**UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI**

**INTRUSION DETECTION AND PREVENTION SYSTEMS**

Report

**NVT: Weak Host Key Algorithm(s) (SSH)**

Nguyễn Tài Anh – 22BI13028 – Cyber Security

**Lecturers: Assoc. Prof. Pham Thanh Giang, Mr. Tran Dai Duong**

Hanoi, October 29, 2024

**A. Introduction to this vulnerability**

1. **What is this vuln and type of vulnerability is this?**
2. What is this vulnerability?

* NVT: Weak Host Key Algorithm(s) (SSH)
* The remote SSH server is configured to allow / support weak host key algorithm(s).

1. Type of vulnerability

* Server-side vulnerability

1. **Outline the technical mechanism of the vulnerability.**
2. Key Exchange Vulnerabilities

* SSH sessions start with a key exchange process to establish a secure communication channel.
* In weak key exchanges, outdated algorithms like Diffie-Hellman Group1 (1024-bit) are often used, which can be vulnerable to man-in-the-middle (MITM) or logjam attacks, as they rely on short key lengths that can be brute-forced.
* Attackers can intercept this exchange, potentially deriving the shared session key if the computation requirements are low due to a weak key size.

1. Cipher Vulnerabilities

* SSH uses symmetric encryption to encrypt data after the key exchange. If weak ciphers (such as 3DES, Blowfish, or AES-128-CBC) are used, an attacker might break the encryption by leveraging known cryptographic weaknesses, such as padding oracle attacks or cipher block chaining (CBC) vulnerabilities.
* Weak ciphers make the encrypted data more susceptible to cryptographic attacks, like known-plaintext or chosen-ciphertext attacks, which could reveal sensitive information or the entire session’s contents.

1. Hashing Algorithm Weaknesses

* Hash functions are used in SSH for creating digital signatures and integrity checks. However, older algorithms like MD5 and SHA-1 have known vulnerabilities (e.g., susceptibility to collision attacks).
* If these weak algorithms are used for message authentication (HMAC), attackers could potentially forge or tamper with messages by exploiting hash collisions, which undermine the integrity of the SSH session.

1. Small Key Sizes

* SSH keys generated with smaller bit sizes, such as RSA-1024 or DSA-1024, are easier to crack with modern computational power.
* If an attacker can break these keys, they could impersonate the server or client, hijack sessions, or decrypt past communications if they have been captured.

1. **Impact and Severity**
2. Potential Impact

* Increased Risk of Man-in-the-Middle (MitM) Attacks.
* Brute-Force Attacks on SSH Keys
* Compromised Confidentiality and Integrity of Data
* Potential Access to the System via Weak Authentication

1. Severity level

* CVSS: 5.3
* Quality of Detection (QoD): 80%

**B. Implementation**

1. **Labwork 1: Creating environment for testing**
2. Requirement

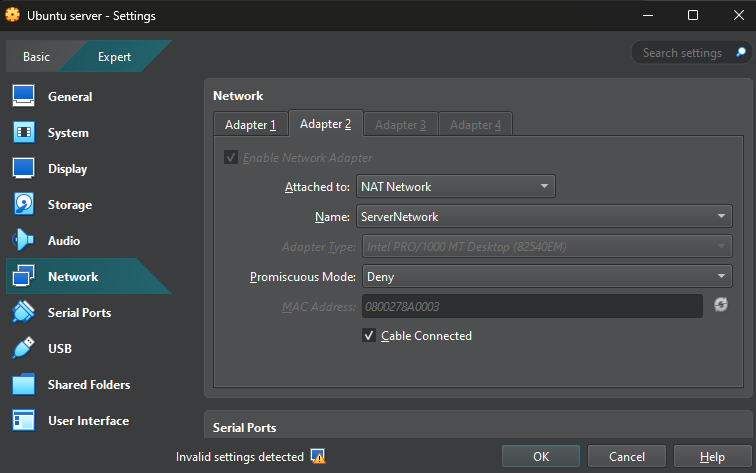
* VirtualBox or VMware
* OVA or ISO file for Kali, Ubuntu server, Metasploitable 2

