JOVITA
CYBER SECURITY

SECURITY REPORT

Presented for :
Mosse Institute

Author :
Jovita Kusuma
(ID : q3OZ4UDFR5dehjlnGg5dSKsET1f2)

Date :
2025-02-28

Version:
1.0

Contents

JOVITA
CYBER SECURITY

1.Revision History

Version	Issue Date	Issued By	Comments
0.1	27 February 2025	Jovita Kusuma	Initial Draft
0.2	28 February 2025	Jovita Kusuma	Revised Draft

2.Introduction

In today's interconnected digital landscape, maintaining robust cybersecurity is paramount. Vulnerability scanning tools play a crucial role in identifying weaknesses within systems and networks, enabling proactive mitigation of potential threats. OpenVAS, a powerful open-source vulnerability scanner, has become a staple for security professionals.

However, traditional installation methods can present challenges due to complex dependencies and configurations. This exercise addresses these challenges by leveraging Docker containerization, offering a streamlined and efficient approach to deploying and utilizing OpenVAS. By encapsulating OpenVAS within a Docker container, users can bypass intricate installation procedures and quickly initiate vulnerability assessments, enhancing their ability to secure their digital assets.

Docker has revolutionized software development and deployment by providing a platform for containerization.

How Docker Works:

- **Docker Images:** These are read-only templates that define the contents of a container. They contain the application code, dependencies, and instructions for running the application.
- **Docker Containers:** These are running instances of Docker images.
- **Docker Engine:** This is the core component of Docker that manages the creation, running, and management of containers.
- **Docker Hub:** This is a cloud-based registry for storing and sharing Docker images.

Docker simplifies the process of building, shipping, and running applications. It provides a consistent and efficient way to deploy software, making it an essential tool for modern software development and operations.

3.Methodology

This report details the process and findings of a vulnerability assessment conducted using OpenVAS, deployed via Docker on an Ubuntu system. The objective was to evaluate the security posture of a target Windows 7 machine.

3.1. Setup of Testing Environment

The following table details the setup of the testing environment for this engagement:

Operating Systems	Application Installed
Kali Linux 2024.4 (Attacker Machine)	Metasploit
Ubuntu 22.4 (Virtual Machine)	Docker, Gparted, OpenVAS
Windows 7 (Victim Machine)	-

3.2. Replication of the Vulnerability Scanning

The following screenshots demonstrate the conducting vulnerability scans using OpenVAS and interpreting the results :

