**PwnKit: Local Privilege Escalation Vulnerability Discovered in polkit’s pkexec (CVE-2021-4034)**

Linux policy toolkit

much similar to sudo but with GUI and more flexibility

main component is pkexec

Example

pkexec useradd test

cve-2021-4034 allows exploitation of pkexec to return shell

by providing zero list of arguments to pkexec

pwnkit exploit is local privilege escalation on Linux policy toolkit or polkit s pkexec.

**The Qualys Research Team has discovered a memory corruption vulnerability in polkit’s pkexec, a SUID-root program that is installed by**

**default on every major Linux distribution. This easily exploited vulnerability allows any unprivileged user to gain full root privileges on a vulnerable host by exploiting this vulnerability in its default configuration.**

**Polkit (formerly PolicyKit) is a component for controlling system-wide privileges in Unix-like operating systems. It provides an organized way for non-privileged processes to communicate with privileged processes. It is also possible to use polkit to execute commands with elevated privileges using the command pkexec followed by the command intended to be executed (with root permission).**

**Description**

**pkexec allows an authorized user to execute *PROGRAM* as another user. If *username* is not specified, then the program will be executed as the administrative super user, *root*.**

**What Are PolKit and pkexec?**

**Formerly known as PolicyKit, PolKit is a widely used component in Unix-like operating systems. It controls system-wide privileges and provides an organized way for non-privileged processes to communicate with privileged processes. PolKit has a command in its toolset called pkexec. pkexec command is a SUID-root program that allows users to run commands as another user such as root [3]. This command is in default configuration of many major Linux distributions such as Ubuntu, Debian, Fedora and CentOS.**

**What is the CVE-2021-4034  Vulnerability?**

**The CVE-2021-4034 vulnerability of pkexec is a memory corruption vulnerability. This vulnerability allows attackers to manipulate environment variables in Unix-like operating systems. This vulnerability existed since the inception of pkexec in May 2009.**

**How do Attackers Exploit Pwnkit CVE-2021-4034  Vulnerability?**

**Attackers use the CVE-2021-4034 vulnerability to add libraries of their choosing to environment variables such as PATH variable. Since pkexec command is mainly used by unprivileged users for executing commands as root, this exploitation allows attackers to gain elevated privileges in the target system.**

**Potential Impact of PwnKit Vulnerability**

**Successful exploitation of this vulnerability allows any unprivileged user to gain root privileges on the vulnerable host. Qualys security researchers have been able to independently verify the vulnerability, develop an exploit, and obtain full root privileges on default installations of Ubuntu, Debian, Fedora, and CentOS. Other Linux distributions are likely vulnerable and probably exploitable. This vulnerability has been hiding in plain sight for 12+ years and affects all versions of pkexec since its first version in May 2009 (commit c8c3d83,  “Add a pkexec(1) command”).**

**As soon as our Research Team confirmed the vulnerability, Qualys engaged in responsible vulnerability disclosure and has coordinated with both vendor and open-source distributions to announce the vulnerability.**

**The CVSSv3 base score for CVE-2021-4034 is 7.8 High**

[**Check if your system is vulnerable**](https://www.datadoghq.com/blog/pwnkit-vulnerability-overview-and-remediation/#check-if-your-system-is-vulnerable)

**If PolicyKit is not installed on your operating system, you are not affected by this vulnerability. Unfortunately, PolicyKit is installed by default in most major Linux distributions except Debian.**

**How to Protect Your Organization From PolKit Vulnerability Exploits?**

**To protect against exploitation of CVE-2021-4034 PwnKit vulnerability, we highly advise organizations to identify vulnerable systems on their networks and update them. Since pkexec was vulnerable from its conception, we can assume all Linux distributions that are using it in default configuration are vulnerable.**

**Seven Critical Steps to Defending Against PwnKit**

**Locate systems running with polkit**

**Polkit is supported by Linux and other Unix-like systems. We recommend using a cyber range to replicate your production environment and determine which machines are impacted.**

**Determine if polkit can be exploited**

**Not all systems that polkit supports are vulnerable. Qualys stated that OpenBSD is not exploitable, as it refuses to run (“execve()”) programs with zero command-line arguments (argc is 0), a key step in performing the exploit. However, they have successfully demonstrated exploitation of Ubuntu, Debian, Fedora, and CentOS systems. At this time, it is still worth testing the non-Linux operating systems that polkit supports, such as Solaris and BSD, as Qualys has yet to do so.**

