

CTF 1
CONDUCTED BY TEAM222

CTF 1 - Report

47716088
Mohnish SHARMA

April 21, 2024

Contents

1	Report Overview	1
1.1	Executive Summary	1
1.2	Scope of Testing	1
1.3	Ethical Disclosure	2
1.4	Tools Utilized	2
2	Flags	3
2.1	Flag 1 - "Yellow Pages"	3
2.1.1	An Overview	3
2.1.2	Our Solution	3
2.1.3	Risk Implications	4
2.2	Flag 2 - "Smoke Signals"	4
2.2.1	An Overview	4
2.2.2	Our Attempts	4
2.3	Flag 3 - "Attribution"	5
2.3.1	An Overview	5
2.3.2	Our Solution	5
2.3.3	Risk Implications	6
2.4	Flag 4 - "Call of Cthulhu"	7
2.4.1	An Overview	7
2.4.2	Our Solution	7
2.4.3	Risk Implications	8
2.5	Flag 5 - "Othello"	8
2.5.1	Our Attempts	8
3	Conclusion	8
3.1	Personal Reflection	8
3.2	Overall Assessment of Risk	9
4	Appendix	11
	References	11

1 Report Overview

1.1 Executive Summary

The CTF-1 was a group based, practical "capture the flag" style exam, consisting of 5 challenges (categorised by difficulty into 3 sections), and was accessible via a web-based kali desktop instance available from 9:00AM 8th April, until 11:45PM Friday 12th April.

Our team, **Team222** was comprised of 4 members, Mohnish Sharma (myself), Tam Tin (Ambrose) Tang, Micheal Zambaka and Santosh Aryal. **Team222** was able to successfully complete 3 challenges, Yellow pages, Attribution & Call of Cthulhu, leaving us with 3 out of 5 flags.

Challenge Name	Team Members Involved	Discovered Flags
Yellow Pages	Mohnish & Santosh	flag{5e78f443800197464cea1f0a53be0be9}
Smoke Signals	Mohnish, Micheal & Ambrose	No flag discovered
Attribution	Mohnish & Ambrose	flag{bddd8e4540a7df82ce7a0914ab19b13d}
Call of Cthulhu	Mohnish, Micheal & Ambrose	flag{d48ccbe365e2e47ed60f3712ceafb492}
Othello	Mohnish	No flag discovered

Table 1. List of challenges, team members involved, and any discovered flags.

For the purpose of this report, the solutions have been refined to get to the flag in the least number of steps in order to improve readability and replicability. For all the steps and commands that were executed on the CTF servers, the team logbook contains these specific details.

Regarding the risk analysis portions in this report, the "flags" will be treated as if they were sensitive information contained within the system, which was penetration tested by our team.

1.2 Scope of Testing

Each CTF level was a recent Ubuntu distribution accessed remotely through the `telnet` protocol, which provided a command line interface allowing us to communicate with the remote server[1]. The following hosts were contacted to access each level.

- Yellow Pages: 10.222.1.10
- Smoke Signals: 10.222.1.20
- Attribution: 10.222.1.30
- Call of Cthulhu: 10.222.1.40
- Othello: 10.222.1.50

Each penetration test on each level was conducted with a high degree of care and attention to detail, the limitation of the guest (& non-root) user further ensured none of the original system was damaged or services affected. Any major changes within the system are highlighted within this report (such as Micheal changing the guest user password by mistake), and all the steps taken and command's executed were recorded in a logbook format that can be reviewed and used to reproduce the methods used to infiltrate the

system and capture the flags, as well as serve as a guideline for reversing any unforeseen adverse effects that arise as a result of the CTF operation.

1.3 Ethical Disclosure

This CTF activity was carried out in a entirely safe and authorized setting intended for learning and academic assesment purposes only. No systems, networks, devices, or data was accessed/tampered with without authorization, and a controlled environment was established through the use of the client-less remote desktop tool, [Apache Guacamole](#), ensuring safety and the integrity of our personal devices and data.