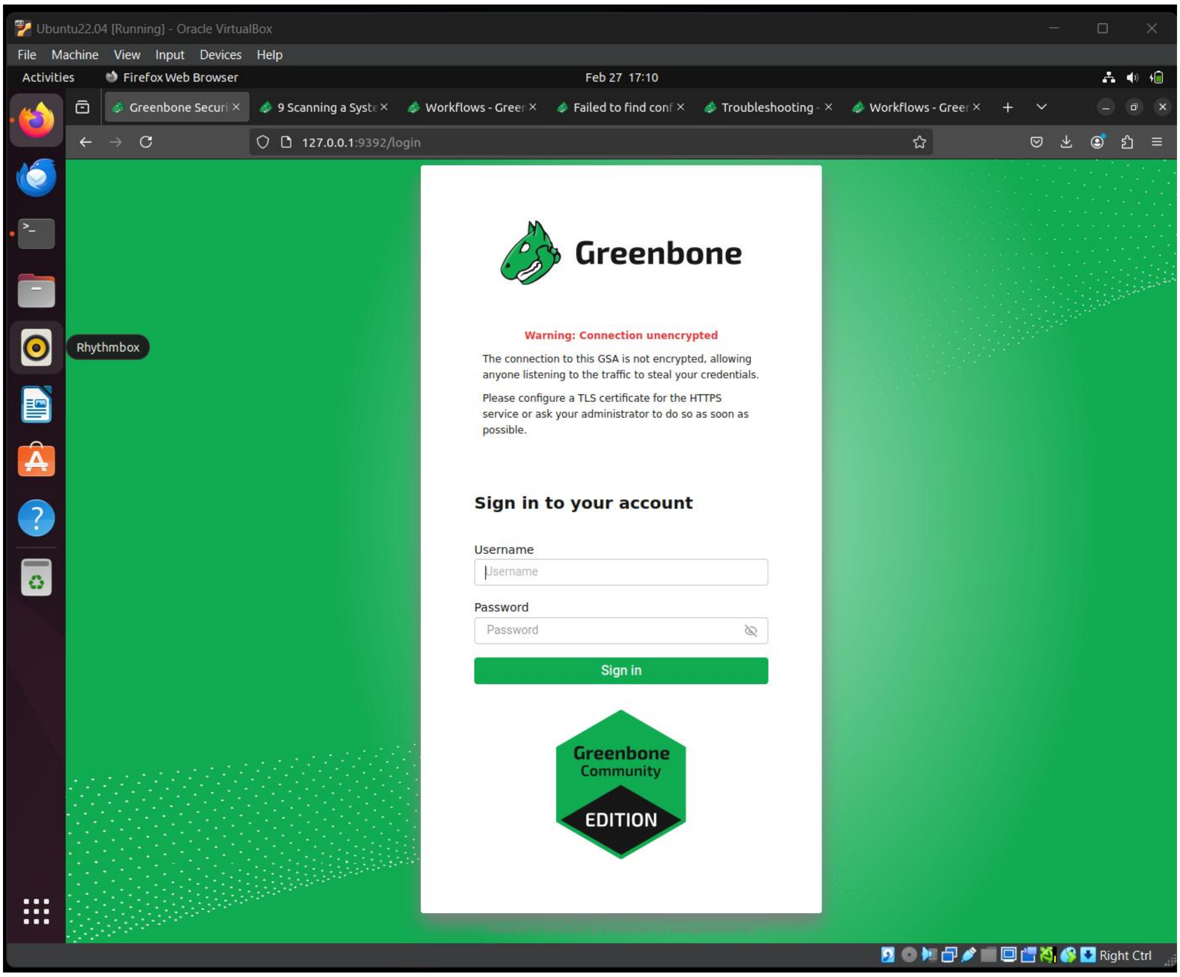

Image 1. Greenbone Security Assistant (GSA) Login Page on Ubuntu Virtual Machine

The assesment followed these steps:

Step 1: Docker Installation

The security assessment began with the installation of **Docker** [1] on an **Ubuntu 20.04** [2] system. The installation followed the **DigitalOcean tutorial** [3], ensuring a stable environment for deploying security tools.

Step 2: OpenVAS Deployment

To conduct vulnerability scanning, **Greenbone Community Containers for OpenVAS** were deployed using Docker. During the deployment, an issue was encountered due to insufficient disk space, triggering **"disk memory low"** errors. This issue was resolved by increasing the virtual disk space to **500GB** using **Gparted** [4].