**Prioritize assets for remediation**

**Once you know which assets in your environment are vulnerable, use t the information available about those assets to prioritize patching and mitigation efforts. Consider both the risk profile and business value of the assets.**

**Validate and apply patches for affected systems**

**The polkit team and vendors for some of the affected operating systems have already released patches:**

**Test and deploy mitigations for systems without available patches**

**Developers have not yet released patches for all affected Linux and Unix-like operating systems. However, you can still temporarily protect systems that do not have a patch or cannot be patched immediately by limiting the permissions of polkit (pkexec).**

**To do so, run the following command to remove the**[**SUID-bit**](https://www.redhat.com/sysadmin/suid-sgid-sticky-bit)**from pkexec:**

***# chmod 0755 /usr/bin/pkexec***

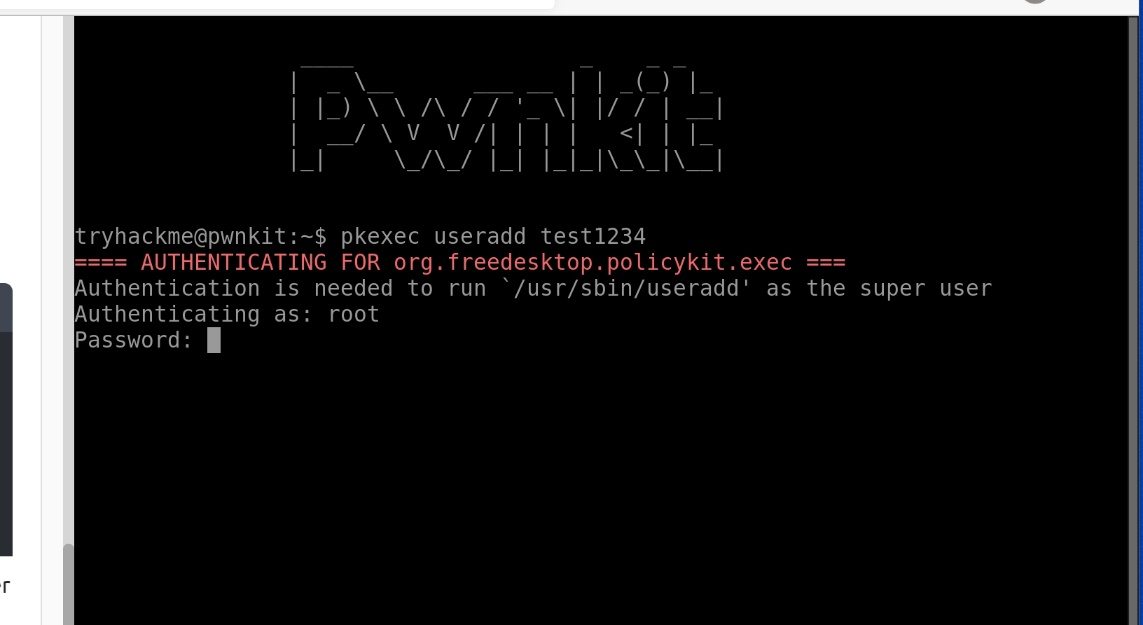
**Understand your defense in depth around polkit**

