Documentation: Finding the Backdoor in Passoire's Original Image

Overview

The task was to identify a backdoor in the `passoire` container image. This document details the systematic approach used to uncover the backdoor, focusing on the identification of a maliciously modified binary and its functionality.

Investigation Process

1. Initial Observations and Suspicious Behavior

- Upon accessing the `passoire` container, I noticed unusual system behavior, including:
  - Unexpected user accounts.
  - Potential tampering with standard system binaries.

---

2. Inspecting Critical System Binaries

- The `ls` binary was flagged as suspicious because:
  - It exhibited unusual behavior during directory listings.
  - Its size and hash did not match the standard `ls` binary.

Command:

```C/C++
sha256sum /usr/bin/ls
```

- Result: The hash of `/usr/bin/ls` did not match the hash of the legitimate `ls` binary from the official `coreutils` package.

---

3. Static Analysis of the Binary

- I used `strings` to analyze the content of the binary for embedded functionality:

Command:

```
C/C++
strings /usr/bin/ls
```

- Key Findings:
  - Presence of the word `backdoor`.
  - Commands such as `useradd` and `chpasswd`:

```
C/C++
useradd %s > /dev/null 2>&1
echo '%s:%s' | chpasswd > /dev/null 2>&1
```

  - Use of the `system` function to execute arbitrary shell commands.

---

4. Dynamic Analysis of the Binary

- I checked for dynamically linked libraries to ensure no unusual dependencies were loaded:

Command:

```
Unset
ldd /usr/bin/ls
```

- Result:
  - The binary depended only on standard libraries (`libc.so.6` and `ld-linux-x86-64.so.2`).
  - This suggested the malicious behavior was embedded directly in the binary.

---

5. Disassembling the Binary

- Using `objdump`, I disassembled the binary to analyze its functionality.

Command:

```
C/C++
objdump -d /usr/bin/ls
```

- Key Findings:
    - **Function `execute_command`:**
        - This function uses the `system()` call to execute arbitrary commands.
        - Example disassembly:

```
C/C++
11c0: e8 cb fe ff ff    callq  1090 <system@plt>
```

    - **Functionality in `main`:**
        - Constructs shell commands with `snprintf` and `strcat`.
        - Passes the constructed commands to `execute_command`.

---

6. Reconstructing the Malicious Behavior

- The `ls` binary was modified to:
    - Execute hidden commands to add unauthorized users.
    - Use `useradd` and `chpasswd` to create accounts and set passwords without logging outputs.

Exploit Scenario:

Running the binary under specific conditions (e.g., listing certain directories) triggers the execution of hidden commands embedded in the binary.

Steps to Mitigate the Backdoor

1. **Replace the Malicious Binary:**
    - Reinstalled `coreutils` to restore the legitimate `ls` binary:

```
C/C++
apt-get install --reinstall coreutils
```

2. **Verify User Accounts:**
    - Checked `/etc/passwd` for unauthorized accounts:

```
C/C++
cat /etc/passwd
```

3. **Inspect Logs:**
    - Reviewed logs to identify the addition of backdoor users:

```
C/C++
grep "useradd" /var/log/*
grep "chpasswd" /var/log/*
```

4. **Monitor for Persistence:**
    - Checked for scheduled tasks or startup scripts:

```
C/C++
crontab -l
find /etc -name '*cron*' -exec cat {} \;
```

Conclusion

By systematically analyzing the `passoire` container, I identified a malicious backdoor embedded in the `ls` binary. This backdoor leveraged shell commands to create unauthorized

users and gain persistent access to the system. The investigation highlights the importance of verifying system binaries and monitoring unusual behavior for potential tampering. Our group image was patched from this issue when we noticed unusual behavior in this program.