International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

A Review Paper on Linux Mitigation & Countermeasures Against It's Vulnerabilities

Archit Rana[1], Chandresh Parekh[2]
[1]M.Tech, Raksha Shakti University, Ahmedabad, Gujarat, India
[2]Assistant Professor, Raksha Shakti University, Ahmedabad, Gujarat, India

ABSTRACT
This paper is intended as finding some Linux vulnerabilities and it's countermeasures for securing Linux operating systems. Here proposed Linux distribution is Red Hat Enterprise Linux . observing Linux system there are some vulnerabilities like unauthorized access done by booting and overwriting root password, granted to local user as super user, network level vulnerabilities, gaining rights on file permissions etc. which compromise the database confidentiality and security. By using shell scripting, limiting or restricting access to systems, network monitoring Linux tools, implementing password policies and Security enhanced Linux policies to overcome these kind of problems in Linux systems at server side or client side to make sure operating system security.

Keywords: Red Hat Enterprise Linux, Shell Scripting, password policies, SE Linux.

I. INTRODUCTION

Linux is a popular version of the UNIX Operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

Kernel: Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It is consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

System Library: System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.

Run level in Linux Operating System: Run level is a state or mode of operating system in Linux. There are "seven" Run levels which is represented by a single digit integer. It is one of the most important Terms in Linux. Linux kernel supports and differentiates all levels.

Seven Run levels are as follows:

Run level 0: It means System in haul state, there is no activity.

Run level 1: It means system is in "single user" mode. Which are rarely used?

Run level 2: It means system is in multiple users' mode but no network file system (NFS) are there. Which also rarely used.

Run level 3: It means system is in multiple users, Command line mode. It is a default run level in most of the Linux system.

Run level 4: It means system is in User-definable mode. That is not in used.

Run level 5: It means system is in multiple users' mode with GUI (graphical user interface). This one is the standard run level.

Run level 6: It is used when the user wants to restart the system basically for the Reboot sake

.

Linux Boot-up Sequence: The Linux boot-up sequence basically follows six steps to complete the boot-up sequence. Each step is responsible for (depends on) the next.

Master boot record (MBR): It is located in the first sector of the boot disk. It takes around.Some boot loaders are GRUB, LILO or Linux Loader.

Basic Input Output System (BIOS): Searches and executes the boot loader program from the MBR.It is like an abstraction layer between the hardware and the software.

II. BACKGROUND

Linux basic commands

1) **cat:** a Unix/Linux command that can read, modify or concatenate text files. Cat commands are most commonly used for displaying the contents of file.
2) **cd:** the cd command changes the current directory in Linux and can toggle between directories conveniently.
3) **chmod:** chmod changes the access mode (permissions) of one or more files. Only the owner of a file or a privileged user may change the access mode.
4) **chown:** chown changes file or group ownership and has the option to change ownership of all objects within a directory tree, as well as having the ability to view information on objects processed.
5) **cp:** the cp command copies files and directories; copies can be made simultaneous to another directory if the copy is under a different name
6) **date:** date sets a system's date and time. This is also a useful way to output/print current information when working in a script file.
7) **echo:** echo allows a user to repeat, or "echo," a string variable to standard output
8) **exit:** the exit command terminates a script and can return a value to the parent script.
9) **find:** find searches the directory tree to find particular groups of files that meet specified conditions, including --name and --type, -exec and --size and --mtime and --user.
10) **grep:** grep searches files for a given character string or pattern and can replace the string with another. This is one method of searching for files within Linux.
11) **gzip:** gzip is the GNU project's open source program used for file compression, compressing web pages on the server end for decompression in the browser. This is popular for streaming media compression and can concatenate and compress several streams simultaneously.
12) **ifconfig:** ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces. After that, it is usually only needed when debugging or when system tuning is needed.
13) **locate:** locate reads one or more databases and writes file names matching certain patterns to output.
14) **ls:** the ls command lists files and directories within the current working directory, allowing admins to see when configuration files were last edited.
15) **man:** short for "manual," man allows a user to format and display the user manual built into Linux distributions which documents commands and other aspects of the system [1], [2], [7].