1. Configure virtual machine network

A screenshot of a computer

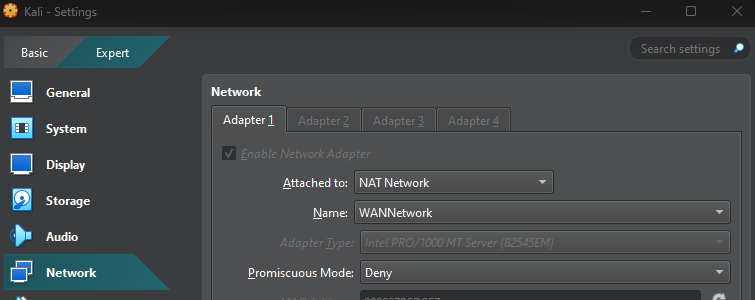
Description automatically generated

* For Ubuntu server, I will create 2 network adapters, both using NAT Network, one connects to ServerNetwork, the other connect to WANNetwork

A computer screen shot of a network

Description automatically generated

* For Kali, I will connect to WANNetwork and for Metasploitable 2, I will connect to ServerNetwork, both using NAT Network

A screenshot of a computer

Description automatically generated

* After configure network outside in VirtualBox, continue to configure the IPs inside the virtual machine 🡪 Result:

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* Kali: **10.10.1.5**
* Metasploitable 2: **172.16.1.6**
* Ubuntu server: Gateway **10.10.1.1** and **172.16.1.1**
* Successfully ping these machines together

A screenshot of a computer

Description automatically generated

1. **Labwork 2: Security threats and scanning**
2. Using nmap (Network Mapper): Scan for ports

* nmap: Host Discovery

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* nmap: Port Scanning
  + SYN packets ***“sudo nmap -sS 172.16.1x”***
  + Full TCP connections ***“sudo nmap -sT 172.1.16.x”***
  + ACK packets ***“sudo nmap -sA 172.16.1.x”***
  + Fin packets ***“sudo nmap -sF 172.16.1.x”***
  + Basic 100 UDP ports ***“sudo nmap -sU 172.16.1.x”***
  + Not scan any port, only host discovery ***“sudo nmap -sn 172.16.1.x”***

A screenshot of a computer program

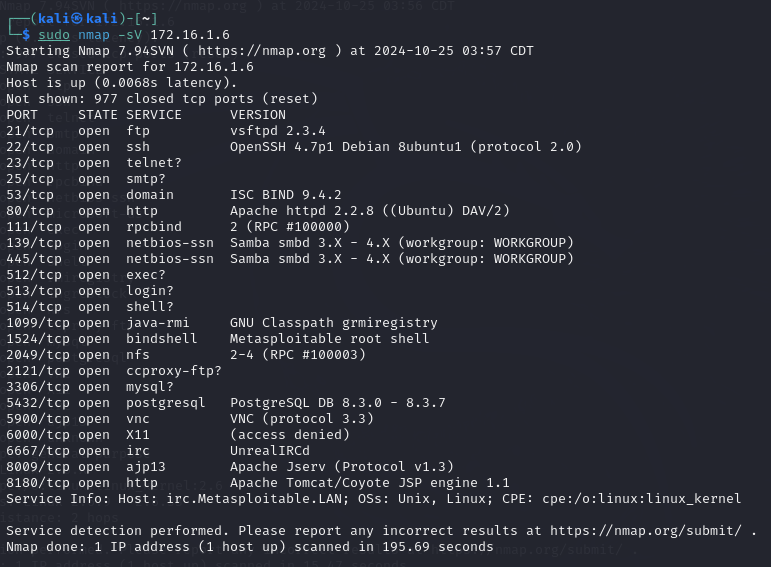
Description automatically generated

* nmap: OS Detection ***“sudo nmap -O 172.16.1.x”***

A screenshot of a computer program

Description automatically generated

* nmap: Service Detection ***“sudo nmap -sV 172.16.1.x”***



1. Using GVM (Greenbone Vulnerability Management): Scan for threats

* Architecture: Maily consists of the following three components
  + GSA (Greenbone Security Assistants)
  + GVM (Greenbone Vulnerability Management)
  + OpenVAS Scanner
* Installation:
  + Install GVM: ***“sudo apt install gvm”***
  + Setup GVM: ***“sudo gvm-setup”***

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* + Setup username and password for GVM: ***“sudo -u \_gvm --gvmd --user=admin --new-password=letmein”***

