



Ethical Hacking

UNIT 3 – LINUX VULNERABILITIES

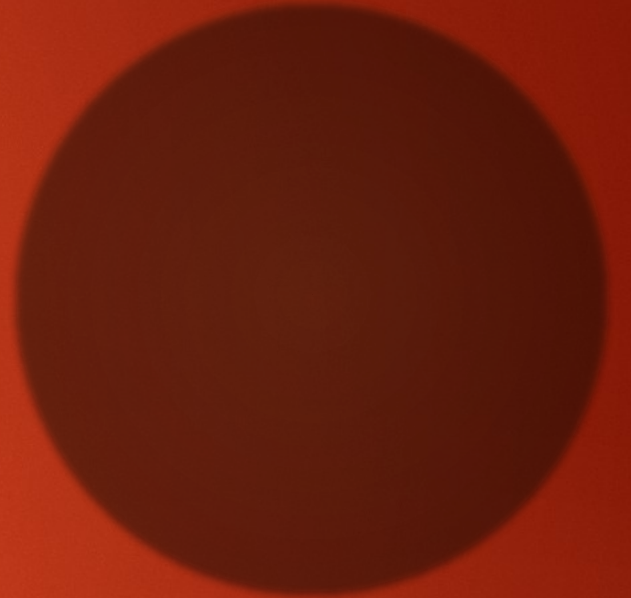
Objectives

2

- ▶ Describe the fundamentals of the Linux operating system
- ▶ Describe the vulnerabilities of the Linux operating system
- ▶ Describe Linux remote attacks
- ▶ Explain countermeasures for protecting the Linux operating system

Review of Linux Fundamentals

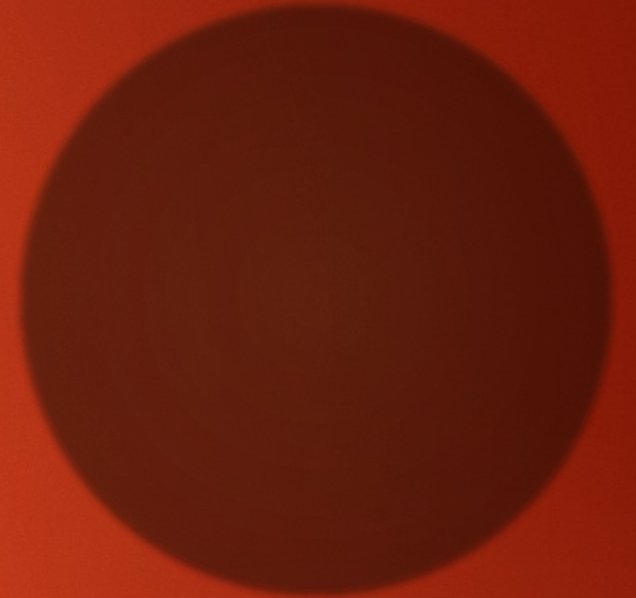
- ▶ Linux is a version of UNIX
 - ▶ Usually available free
 - ▶ Red Hat
 - ▶ Includes documentation and support for a fee
- ▶ Linux creates default directories