Basic Architecture of SELinux

NSA tried to get their SELinux patches. development branch kernel back in 2001, but Linus Torvalds rejected the proposal since there were other similar ongoing projects at the same time. A more general solution was needed so that the kernel would be able to support as many security architectures and implementations as possible, with-out sticking too much to the ideas of any specific implementation.[3]

3 Flask architecture and concepts

The very flexible MAC architecture used in SELinux is Flask, which was derived from a micro-kernel based operating system named Fluke. Flask clearly separates the security policy from the enforcement mechanism.

All subjects (processes) and objects (files, sockets, etc.) have a set of security attributes, referred to as the security context of the object. The attributes depend on the specific security policy in use, but generally contain the user id, a role and type enforcement domain.

Instead of working with the security context all the time, the security server maintains a mapping between security at-tribute sets and security identifiers (SIDs). When the object manager, the enforcing component, request labeling or ac-cess decisions, it typically passes a pair of SIDs to the secu-rity server, which looks up the related security contexts and makes the decision based on the policy in use.

Polyinstantiation is used when a certain resource needs to be shared by many clients. Such a resource could be the /tmp directory or the TCP port space. Filenames or port numbers might disclose some information about

the owning process, and shared directories are subject to race-condition attacks. With polyinstantiation, each user can only see his or her own version of the resource based on username and/or security context. The security server exists to provide policy decisions, map security contexts to SIDs, provide new SIDs and manage the Access Vector Caches (AVC), It usually also provides means for loading policies and it keeps track of which subjects can access its services.

III. THREATS

Threats To Network Security

Bad practices when configuring the following aspects of a network can increase the risk of attack.

Insecure Architectures

A misconfigured network is a primary entry point for unauthorized users. Leaving a trust-based, open local network vulnerable to the highly-insecure Internet is much like leaving a door ajar in a crime-ridden neighborhood nothing may happen for an arbitrary amount of time, but eventually someone exploits the opportunity.

Broadcast Networks

System administrators often fail to realize the importance of networking hardware in their security schemes. Simple hardware such as hubs and routers rely on the broadcast or non-switched principle; that is, whenever a node transmits data across the network to a recipient node, the hub or router sends a broadcast of the data packets until the recipient node receives and processes the data. This method is the most vulnerable to address resolution protocol (ARP) or media access control (MAC) address spoofing by both outside intruders and unauthorized users on local hosts.

Centralized Servers

Another potential networking pitfall is the use of centralized computing .A common cost-cutting measure for many businesses is to consolidate all services to a single powerful machine. This can be convenient as it is easier to manage and costs considerably less than multiple-server configurations. However, a centralized server introduces a single point of failure on the network. If the central server is compromised, it may render the network completely useless or worse, prone to data manipulation or theft. In these situations,a central server becomes an open door which allows access to the entire network.

Threats To Server Security

Server security is as important as network security because servers often hold a great deal of an organization's vital information. If a server is compromised, all of its contents may become available for the cracker to steal or manipulate at will. The following sections detail some of the main issues.

Unused Services and Open Ports

A full installation of Red Hat Enterprise Linux 6 contains 1000+ application and library packages. However, most server administrators do not opt to install every single package in the distribution, preferring instead to install a base installation of packages, including several server applications. A common occurrence among system administrators is to install the operating system without paying attention to what programs are actually being installed. This can be problematic because unneeded services may be installed, configured with the default settings, and possibly turned on. This can cause unwanted services, such as Telnet, DHCP, or DNS, to run on a server or workstation without the administrator realizing it, which in turn can cause unwanted traffic to the server, or even, a potential pathway into the system for crackers.

Unpatched Services

Most server applications that are included in a default installation are solid, thoroughly tested pieces of software. Having been in use in production environments for many years, their code has been thoroughly refined and many of the bugs have been found and fixed. However, there is no such thing as perfect software and there is always room for further refinement. Moreover, newer software is often not as rigorously tested as one might expect, because of its recent arrival to production environments or because it may not be as popular as other server software.

Although these mechanisms are an effective way of alerting the community to security vulnerabilities, it is up to system administrators to patch their systems promptly. This is particularly true because crackers have access to these same vulnerability tracking services and will use the information to crack unpatched systems whenever they can. Good system administration requires vigilance, constant bug tracking, and proper system maintenance to ensure a more secure computing environment.