***A blue and white text on a black background

Description automatically generated***

A screenshot of a computer program

Description automatically generated

* + Start gvm: ***“sudo gvm-start”***

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

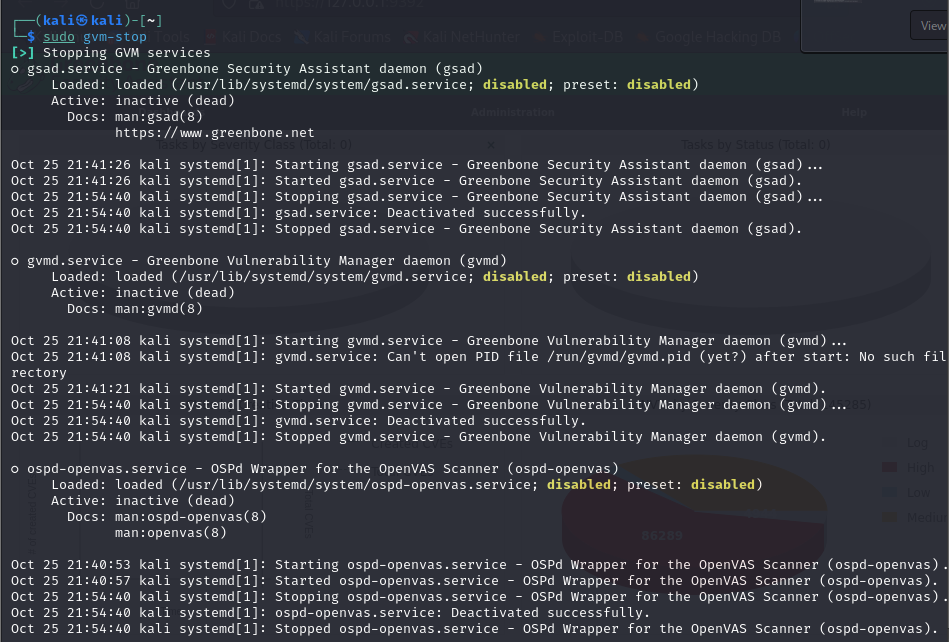
A screenshot of a computer

Description automatically generated

* + Check if GVM start successfully: ***“sudo ss -lntp”***



* + Stop GVM: ***“sudo gvm-stop”***



* GVM – Usage
  + Create a target

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* + - As can be seen, my Metasploitable machine has IP 172.16.1.6 and it’s OpenSSH connection is at port 22
    - Port List using: All IANA assigned TCP (can check at Port Lists in Configuration tab)
  + Create a task

A screenshot of a computer

Description automatically generated

* + - Navigate to Scans tab, choose “Tasks” and click to “\*” icon to create a new task

A screenshot of a computer

Description automatically generated

* + - In Scan Targets, choose the target you have just created.



* + Start the task
    - To run this task, click on the “start” button (triangle one)
    - After the task is started, the web UI of GVM will automatically refresh about the progress of the scan.







* + Report

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a document

Description automatically generated

1. Exploitation **NVT: Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)**

* Connect to Metasploitable VM using SSH
  + Start SSH in Metasploitable by using command ***“sudo nano /etc/rc.local”***, and then add a line ***“/etc/init.d/ssh start”*** to start SSH service each time Metasploitable VM is booted.
  + In Kali, because Kali cannot detect and matching host key, so we have to use command ***“ssh -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedAlgorithms=+ssh-rsa*** [***msfadmin@172.16.1.6***](mailto:msfadmin@172.16.1.6)***”*** to connect to Metasploitable VM using SSH service.

A computer screen with white text

Description automatically generated

* Start Metasploit Framework
  + Metasploit relies on a database to speed up searches. Ensure the database is started with
    - ***“sudo service postgresql start”***
    - ***“sudo msfdb init”***
  + Using command ***“msfconsole”*** to start Metasploit framework

A computer screen shot of a computer

Description automatically generated

* + Using ***“search ssh”***

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

* + Set up the brute-force module

A computer screen shot of a computer

Description automatically generated

* + - Using command ***“show options”*** to check for everything that has been changed.

A screenshot of a computer

Description automatically generated

* + Searching for currently running SSH sessions
    - Using command ***“sessions -l”***

A computer screen with white text

Description automatically generated

* + Interaction with the currently running SSH connections
    - Using command ***"session -i [session\_id]”***

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

* + - As can be seen, I can interact with the SSH connection from outside of the main user terminal, it means that I can also interact with any SSH connections that are connected to the Metasploitable VM, and of course steal or cat data from them.