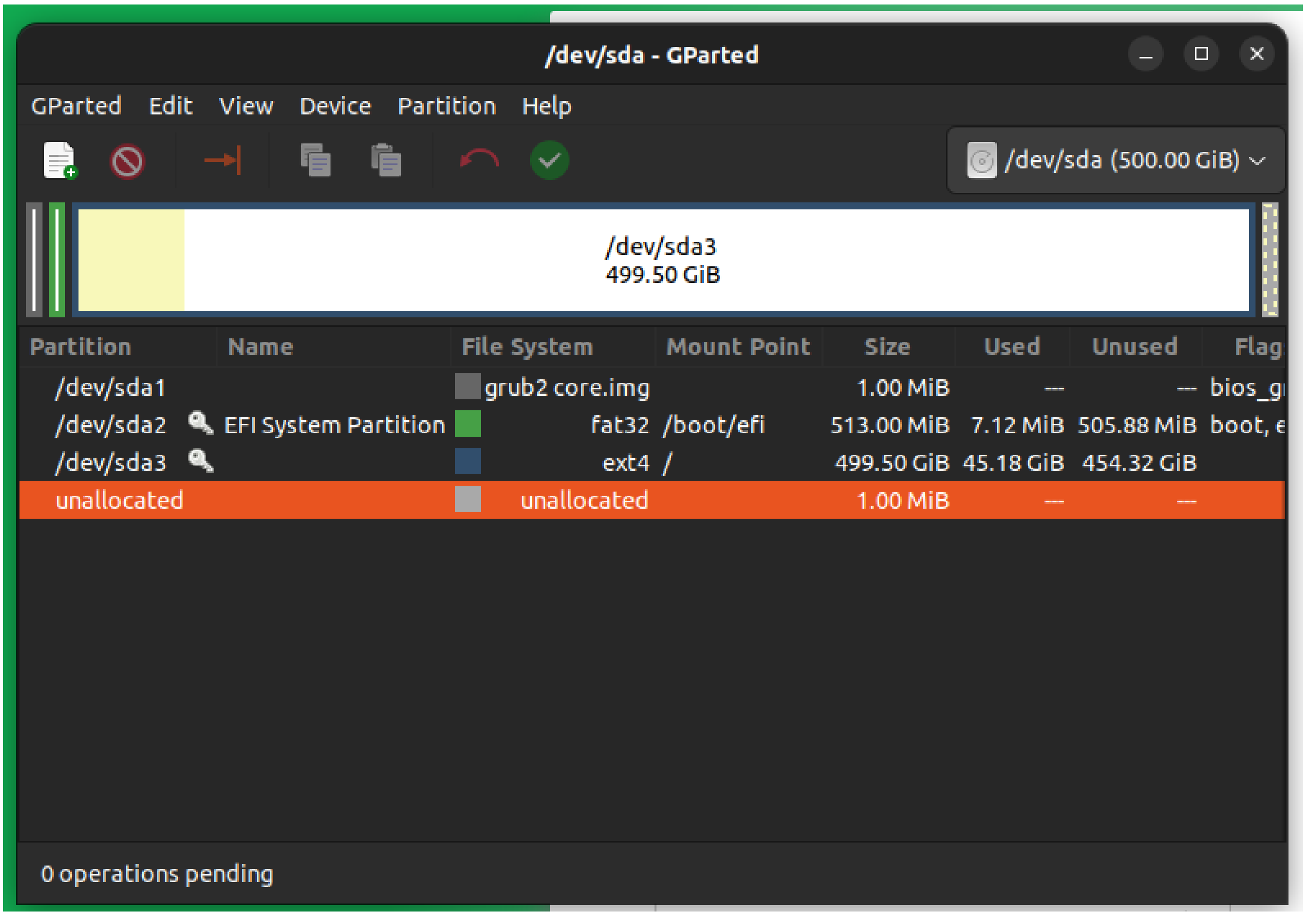

Image 2. Gparted Partition Manager Showing Disk Partitions on Ubuntu

Once resolved, the OpenVAS containers were successfully pulled and executed.

Step. 3: Feed Update

As recommended, the OpenVAS feed was allowed to complete its initial update process to ensure accurate vulnerability data.