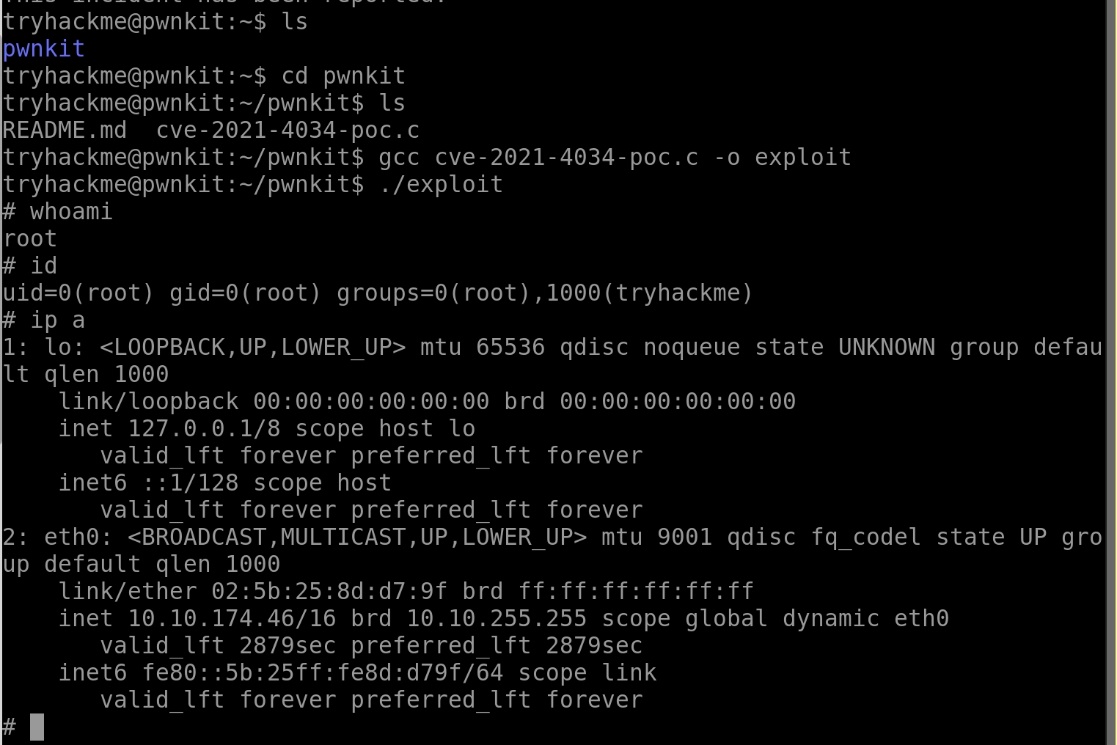
**With new vulnerabilities coming out regularly, even the best patch management response is only one part of a good defense in depth strategy to prepare for such risks. It is also important to have layers of controls that can prevent, detect or remediate exploitation. Since a user account on a vulnerable machine is required to exploit polkit, appropriate access controls and a zero-trust stack are important measures for limiting an attacker’s ability to gain root access through the PwnKit vulnerability.**

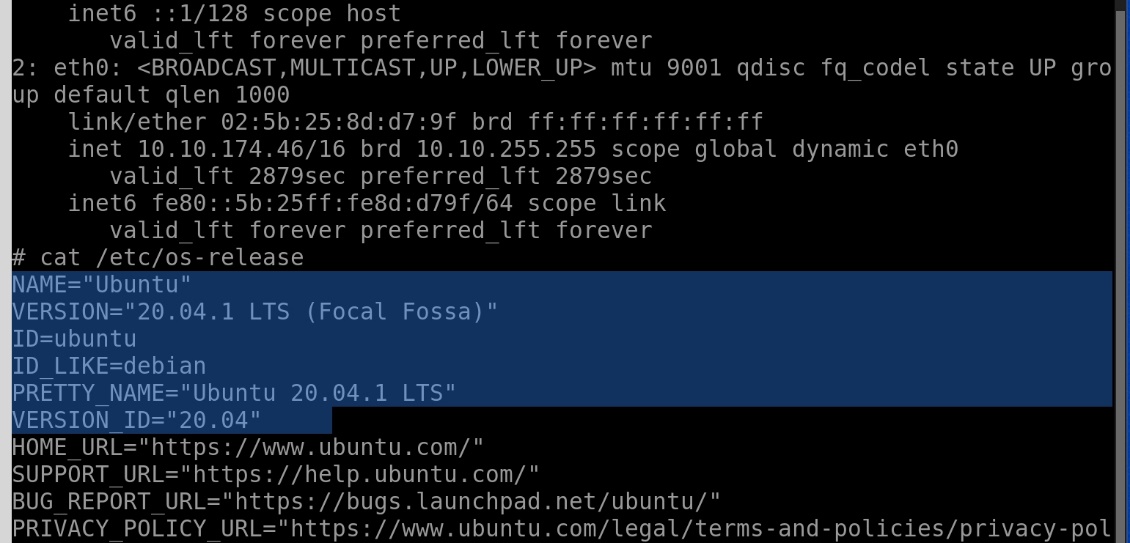
**Using a cyber range, you can safely run the PwnKit exploit within an accurate model of your environment and see what additional attack steps would be required to access vulnerable machines, test the speed and clarity of your detection suite, and understand your business risk. You can easily use open-source proofs of concept for the PwnKit exploit, such as those by**[**Davide Berardi**](https://github.com/berdav/CVE-2021-4034)**and**[**Andris Raugulis**](https://github.com/arthepsy/CVE-2021-4034)**; SimSpace is using these two in our own research and response efforts to this vulnerability.**