Linux File System

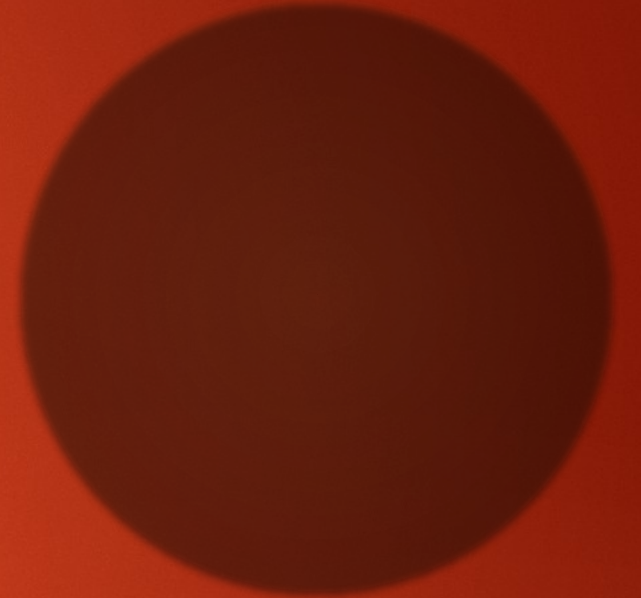
4

- ▶ Most vital part of any OS
- ▶ Provides many functions
 - ▶ Enables directories or folders organization
 - ▶ Establishes a file-naming convention
 - ▶ Includes utilities to compress or encrypt files
 - ▶ Provides for both file and data integrity
 - ▶ Enables error recovery
 - ▶ Stores information about files and folders
- ▶ *NIX systems store information about files in information nodes (inodes)



Linux File System (continued)

- ▶ Information stored in an inode
 - ▶ An inode number
 - ▶ Owner of the file
 - ▶ Group the file belongs to
 - ▶ Size of the file
 - ▶ Date the file was created
 - ▶ Date the file was last modified or read
- ▶ File systems use a fixed number of inodes
- ▶ *NIX mounts a file system as a subfile system of the root file system



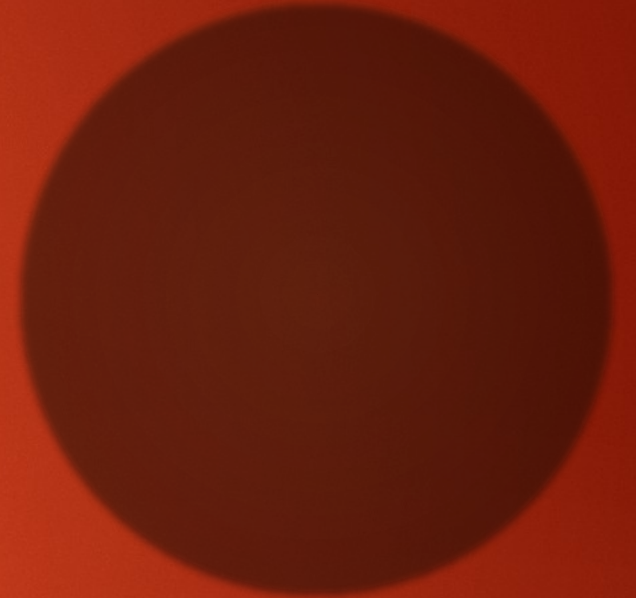
Linux File System (continued)

- ▶ *mount* command is used to mount file systems
- ▶ *df* command displays the currently mounted file systems
- ▶ File system history on *NIX systems
 - ▶ Minix file system
 - ▶ Extended File System (Ext)
 - ▶ Second Extended File System (Ext2fs)
 - ▶ Third Extended File System (Ext3fs)

Linux File System Commands

7

- ▶ Linux has file system commands for
 - ▶ Viewing files
 - ▶ Copying files
 - ▶ Moving files



Linux File System Commands (continued)

8

- ▶ Many of these commands have multiple parameters and additional functionality

Linux OS Vulnerabilities

9

- ▶ UNIX has been around for quite some time
- ▶ Attackers have had plenty of time to discover vulnerabilities in *NIX systems
- ▶ Enumeration tools can also be used against Linux systems
- ▶ Knoppix
 - ▶ A bootable, open-source version of Linux
- ▶ Nessus can be used to enumerate Linux systems

Linux OS Vulnerabilities (continued)

- ▶ Nessus can be used to
 - ▶ Discover vulnerabilities related to SMB and NetBIOS
 - ▶ Enumerate shared resources
 - ▶ Discover the root password

Linux OS Vulnerabilities (continued)

- ▶ Test Linux computer against common known vulnerabilities
 - ▶ Review the CVE and CAN information
- ▶ Differentiate between local attacks and remote attacks
 - ▶ Remote attacks are harder to perform