Inattentive Administration

Administrators who fail to patch their systems are one of the greatest threats to server security. According to the SysAdmin, Audit, Network, Security Institute (SANS), the primary cause of computer security vulnerability is to "assign untrained people to maintain security and provide neither the training nor the time to make it possible to do the job." This applies as much to inexperienced administrators as it does to overconfident or a motivated administrators. Some administrators fail to patch their servers and workstations, while others fail to watch log messages from the system kernel or network traffic. Another common error is when default passwords or keys to services are left unchanged. For example, some databases have default administration passwords because the database developers assume that the system administrator changes these passwords immediately after installation. If a database administrator fails to change this password, even an inexperienced cracker can use a widely- known default password to gain administrative privileges to the database. These are only a few examples of how inattentive administration can lead to compromised servers.

IV. COUNTERMEASURES

Shell scripting: Security is often overlooked when writing shell scripts. Many programmers ignore shell script security under the assumption that anything an attacker can do by attacking a script can be achieved more easily by simply executing the commands themselves. This is not true, however, when the script takes input from an untrusted third party:

- ✓ Shell scripts running as CGI scripts on a web server take input from the network.
- ✓ Shell scripts that read files and take actions based on their contents may take input from untrusted files.
- ✓ Shell scripts that perform web queries (with curl, for example) or other network requests may take input from untrusted servers or clients.

Further, most security problems are also correctness bugs even if someone is not trying to attack your code.

Shell script basics: Writing a shell script is like riding a bike. You fall off and scrape your knees a lot at first. With a bit more experience, you become comfortable riding them around town, but also quickly discover why most people drive cars for longer trips.Shell scripting is generally considered to be a glue language, ideal for creating small pieces of code that connect other tools together. While shell scripts can be used for more complex tasks, they are usually not the best choice.If you have ever successfully trued a bicycle wheel (or paid someone else to do so), that's similar to learning the basics of shell scripting. If you don't true your scripts, they wobble. Put another way, it is often easy to write a script, but it can be more challenging to write a script that consistently works well.This chapter and the next two chapters introduce the basic concepts of shell scripting. The remaining chapters in this document provide additional breadth and depth. This document is not intended to be a complete reference on writing shell scripts, nor could it be. It does, however, provide a good starting point for beginners first learning this black art.

Overcome vulnerability using tools:

1. Wireshark

The very first step in vulnerability assessment is to have a clear picture of what is happening on the network. Wireshark (previously named Ethereal) works in promiscuous mode to capture all traffic of a TCP broadcast domain.Customised filters can be set to intercept specific traffic; for example, to capture communication between two IP addresses, or capture UDP-based DNS queries on the network. Traffic data can be dumped into a capture file, which can be reviewed later. Additional filters can also be set during the review.Typically, the tester is looking for stray IP addresses, spoofed packets, unnecessary packet drops, and suspicious packet generation from a single IP address. Wireshark gives a broad and clear picture of what is happening on the network.However, it does not have its own intelligence, and should be used as a data provider. Due to its great GUI, any person with even some basic knowledge can use it.

2. Nmap

This is probably the only tool to remain popular for almost a decade. This scanner is capable of crafting packets and performing scans to a granular TCP level, such as SYN scan, ACK scan, etc. It has built-in signature-checking algorithms to guess the OS and version, based on network responses such as a TCP handshake.Nmap is effective enough to detect remote devices, and in most cases correctly identifies firewalls, routers, and their make and model. Network administrators can use Nmap to check which ports are

open, and also if those ports can be exploited further in simulated attacks. The output is plain text and verbose; hence, this tool can be scripted to automate routine tasks and to grab evidence for an audit report.

3. Metasploit

Once sniffing and scanning is done using the above tools, it's time to go to the OS and application level. Metasploit is a fantastic, powerful open source framework that performs rigorous scans against a set of IP addresses.

Unlike many other frameworks, it can also be used for anti-forensics. Expert programmers can write a piece of code exploiting a particular vulnerability, and test it with Metasploit to see if it gets detected. This process can be reversed technically-when a virus attacks using some unknown vulnerability, Metasploit can be used to test the patch for it.

