and they require a lot of effort on the part of security personnel to accomplish the entire task they need to carry out. Therefore, there was a need felt to develop an open source, web-based, user-friendly tool, which can integrate and automate the entire vulnerability assessment cycle.

References


[2] Development of Security Scanner with High Portability and Usability by Michitaka Yoshimoto Bhed Bahadur Bista Toyoo Takata Faculty of Software and Information
Science, Iwate Prefectural University Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA’05) 2005 IEEE.