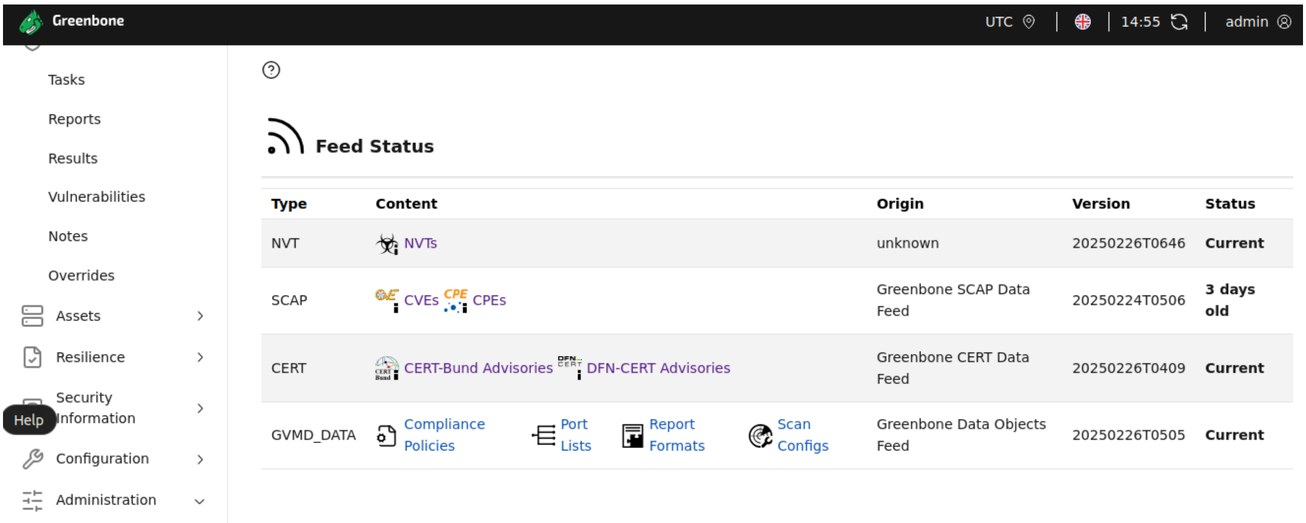

Image 3. Greenbone Security Assitant (GSA) – Feed Status Overview

Step 4: Launching Metasploit

Once OpenVAS confirmed the vulnerability, the **Metasploit Framework** [5] was launched in **Kali Linux** [6]. The **EternalBlue exploit module** was selected to target **MS17-010**, which allows remote code execution via SMBv1.

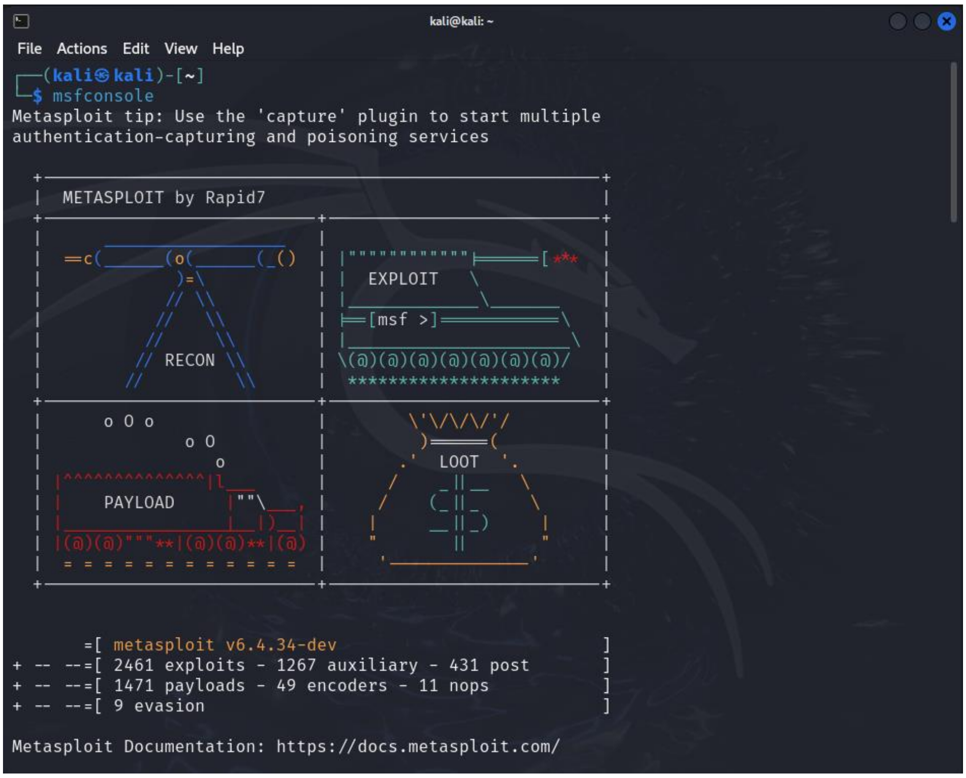

Image 4. Metasploit Framework Console in Kali Linux

Step 5: Target Identification and Exploit Execution

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > show options
```

Image 5. Using the EternalBlue Exploit in Metasploit Framework

The target machine, **Windows 7 (IP: 10.0.2.5)**, was confirmed to be vulnerable. The exploit was executed, which successfully:

1. **Established a connection** to the victim machine over port **445 (SMB)**.
2. **Sent crafted SMB packets** to exploit the memory corruption vulnerability.
3. **Injected malicious code** to gain remote access.
4. **Opened a Meterpreter session**, granting full control over the system.