**Train your team on the ins and outs of this vulnerability**

**learn more about this vulnerability, how it can be exploited, and how to protect yourself.**

****

****

****

**Becoming Root Through An SUID Executable**

**Privilege escalation is a way that attackers can escalate their privileges on a system. For example, let’s say that an attacker has gained access to your web server, but only as a low privileged user. They cannot read or write sensitive files, execute scripts, or change system configuration. How could they compromise your server and maintain their access there?**

**If attackers can find a way to trick the system into thinking that they are the root user, they can carry out more powerful attacks like reading and writing sensitive files and inserting permanent backdoors into the system. And this is where privilege escalation comes in. Today, let’s talk about how attackers can exploit SUID programs to escalate their privileges to become root.**

**The SUID Bit**

**SUID stands for “SetUID”. It is a Linux permissions flag that allows users to run that particular executable as the executable's owner. For example, if a file is owned by root, the program will always run as root, regardless of who started the execution.**

**Why would this functionality be useful? A common use-case for SetUID is the password change utility. To change your own password, you would have to modify sensitive system files, such as /etc/shadow. This file is normally only accessible by root users, so you will need to have root privileges to carry out a password change. But since the system doesn't want to give a normal user root privileges, SUID allows users to obtain root privileges only when running certain programs. In this case, SUID on the password change utility allows users to gain temporary root access to change their passwords without obtaining root access across the board. For the most part, this is a normal and necessary behavior. But if an attacker can find a way to execute arbitrary code when running these SUID programs, they can exploit the temporary root access to execute code as the root user on the system!**

**For example, let’s look at the “Vim” file editor first. Let’s say that Vim is owned by the root user and has the SUID bit set on a system. This means that whenever a user runs the Vim editor, Vim is running with root privileges. No biggie, right?**

**This setting could actually spell the death sentence for your system because you can actually run arbitrary system commands from within the vim editor! To run commands in Vim, you have to type the characters colon bang “:!”, then the command you want to run. For example, to run the “ls” command, you can type “:!ls” then press enter. You should now see the results of the ls command on your terminal.**

**So this means that if the Vim executable has the SUID bit set, the attacker can execute system commands as root from the Vim editor! Other editors, such as “more” and “less” also allows for command execution from within the program. You can type the bang character “!” then the command that you want to execute for these programs. For example, to run the “ls” command, you can type “!ls” then press enter. You should now see the results of the ls command on your terminal.**

**Besides file editors, another program that allows users to run arbitrary system commands is the “find” command. The find command is usually used for locating files and often has the SUID bit set to allow users to find files across the system. But find allows the execution of system commands through the “-exec” flag! For example, to run the “ls” command from within the find command, you can use the command “find . -exec ls \;”. So if the find executable has the SUID bit set, the attacker can execute system commands as root!**

**Escalating Privileges Using The Vulnerability**

**These misconfigurations make privilege escalation trivial. For example, an attacker can use the ability to execute commands as root and add themselves as a root user in the /etc/passwd file. This command will do just that.**

**echo “vickie::0:0:System Administrator:/root/root:/bin/bash” >> /etc/passwd**

**This command adds a root user with the username of “vickie” and an empty password. Since “0” is the UID of the root user, adding a user with the UID of “0” will give that user root privileges. This command is not possible for regular users because only privileged users can modify system-critical files such as the /etc/password file.**

**More SUID Dangers**

**Programs that lead to privilege escalation when run with SUID are not just limited to programs that allow for arbitrary system code execution. Any programs that allow arbitrary writes to system files are owned by root and have the SUID bit set can lead to privilege escalation.**

**For example, if the file editor “Nano” has the SUID bit set, the attacker can use Nano’s root permissions to open the “/etc/passwd” file and add themselves as the root user directly in the file editor!**

**And the system utility “cp” is used to copy and overwrite files. If it has the SUID bit set, attackers can tamper with any file on the system by overwriting the original file with its root privileges! For example, the attacker can create a copy of the original /etc/password to a file they own. Then, they can add themselves as a root user by editing the copy of the passwd file. Finally, they use the “cp” command to overwrite the original/etc/password file with the modified one.**

**Be Careful!**

**You can see that SUID could become incredibly dangerous when misused. SUID rights should only be granted to programs when necessary and not to programs that allow command execution or arbitrary writes to files on the system.**