Comprehensive Security Demonstration on Fedora

Introduction

This document provides a detailed explanation of two security demonstrations on Fedora:

1. **SELinux Setup & Policy Enforcement**
2. **Simulated Attack & Defense**

Each demonstration includes installation, setup, execution, and mitigation procedures to help understand security principles in Fedora systems.

---

1. SELinux Setup & Policy Enforcement

Objective

Security-Enhanced Linux (SELinux) is a security architecture integrated into the Linux kernel that provides mandatory access control. The objective of this section is to configure SELinux, create secure directories, enforce access policies, and monitor security violations. Understanding SELinux is crucial as it adds an additional layer of security, preventing unauthorized access to system files and applications.

Steps Involved

1.1 Checking & Installing SELinux

SELinux is usually enabled by default in Fedora, but verifying its status is essential to ensure proper security configurations. To check if SELinux is installed and running, use:

```
sestatus
```

If SELinux is not installed, install the necessary packages using the following command:

```
sudo dnf install -y selinux-policy selinux-policy-targeted policycoreutils policycoreutils-python-utils
```

These packages include the core SELinux policy, management tools, and Python utilities for extended functionalities.

1.2 Enabling SELinux

By default, SELinux operates in one of three modes: enforcing, permissive, or disabled. The enforcing mode actively blocks unauthorized access, while permissive logs violations without enforcing restrictions. To enable enforcing mode:

```
sudo setenforce 1
```

Verify the mode using:

```
sestatus
```

This ensures that SELinux is actively protecting the system.

1.3 Creating a Secure Directory

To demonstrate SELinux's control over file access, we create a secure directory:

```
mkdir -p /secure_data
echo "Confidential Information" > /secure_data/secret.txt
```

By default, SELinux assigns a security context to new files. We explicitly set the SELinux context to `user_home_t` to simulate protection against unauthorized access:

```
sudo chcon -t user_home_t /secure_data
sudo chcon -t user_home_t /secure_data/secret.txt
```

This command changes the security attributes of the directory and file, restricting access to only authorized users.

1.4 Defining & Applying a Custom SELinux Policy

For additional control, custom SELinux policies can be created. Below is an example of defining a policy to restrict unauthorized access:

```
cat <<EOF > custom_policy.te
module custom_policy 1.0;
require {
    type user_home_t;
    class file { read write };
}
allow user_home_t self:file { read write };
EOF
```

This defines a policy allowing specific read/write access. The policy is then compiled and applied:

```
checkmodule -M -m -o custom_policy.mod custom_policy.te
semodule_package -o custom_policy.pp -m custom_policy.mod
sudo semodule -i custom_policy.pp
```

Applying this policy enforces strict access control based on SELinux's mandatory access control rules.

1.5 Monitoring Security Violations

One of SELinux's core benefits is logging unauthorized access attempts. These can be reviewed using:

```
sudo ausearch -m AVC,USER_AUTH | tee selinux_logs.txt
```

This command retrieves and saves audit logs, helping administrators analyze security breaches and modify policies accordingly.

2. Simulated Attack & Defense

Objective

Cybersecurity is a continuous battle between attackers and defenders. This section simulates a real-world cyber attack scenario, illustrating how a hacker might exploit system vulnerabilities. The simulation includes creating a fake attacker account, escalating privileges, deploying malware, and detecting the breach using system logs.

Steps Involved

2.1 Creating a Fake Attacker Account

Hackers often create hidden accounts to maintain unauthorized access. To simulate this:

```
sudo useradd -m hacker
echo "hacker:password123" | sudo chpasswd
```

This creates a user with a default password, similar to an initial reconnaissance phase where attackers gain a foothold in the system.

2.2 Privilege Escalation

Privilege escalation is a crucial step in an attack where an attacker attempts to gain root access. Here, we intentionally allow the `hacker` account to execute any command without requiring a password:

```
echo "hacker ALL=(ALL) NOPASSWD:/bin/bash" | sudo tee -a /etc/sudoers
```

This mirrors real-world privilege escalation tactics where attackers manipulate sudo permissions.

2.3 Running the Exploit

Once the attacker gains access, they execute commands to confirm control:

```
sudo -u hacker /bin/bash -c 'whoami && id'
```

This confirms whether the hacker has root access, a critical phase in an attack.

2.4 Deploying Malicious Process

Attackers often deploy malware to maintain persistence. Below, a simple script mimics an actual malware attack:

```
echo '#!/bin/bash' > /tmp/malware.sh
echo 'echo "🚨 Malware Executed! 🚨"' >> /tmp/malware.sh
chmod +x /tmp/malware.sh
nohup /tmp/malware.sh &
```

The `nohup` command ensures the malware runs in the background. Detecting this is critical to preventing a full-blown breach.

2.5 Detecting Unauthorized Access

To identify suspicious activity, administrators review system logs:

```
sudo journalctl -p err -n 20 | tee attack_logs.txt
```

Logs help in identifying unauthorized access attempts and potential privilege escalations.

2.6 Mitigation & Rollback

To mitigate the attack, we remove the hacker account and restore normal sudo configurations:

```
sudo userdel -r hacker
sudo sed -i '/hacker ALL/d' /etc/sudoers
sudo rm -f /tmp/malware.sh
```

This ensures that the system is secured and no backdoor access remains.

Conclusion

These demonstrations illustrate:

- The importance of **SELinux policies** in enforcing strict access controls.
- How **real-world attacks** operate and how security teams can detect them.
- How to **detect unauthorized access** and **mitigate threats** efficiently.

Cleanup Commands (If Needed)

To restore the system to its original state:

```
sudo userdel -r hacker
sudo semodule -r custom_policy
sudo rm -rf /secure_data custom_policy.*
sudo rm -f attack_fedora.sh selinux_fedora.sh
```

By following these security principles, Fedora users can strengthen their system against potential cyber threats.