```
meterpreter > sysinfo
Computer        : WINDOWS7
OS              : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
```

Image 6. System Information Retrieved via Meterpreter on Windows 7

Step 6: Target Scan

A vulnerability scan was performed against a Windows 7 machine with its firewall disabled.

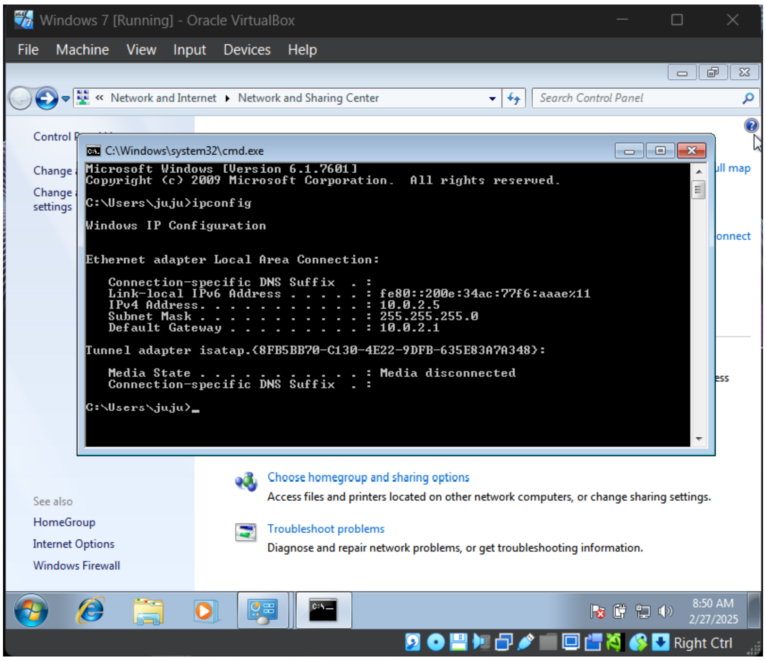

Image 7. Windows 7 IP Configuration in VirtualBox Using Command Prompt

Detection Method

The OpenVAS scan identified that the target machine responds to **ICMP Timestamp Requests (Type 13)** [7] and provides a **Timestamp Reply (Type 14)**.

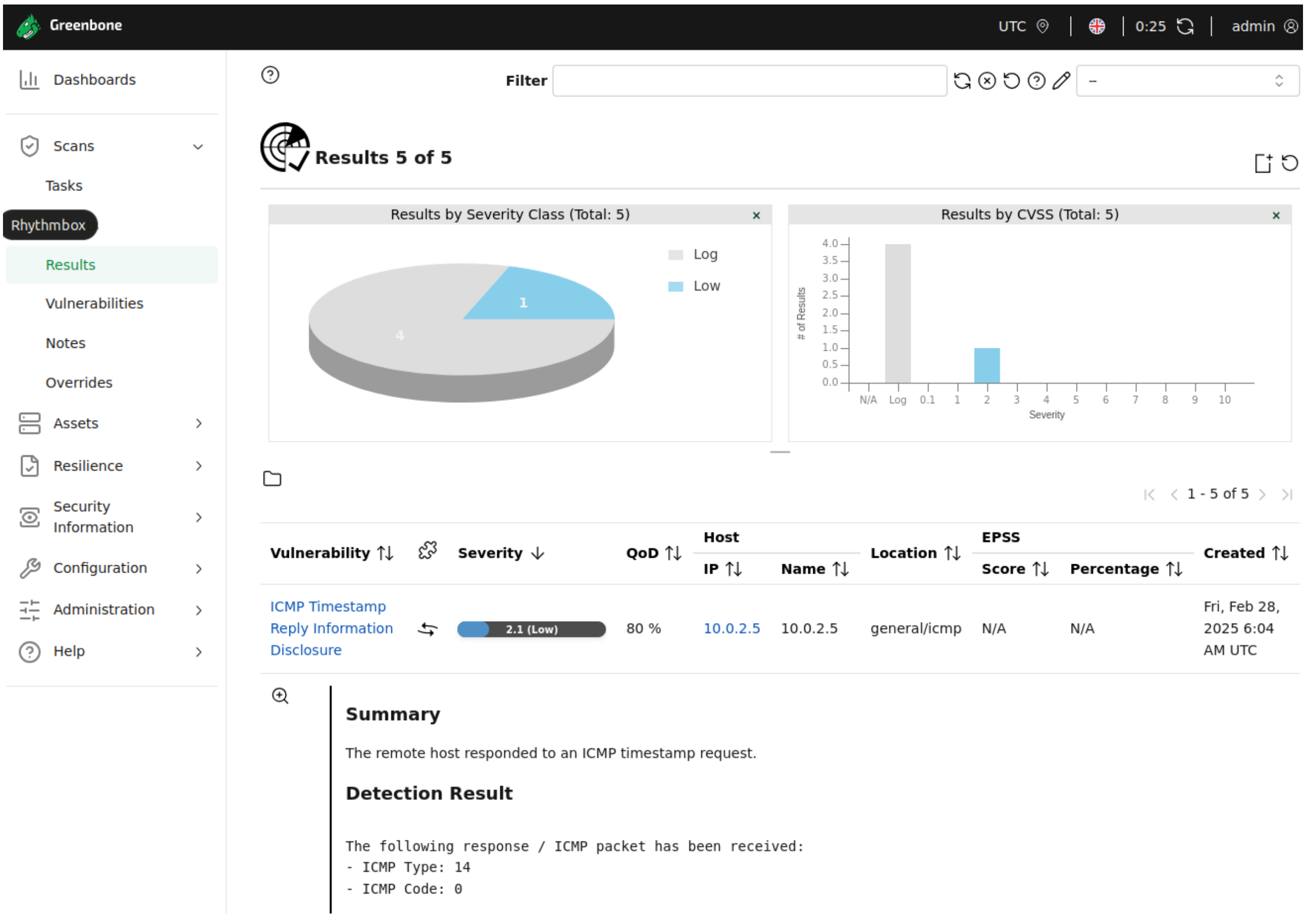

Image 8. Greenbone Security Assistant (GSA) Scan Results

Impact

This information could be exploited by attackers to infer the system uptime and potentially **exploit weak time-based random number generators** in cryptographic operations.

Solution

To mitigate this vulnerability, the following actions should be taken:

- **Disable ICMP Timestamp responses** on the remote host.
- **Protect the system with a firewall** and block ICMP timestamp requests from untrusted networks.

References

- **CVE-1999-0524** [8]
- **CERT Advisories:** DFN-CERT-2014-0658, CB-K15/1514, CB-K14/0632

4. Findings

ICMP Timestamp Vulnerability Risks

Although not as severe as MS17-010, the **ICMP Timestamp vulnerability** presents a security risk, as attackers can use system timestamps for **fingerprinting and reconnaissance** before launching more sophisticated attacks.

5. Conclusion

Key Takeaways