Strictly only **Ethical Hacking** techniques abiding to the specified parameters were used inside the CTF to locate and retrieve the 32 character flag strings. It is vital to bear in mind that trying to exploit security holes or access systems without proper approval can constitute illegal activity, and can have serious consequences. The methods detailed in this report should only be practiced in authorized environments and must not be reproduced with malicious intent.

1.4 Tools Utilized

Various tools were utilized throughout the CTF to assist with the operation of the CTF as a whole and the 5 levels within:

- [Apache Guacamole](#)
 - Mentioned earlier, this client-less remote desktop tool was used to access a virtual kali-box to isolate our personal computers for safety and compatability reasons.
- [telnet](#)
 - This command was used in the kali-box to connect to each individual CTF level. It had to be installed via `$ sudo apt install telnet` on the virtual kali-box as it had a fresh install of Kali Linux.
- [base64.guru/convertor/decode/file](#)
 - Ambrose referred me to this website for easy base64 to file conversions. It was of help in Flag 3 where I needed to convert the base64 string into a png, as putting the decoded metadata in a empty file kept resulting in a corrupt image.
- **Phone Camera**
 - A phone camera was used to scan the QR code in the image generated after decoding the base64.
- [CVSS Calculator](#)
 - FIRST, the company selected by NIAC to be in charge of CVSS development has a calculator for calculating the threat level of any computer system security vulnerability.

2 Flags

2.1 Flag 1 - "Yellow Pages"

2.1.1 An Overview

Provided Hint: Based!

This was the first challenge in the CTF and first of two in the "Daylight" (easy difficulty) section.

Through mine and Santosh's research, we speculated that the name "Yellow Pages" hinted to the former name for the Network Information Service (or NIS), further supported by the hint of "**Based!**" as the name was later claimed by British Telecom PLC for their paper-based, business phone catalog[2]. And is also why most of the commands and directory names were prefixed by "yp"[3].

Initially, I decided to check the `/var/yp/` directory to see if anything relating to the flag could be residing there, but the directory did not exist and it was clear now that the domain was not configured as an NIS server and I decided it must be simpler than that, such as relating to one of the basic NIS elements[2]:

- Domains
- Guides
- Daemons
- Utilities
- NIS Command Sets

And which the flag for this challenge was, being closely related to the domain element the flag was discovered to be in the domain name for the server as a string encoded in base32.

2.1.2 Our Solution

Exploring the clue that it may be related to basic NIS elements I decided to connect to the ctf server via:

```
$ telnet 10.222.1.10
```

and executed:

```
$ ypserv
```

which yielded an output of:

```
> MZWGCZ33GVSTOODGGQ2DGOBQGAYTSNZUGY2GGZLBGFTDAYJVGNRGKMDCMU4X2===
```

This was identified as a base32 encoded string, decoded via:

```
$ echo "MZWGCZ33GVSTOODGGQ2DGOBQGAYTSNZUGY2GGZLBGFTDAYJVGNRGKMDCMU4X2===" | base32 -d
```

yielding the output (and flag) of:

```
> flag {5e78f443800197464cea1f0a53be0be9}
```

2.1.3 Risk Implications

Assuming the flags as private information, this would prove to be a very, very poor location to store any remotely sensitive information, as a `domain name` is publicly accessible in all scenarios, whether it be a website or a remote server in this case. The encoding of the sensitive information (flag) in `base32` in effort that an attacker would not be able to read it also proves to be a poor implementation of security principles, as text encoding is far different to encryption due to the fact that, unlike encryption where a key or a set of keys are required for decryption, decoding `base32` is almost effortless as the conversion algorithms are public and well known. Therefore, any sensitive information would almost definitely be compromised in the event of an attacker gaining guest access to the system.

2.2 Flag 2 - "Smoke Signals"

2.2.1 An Overview

Provided Hint: Don't forget your pipe cleaner!

This was the second challenge in the CTF and the last of two in the "Daylight" (easy difficulty) section.