Remote Access Attacks on Linux Systems

- ▶ Attacking a network remotely requires
 - ▶ Knowing what system a remote user is operating
 - ▶ The attacked system's password and login accounts

Footprinting an Attacked System

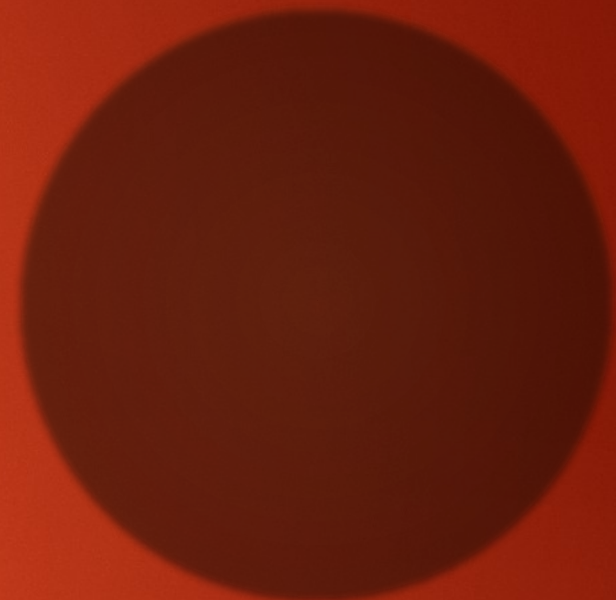
13

- ▶ Footprinting techniques
 - ▶ Used to find out information about a target system
- ▶ Determining the OS version the attacked computer is running
 - ▶ Check newsgroups for details on posted messages
 - ▶ Knowing a company's e-mail address makes the search easier
- ▶ Other footprinting tools include: Whois databases, DNS zone transfers, Nessus, and port scanning tools

Using Social Engineering to Attack Remote Linux Systems

14

- ▶ Goal
 - ▶ To get OS information from company employees
- ▶ Common techniques
 - ▶ Urgency
 - ▶ Quid pro quo
 - ▶ Status quo
 - ▶ Kindness
 - ▶ Position
- ▶ Train your employees about social engineering techniques



Installing Trojan Programs

15

- ▶ Trojan programs spread as
 - ▶ E-mail attachments
 - ▶ Fake patches or security fixes that can be downloaded from the Internet
- ▶ Trojan program functions
 - ▶ Allow for remote administration
 - ▶ Create a FTP server on attacked machine
 - ▶ Steal passwords
 - ▶ Log all keys a user enters, and e-mail results to the attacker

Installing Trojan Programs (continued)

- ▶ Linux Trojan programs are sometimes disguised as legitimate programs
- ▶ Trojan programs can use legitimate outbound ports
 - ▶ Firewalls and IDSs cannot identify this traffic as malicious
 - ▶ Example: Sheepshank
- ▶ It is easier to protect systems from already identified Trojan programs
 - ▶ Trojan.Linux.JBellz
 - ▶ Remote Shell
 - ▶ Dextenea

Installing Trojan Programs (continued)

▶ Rootkits

- ▶ Contain Trojan binary programs ready to be installed by an intruder with root access to the system
- ▶ Attacker hide the tools used for later attacks
- ▶ Replace legitimate commands with Trojan programs
- ▶ Example: LRK5
- ▶ Security testers should check their Linux systems for rootkits
 - ▶ Rootkit Hunter
 - ▶ Chkrootkit

Creating Buffer Overflow Programs

- ▶ Buffer overflows write code to the OS's memory
 - ▶ Then run some type of program
 - ▶ Can elevate the attacker's permissions to the level of the owner
- ▶ Security testers should know what a buffer overflow program looks like

Creating Buffer Overflow Programs (continued)

- ▶ A C program that causes a buffer overflow

[illegible]

Creating Buffer Overflow Programs (continued)