1. **Unpatched Windows 7 systems remain highly vulnerable** to EternalBlue. Organizations should update to supported Windows versions and disable SMBv1.
2. **OpenVAS proved effective in identifying vulnerabilities**, highlighting the importance of regular vulnerability scanning.
3. **ICMP Timestamp responses should be disabled** to prevent reconnaissance-based attacks.

Recommendations

- **Patch Windows systems**: Install security updates to mitigate MS17-010.
- **Disable SMBv1**: Prevent future EternalBlue attacks by enforcing SMBv2 or SMBv3.
- **Use strong password policies**: Hashes can be cracked; enforcing multi-factor authentication (MFA) can reduce risks.
- **Conduct regular vulnerability assessments**: OpenVAS scans should be automated and reviewed periodically.
- **Block ICMP Timestamp Requests**: Disable the feature or use a firewall to filter ICMP traffic.
- **Monitor network traffic for anomalies**: EternalBlue exploitation can often be detected by unusual SMB traffic spikes.

This assessment highlights the importance of proactive security measures. By combining **OpenVAS vulnerability scanning** with **Metasploit exploitation**, organizations can better understand their security risks and take **preventive actions** before attackers exploit vulnerabilities.

6. References

[1] https://docs.docker.com/

[2] https://ubuntu.com/tutorials

[3] https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

[4] https://gparted.org/

[5] https://www.metasploit.com/

[6] https://www.kali.org/get-kali/#kali-virtual-machines

[7] https://www.tenable.com/plugins/nessus/10114

[8] https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-1999-0524