Our team was unsuccessful in capturing this flag, but I do believe we were close. The name led us to believe the flag was closely related to Linux signals, but the hint clearly mentioned it was related to pipes. Since pipes are objects that allow two processes to communicate[4], and signals can be sent from the kernel to a process, from a process to another process, or from a process back to itself[5].

2.2.2 Our Attempts

I started by running:

```
$ ls -l proc/*/fd/* | grep pipe
```

To look for any pipes that might've been interacting with a process, and sure enough there was a common pipe:

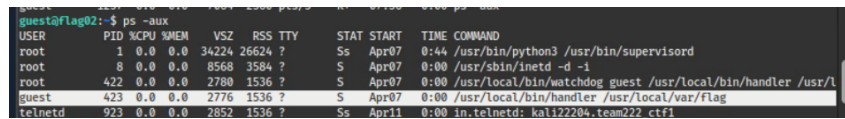
```
ls: cannot access '/proc/1141/fd/255': No such file or directory
ls: cannot access '/proc/1141/fd/3': No such file or directory
ls: cannot read symbolic link '/proc/423/fd/0': Permission denied
ls: cannot read symbolic link '/proc/423/fd/1': Permission denied
ls: cannot read symbolic link '/proc/423/fd/2': Permission denied
ls: cannot access '/proc/self/fd/255': No such file or directory
ls: cannot access '/proc/self/fd/3': No such file or directory
ls: cannot access '/proc/thread-self/fd/255': No such file or directory
ls: cannot access '/proc/thread-self/fd/3': No such file or directory
l-wx----- 1 guest guest 64 Apr 12 05:41 /proc/1141/fd/1 -> pipe:[2387447862]
lr-x----- 1 guest guest 64 Apr 12 05:41 /proc/1142/fd/0 -> pipe:[2387447862]
l-wx----- 1 guest guest 64 Apr 12 05:41 /proc/self/fd/1 -> pipe:[2387447862]
l-wx----- 1 guest guest 64 Apr 12 05:41 /proc/thread-self/fd/1 -> pipe:[2387447862]
```

Figure 1. The output after running `$ ls -l proc/*/fd/* | grep pipe`.

But I was quite unsure what to do with this information, so I disregarded it for now and started brainstorming with Ambrose and Micheal, they suggested looking for a process directly instead of the pipe. So I executed the command:

```
$ ps aux
```

To view any processes currently running, and sure enough there was a process that mentioned "flag".



USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	34224	26624	?	Ss	Apr07	0:44	/usr/bin/python3 /usr/bin/supervisord
root	8	0.0	0.0	8568	3584	?	S	Apr07	0:00	/usr/sbin/inetd -d -l
root	422	0.0	0.0	2780	1536	?	S	Apr07	0:00	/usr/local/bin/watchdog guest /usr/local/bin/handler /usr/l
guest	423	0.0	0.0	2776	1536	?	S	Apr07	0:00	/usr/local/bin/handler /usr/local/var/flag
telnetd	923	0.0	0.0	2852	1536	?	Ss	Apr11	0:00	in.telnetd: kali22204.team222.ctf1

Figure 2. The output after running `$ ps aux`.

Navigating to the directory with 'file', and executing:

```
$ file flag
```

revealed the file type to be a 'fifo (named pipe)', which brought my attention back to the hint and I tried flushing the pipe by killing the process currently using it to see if it would change anything:

```
$ kill 423(423isthePIDoftheprocess)
```

But all it did was create a new **watchdog** process in the currently running processes (visible through `ps aux`), and is where we seemed to run out of ideas on what to try next.

2.3 Flag 3 - "Attribution"

2.3.1 An Overview

Provided Hint: That's so Meta!

This was the third challenge in the CTF and the first of two in the "Moonlight" (medium difficulty) section.

As the name "Attribution" suggested, this flag was related to the hidden attributes that Linux and most common file systems use to regulate the level of interaction that any given system processes can have with files or directories[6].