While this is a commercial tool, I have mentioned it here because the community edition is free, yet makes no compromises on the feature set.

4. OpenVAS

The Nessus scanner is a famous commercial utility, from which OpenVAS branched out a few years back to remain open source. Though Metasploit and OpenVAS are very similar, there is still a distinct difference.OpenVAS is split into two major components — a scanner and a manager. A scanner may reside on the target to be scanned and feed vulnerability findings to the manager. The manager collects inputs from multiple scanners and applies its own intelligence to create a report.In the security world, OpenVAS is believed to be very stable and reliable for detecting the latest security loopholes, and for providing reports and inputs to fix them. A built-in Greenbone security assistant provides a GUI dashboard to list all vulnerabilities and the impacted machines on the network.

5. Aircrack

The list of network scanners would be incomplete without wireless security scanners. Today's infrastructure contains wireless devices in the data centre as well as in corporate premises to facilitate mobile users. While having WPA-2 security is believed to be adequate for 802.11 WLAN standards, misconfiguration and the use of over-simple passwords leaves such networks open to attacks.Aircrack is a suite of software utilities that acts as a sniffer, packet crafter and packet decoder. A targeted wireless network is subjected to packet traffic to capture vital details about the underlying encryption. A decryptor is then used to brute-force the captured file, and find out passwords. Aircrack is capable of working on most Linux distros, but the one in BackTrack Linux is highly preferred.

V. CONCLUSION

Thus the Linux provides the more sophisticated security mechanism for securing Linux operating system. The baseline security configuration is almost usable for most environments, but some configuration is needed in most cases. The vital role in using Linux is how you are configuring the Linux security module. Linux is a set of security policies/modules which are going to apply on the machine to improve the overall security of the machine. It is being included in most major Linux distributions, even though it might not be enabled by default. Installing or activating Linux is pretty straight- forward, and no enforcement is being done until the user has checked the log files for possible problems and decides that the configuration is good enough.

Though Linux is much secure operating system various Linux Vulnerabilities and attacks occurs because of the weak configuration, Administration, unpatched services and insecure network architecture.

VI. ACKNOWLEDGMENT

I would first like to thank prof. Chandresh D. Parekh for his mentorship and guidance on my experiment. I would like to extend a special thanks to My family members for their affection, care and encouragement. Above all I am thankfultoalmightyGodfor everything.

VII. REFERENCES

[1]. Wander boessenkool,George hacker, bruce wolfe scott mcbrien Red hat enterprise Linux, system administrator -2

[2]. Susan lauber, Philip sweany,Rudolf kastl,George hacker Red Hat System Administrator-1

[3]. C. J. PeBenito, F. Mayer, and K. MacMillan. Refer-ence Policy for Security Enhanced Linux. In SELinux Symposium, 2006. H.H. Crokell,

"Specialization and International Competitiveness," in Managing the Multinational Subsidiary, H. Etemad and L. S, Sulude (eds.), Croom-Helm, London, 1986. (book chapter style)

[4]. S.Rajeswari, Dr. A.Ramamurthy, K.T.V Subbarao, "An Efficient Intrusion-Detection Mechanisms To Protect Manet From Attacks" , International Journal of Science Engineering and Advance Technology, IJSEAT, Vol 2, Issue 12, December-2014.

[5]. P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. The Inevitabil-ity of Failure: The Flawed Assumption of Security in Modern Computing Environments. In 21st National Information Systems Security Conference, pages 303-314. NSA, 1998.R. Caves, Multinational Enterprise and Economic Analysis, Cambridge University Press, Cambridge, 1982. (book style)

[6]. R. Spencer, S. Smalley, P.Loscocco , M.Hibler, D. An- dersen, and J. Lepreau. The Flask Security Architec- ture: System Support for Diverse Security Policies. In The Eighth USENIX Security Symposium

[7]. Ashvini T. Dheshmukh1, Dr. Parikshit. N . Mahalle 2 1 Department of Computer Engineering

[8]. Crispin Cowan, Steve Beattie, Calton Pu, PerryWagle, and Virgil Gligor. SubDomain: Parsimonious Server Security. In USENIX 14th Systems Administration Conference (LISA), New Orleans, LA, December 2000