- ▶ A C code snippet that fills the stack with shell code

```
include <string.h>
char info[] = "\xeb\x2a\x08\x09\x00\x00\x89\x05\xed\xff"
              "\xeb\x2a\x03\x09\x00\x00\x89\x05\xed\xff"
              "\xeb\x2a\x02\x09\x00\x00\x89\x05\xed\xdf"
"\xeb\x2a\x08\x01\x01\x02\x83\x04\xef\xcf"
              "\x6e\x2a\x08\x09\x00\x00\x89\x05\xed\xff"
              "\xeb\x2a\x08\x11\x00\x00\x89\x05\xed\xae"
"\xca\x2a\x08\x09\x00\x00\x67\xcc\xed\xef"
              "\x22\x2a\x08\xaa\x00\x00\xff\x05\xed\xff"
              "\xdd\x2a\x08\x09\x00\x00\xcc\x05\xed\xcd"
"\xfa\x2a\x08\x09\x00\x00\x00\x11\xed\xde"
              "\xeb\x2a\x08\x09\x00\x00\x89\x05\xed\xff"
main()
{ //Contains a function that copies the info[] array into a
  //buffer area.
    [Remaining code omitted]
}
```


Creating Buffer Overflow Programs (continued)

- ▶ Guidelines to help reduce this type of attack
 - ▶ Write code that avoids functions known to have buffer overflow vulnerabilities
 - ▶ strcpy()
 - ▶ strcat()
 - ▶ sprintf()
 - ▶ gets()
 - ▶ Configure OS to not allow code in the stack to run any other executable code in the stack
 - ▶ Use compilers that warn programmers when functions listed in the first bullet are used

Using Sniffers to Gain Access to Remote Linux Systems

- ▶ Sniffers work by setting a network card adapter in promiscuous mode
 - ▶ NIC accepts all packets that traverse the network cable
- ▶ Attacker can analyze packets and learn user names and passwords
 - ▶ Avoid using protocols such as Telnet, HTTP, and FTP that send data in clear text
- ▶ Sniffers
 - ▶ Tcpcat, Ethereal

Countermeasures Against Linux Remote Attacks

- ▶ Measures include
 - ▶ User awareness training
 - ▶ Keeping current on new kernel releases and security updates

User Awareness Training

24

- ▶ Social Engineering
 - ▶ Users must be told not to reveal information to outsiders
 - ▶ Make customers aware that many exploits can be downloaded from Web sites
 - ▶ Teach users to be suspicious of people asking questions about the system they are using
 - ▶ Verify caller's identity
 - ▶ Call back technique

Keeping Current

25

- ▶ Never-ending battle
 - ▶ New vulnerabilities are discovered daily
 - ▶ New patches are issued to fix new vulnerabilities
- ▶ Installing these fixes is essential to protecting your system
- ▶ Many OSs are shipped with automated tools for updating your systems
 - ▶ Red Hat Update Agent

Summary

26

- ▶ File systems store and manage user data and system data
- ▶ Linux uses default directories to store user data and system data
- ▶ Extended File System (Ext) is Linux default file system
- ▶ Information about *NIX files are stored in inodes
- ▶ Vulnerabilities of the Linux OS can be determined by the use of security tools and from the CVE Web site

Summary (continued)

27

- ▶ Techniques for remotely attacking Linux systems
 - ▶ Footprinting
 - ▶ Social engineering
 - ▶ Trojan programs
 - ▶ Buffer overflows
- ▶ Social engineering can be the most effective way to gather information
- ▶ Countermeasures to Trojan programs include
 - ▶ Remove any unneeded services
 - ▶ Apply test security updates

Summary (continued)

28

- ▶ Countermeasures to buffer overflows include
 - ▶ Writing secure code
 - ▶ Preventing code from being run in the stack
 - ▶ Using compilers that warn when a function is dangerous or risky
- ▶ Other countermeasures
 - ▶ Employee training
 - ▶ Keeping systems updated