This flag, while still related to the hidden attributes, was more precisely located within the **extended attributes** of the `flag.txt` file, as a user extended attribute class.

2.3.2 Our Solution

After connecting the CTF server via:

```
$ telnet 10.222.1.30
```

I was initially under the impression the flag was related to the basic hidden attributes of a certain file in the server, I executed:

```
$ lsattr -R | grep "flag*"
```

To see if there was a file related to the flag with differing attribute options[7]. In hindsight, this may have been an unnecessary action as there is no way the 32 character long flag could have been contained in the predefined attribute options, although it did also perform a recursive search of the file structure and revealed the location of the `flag.txt` file in:

```
./home/david/flag.txt
```

I got a list of the extended attributes through:

```
$ getfattr flag.txt
```

yielding all the extended attributes on the file, which was just:

```
> user.flag
```

I then executed:

```
$ getfattr -d -only-values flag.txt
```

to view the contents of the extended attribute, and used the `-only-values` option to dump the raw attribute values without encoding. [8]

The contents of the extended attribute was a very long base64 encoded string, I have placed it in it in the [appendix](#) to reduce clutter. You can view the appendix by scrolling down or clicking [here](#).

Ambrose then suggested I used an online base64 to file convertor so I used this website to convert the base64 into a png.

base64.guru/convertor/decode/file



Figure 3. The resultant PNG after decoding.

Scanning this QR code with a mobile phone then returned the flag of:

```
flag{bddd8e4540a7df82ce7a0914ab19b13d}
```

2.3.3 Risk Implications

Risk Implications that arose with this flag were certainly not as severe as those in earlier levels such as 'Yellow Pages' but nonetheless pose a huge security risk to the system in the case that sensitive information was being stored as the flag. Imagine the `user.flag` property to be a patient's medical records at a hospital, encoding a QR code in base64 certainly would work as a deterrent to any malicious user, but is still accessible fairly easily and to a guest remote user at that. A much better approach would be modifying the permissions of the file so that the guest user cannot read it, as well as removing the guest user's ability to modify permissions, or just removing the guest user altogether.

2.4 Flag 4 - "Call of Cthulhu"

2.4.1 An Overview

Provided Hint: What process should you follow to insure he doesn't rise from his environment?

This was the fourth challenge in the CTF and the last of two in the "Moonlight" (medium difficulty) section.

The solution for this lied in the given hint, as it clued toward the flag being involve a process running in Linux, which refers to the active execution of a program[9], and are kept in the `/proc/` directory, which represents the current state of the kernel, and allows for both users and applications to peer into the kernel's view of the whole system[10].

Knowing this, the flag must lie in this folder and be a currently running application. As the hint suggested, Ambrose "followed" the sleep process (PID: 10) to its "environment" by using the `cat` command to print the filepath of the `PATH` element in this directory:

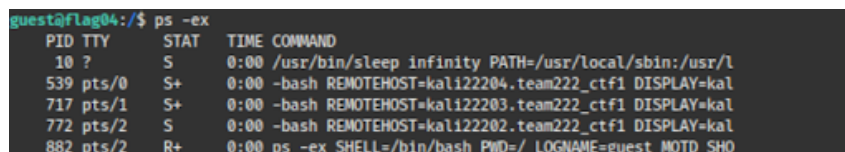
```
/proc/10/environ
```

The output contained the flag in the middle of the string.

2.4.2 Our Solution

I ran the following command to view all the current processes running: I used the `'-e'` option to list 'every' process running and `'-x'` to include those not attached to the current terminal session.

```
$ ps -ex
```



```
guest@flag04:/$ ps -ex
  PID TTY          STAT       TIME COMMAND
    10 ?            S          0:00 /usr/bin/sleep infinity PATH=/usr/local/sbin:/usr/l
   539 pts/0      S+         0:00 -bash REMOTEHOST=kali22204.team222_ctf1 DISPLAY=kali
   717 pts/1      S+         0:00 -bash REMOTEHOST=kali22203.team222_ctf1 DISPLAY=kali
   772 pts/2      S          0:00 -bash REMOTEHOST=kali22202.team222_ctf1 DISPLAY=kali
   882 pts/2      R+         0:00 ps -ex SHELL=/bin/bash PWD=/ LOGNAME=guest MOTD_SHO
```

Figure 4. Output from `$ ps -ex`.

