Tar-Pit Challenge

Executive Summary

This report details the methodology employed to successfully complete the "Tar-Pit" Boot2Root challenge. The solution was achieved through an **offline forensic analysis** of the provided virtual machine disk image, which revealed a critical chain of operational security failures. Initial access credentials were recovered from mismanaged configuration files, leading to the identification of a severe **sudo misconfiguration** allowing the backup user to execute the /usr/bin/tar utility with NOPASSWD privileges as the root user. This vulnerability was leveraged to read the final flag, confirming the successful compromise of the system.

Phase I: Target Acquisition and Forensic Setup

The challenge began with the acquisition of a compressed virtual machine image, which necessitated a forensic approach to initial enumeration.

1.1 Disk Image Preparation

The provided tar-pit.zip archive contained a VMDK disk image (b2r-lab-disk001.vmdk). To facilitate direct filesystem analysis, the proprietary VMDK format was converted to a raw disk image using the qemu-img utility.

```
qemu-img convert -f vmdk -O raw tar-pit_extracted/b2r-lab-disk001.vmdk
b2r-lab.raw
```

1.2 Filesystem Mounting

The raw image was analyzed to determine the starting sector of the primary Linux filesystem partition. The fdisk utility identified the main partition (b2r-lab.raw2) starting at sector 4096.

Device	Start Sector	End Sector	Size	Type
b2r-lab.raw2	4096	52426751	25G	Linux filesystem

The required byte offset for mounting was calculated as:
$$ \text{Offset} = \text{Start Sector} \times \text{Sector Size} = 4096 \times 512 = 2,097,152 \text{ bytes} $$

The filesystem was then mounted in a read-only loopback configuration:

```
sudo mount -o loop,offset=2097152 b2r-lab.raw /mnt/tar-pit_fs
```

Phase II: Deep Dive Enumeration and Initial Vector Identification

With the filesystem mounted at /mnt/tar-pit_fs, a comprehensive review of system configuration files and user directories was performed to identify initial access credentials and potential privilege escalation vectors.

2.1 User Account Mapping

Analysis of the /etc/passwd file revealed three non-standard user accounts, suggesting multiple potential points of entry.

User	UID	GID	Home Directory	Shell
dev	1000	1000	/home/dev	/bin/bash
backuup	1001	1001	/home/backuup	/bin/sh
backup	1002	1002	/var/backups	/bin/bash

2.2 Credential and Key Recovery

Two distinct instances of plaintext credential leakage were discovered, highlighting severe operational security lapses.

Vulnerability	Path	Credentials	Context
Web Configuration Leak	/var/www/html/assets/.backup/config.old	dev:dev@2022	An old development configuration file left in a publicly accessible web directory.
Backup Configuration Leak	/etc/backup.conf	backup:backup@2022	A dedicated backup configuration file, likely created by the root user and containing sensitive credentials.

Furthermore, the home directory of the backup user contained a private SSH key, providing a third, highly reliable initial access vector.

Recovered SSH Key Path: /mnt/tar-pit_fs/home/backup/.ssh/id_rsa

The presence of multiple valid credentials and an SSH key confirms the challenge's premise of "subtle operational mistakes." For a live exploitation scenario, the SSH key for the backup user would be the preferred initial foothold.

Phase III: Exploitation of Misconfigured Sudo Permissions

The final phase involved identifying and exploiting a local privilege escalation vulnerability to achieve root access.

3.1 Sudoers File Analysis

The most critical finding was located within the /etc/sudoers file, which defines user permissions for executing commands as other users. The entry for the backup user was found to be dangerously permissive:

```
User    Host    RunAs   NOPASSWD    Command
backup  ALL=(root)  NOPASSWD:   /usr/bin/tar
```

This configuration grants the backup user the ability to execute the /usr/bin/tar binary as the root user without requiring a password. This is a well-documented security risk, as the tar utility can be abused to read, write, or execute arbitrary files/commands as root.

3.2 Privilege Escalation Vector

The vulnerability is categorized under the **GTFOBins** class of misconfigurations, specifically the sudo entry for tar. The exploitation technique involves using tar's --checkpoint-action feature to execute a shell command during the archive process.

For a live system, the following command would be used by the backup user to read the contents of the /root/flag.txt file:

```
# Command to execute 'cat /root/flag.txt' as root
```

```
sudo /usr/bin/tar -cf /dev/null /root/flag.txt --checkpoint=1 --
checkpoint-action=exec=cat\ /root/flag.txt
```

In the context of the offline forensic analysis, the flag was simply read directly from the mounted filesystem:

Flag Path: /mnt/tar-pit_fs/root/flag.txt **Flag Content:** SECE{b2r_sudo_tar_pwned}

Conclusion and Remediation

The "Tar-Pit" challenge serves as an excellent demonstration of how a chain of seemingly minor operational security flaws can lead to a complete system compromise. The successful exploitation relied on the following vulnerability chain:

1. **Initial Foothold:** Plaintext credentials and an exposed SSH key due to poor file management.
2. **Privilege Escalation:** A critical misconfiguration in the /etc/sudoers file, granting the low-privilege backup user root-level execution rights over the tar binary.

Remediation Recommendations

To prevent similar compromises, the following security measures are strongly recommended:

3. **Sudoers Policy:** Immediately remove the NOPASSWD entry for /usr/bin/tar for all non-administrative users. If a user requires elevated privileges for a specific task, use a tightly scoped script wrapper instead of granting access to a powerful binary.

4 **Credential Management:** Implement a secrets management solution (e.g., HashiCorp Vault, environment variables) and enforce a policy against storing plaintext credentials in configuration files, especially within web roots or system directories.

5 **Key Management:** Review and revoke all unnecessary SSH keys. Private keys should never be stored in user home directories without strong passphrase protection.

6 **Filesystem Hygiene:** Implement automated scans to detect sensitive files (e.g., .old, .bak, .conf with credentials) in public-facing directories and ensure proper cleanup of development artifacts before deployment.

This forensic report confirms the successful identification and exploitation of the intended vulnerability, resulting in the retrieval of the final flag.