1. **Labwork 3: Mitigation and Prevention**
2. Detection (Next Lab)
3. Prevention (Block the port 22 immediately / Blocking Hacker’s IP)

* Blocking the port 22 immediately
  + Using Iptables to set rule for the Ubuntu server (firewall) to block all SSH connections to the Metasploitable2 VM
    - Command: ***“sudo iptables -A FORWARD -d 172.16.1.6 -p tcp --dport 22 -j REJECT”***



* Result:

A screen shot of a computer

Description automatically generated

* + Optional: We can also set the allow IP that can SSH to the machine by using command: ***“sudo iptables -I FORWARD -s 10.10.1.6 -d 172.16.1.6 -p tcp --dport 22 -j ACCEPT”***
* Result: The machine with IP 10.10.1.6 (In my case is Kali) now can SSH to the Metasploitable2 machine while others still be rejected.
* Save the rules
  + Command: ***“sudo netfilter-persistent save”***
  + The iptables-persistent service will automatically load saved rules on boot. If you want to manually reload saved rules, you can use: ***“sudo netfilter-persistent reload”***
  + Save rule manually: ***“sudo iptables-save > /etc/iptables/rules.v4”***
  + Load Rules on Boot Using “/etc/rc.local”: ***“iptables-restore < /etc/iptables/rules.v4”***
  + Check if rules are in place: ***“sudo iptables -L -v -n”***

A computer screen with white text

Description automatically generated

* + Do not need to block each hackers’ IPs, but no one can freely access the server through SSH connections except when the server allow an specific IP to connect to.
* Blocking directly hackers’ Ips
  + Using tcmdump to catch all the SSH connections that are sending to the Metasploitable2 VM

A screenshot of a computer

Description automatically generated

* + Here I have two VMs with IP 172.16.1.6 and 172.16.1.8 are SSH to Metasploitable2, and we can see that there are these IPs’ log show on the tcpdump.
  + Use iptables log to log the hackers’ IPs: ***“sudo iptables -A INPUT -p tcp --dport 22 -j LOG --log-prefix "SSH attempt: " --log-level 4”***
  + Blocking hackers’ IPs by using command: ***“sudo iptables -A INPUT -s 172.16.1.8 -j DROP”*** or ***“sudo iptables -A INPUT -s 172.16.1.9 -j DROP”***
  + Do not need to block the whole port 22, but each time being attacked, we have to log again to check the hackers’ IPs and block them.

**C. Conclusion**

1. Vulnerability Description:

* This NVT detects SSH configurations using weak or outdated cryptographic algorithms, such as low-bit RSA or DSS (DSA) keys, and hash algorithms like MD5 or SHA-1.

1. Technical Mechanism:

* SSH keys with insufficient bit strength (e.g., RSA < 2048 bits) or deprecated algorithms (e.g., DSS/DSA) are easier to attack due to advancements in computing power and known cryptographic weaknesses.

1. Impact

* Man-in-the-Middle (MitM) Risk: Weak algorithms allow attackers to intercept or impersonate SSH connections.
* Brute-Force Susceptibility: Weaker encryption makes SSH connections more vulnerable to brute-force attacks.
* Data Confidentiality and Integrity Threat: Compromised encryption endangers the security of data transferred over SSH.
* Compliance Risks: Weak algorithms may violate security standards like PCI-DSS or GDPR.
* System Access Vulnerability: Weak host keys can potentially allow attackers to bypass authentication, leading to unauthorized access.

1. Mitigation

* Update the SSH configuration to use strong algorithms (e.g., ecdsa-sha2-nistp256, ed25519) and a minimum of RSA-2048 if RSA keys are used.
* Block SSH port 22 at the firewall if SSH access is unnecessary to prevent exploitation.

1. Practical Example

* Setup a system with Kali Linux as client (attacker), Ubuntu as a firewall, and Metasploitable2 as the target, blocking SSH traffic on the firewall or updating SSH settings on Metasploitable2 (if possible) could mitigate the risk of exploitation.

**D. References**

<https://www.rfc-editor.org/rfc/rfc8332>

<https://www.rfc-editor.org/rfc/rfc8709>

<https://www.rfc-editor.org/rfc/rfc4253#section-6.6>

<http://www.securityspace.com/smysecure/catid.html?id=1.3.6.1.4.1.25623.1.0.117687>

<https://www.linkedin.com/pulse/weak-host-key-algorithm-vulnerability-mikrotiks-ssh-alves-pereira>

<https://security.stackexchange.com/questions/131010/which-host-key-algorithm-is-best-to-use-for-ssh>