Micheal and Ambrose then pointed out the "?" process and decided to investigate further by trying to read the process' environment via:

```
$ cat /proc/10/environ
```

Which then returned this string with the flag:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binHOSTNAME=flag04flag=
d48ccbe365e2e47ed60f3712ceafb492team=Team222ctf=01index=10DEBIAN_FRONTEND=
noninteractiveHOME=/rootLC_CTYPE=C.UTF-8SUPERVISOR_ENABLED=1SUPERVISOR
_SERVER_URL=unix:///var/run/supervisor.sockSUPERVISOR_PROCESS_NAME=sleep
SUPERVISOR_GROUP_NAME=sleep
```

2.4.3 Risk Implications

This level is a more likely scenario than say having sensitive info in the hostname, and could be constituted for a doctor browsing a patients medical records using an application to do so, logging out without closing it, which leaves the guest user able to search the ongoing processes and extract said sensitive info. This would likely stem from a lack of security and effort put in creation of the application, likely created as a custom-made for the hospital which leaves it vulnerable to other users that are monitoring running processes.

2.5 Flag 5 - "Othello"

Provided Hint: Talk to you later! Reset your expectations

This was the fifth challenge in the CTF and the one one in the "Midnight" (hard difficulty) section.

2.5.1 Our Attempts

We did not get very far on this level, as every time I tried to use the 'guest' user to logon (the standard for the previous levels). I was met with a ASCII-art message telling me "ACCESS DENIED", I tried to go through telnet manuals, but could not find anything that would allow me to easily bypass the login[11]. I did some further research and discovered telnet is inherently insecure as passwords were transmitted through plaintext[12], but without any way to intercept someone interting the correct login details, I was stuck.

3 Conclusion

In the end our team, **Team222** ended the CTF with 30 points (10 points per flag), and 3 out of the 5 flags. I do think we could have done better especially being so close to getting Flag 2, but it served as a great learning experince and believe all members of the team benefited from participating.

3.1 Personal Reflection

Engaging in this CTF activity has been an eventful experience that explored the bounds of my technical abilities and forced me to engage in extensive research, expanding my current knowledge of Linux systems and ethical hacking. I found the vague instructions and hints very confusing at first, but learnt to eventually adapt to the unfamiliar structure and started to think outside of a purely technical level, instead analysing the names of the levels and the short hints given to relate it to different parts of a Linux system, such as processes, extended attributes, or signals.

3.2 Overall Assessment of Risk

As was mentioned in the executive summary and the captured flags, for the purposes of this report the flags were considered to be sensitive information, i.e. a patients medical records on a hospital computer.

The Common Vulnerability Scoring System (CVSS) is an industry standard way of assessing the threat level of any computer system security vulnerabilities[13]. The Forum of Incident Response and Security Teams (FIRST) has a calculator developed for this, where you can calculate a base score off certain factors:

- Attack Vector (AV)
 - 'Network (N)' has been selected for this as we remotely connected to each level and therefore an attackers path, should they attack any of these levels would also be through OSI layer 3, otherwise known as the network layer[13].
- Attack Complexity (AC)
 - 'Low (L)' has been selected as even though our team did have difficulties and also were not able to capture 2 of 5 the flags, this CTF had a limited time frame where in a real world scenario where a vulnerability has gone undetected the attackers would likely have more time to exploit, and most likely more expertise than 2nd year cybersecurity students. If we could crack it, they could too.
- Privileges Required (PR)
 - None (N) has been selected as all the vulnerabilities found were conducted through a guest user.
- User Interaction (UI)
 - None (N) has been selected as the vulnerable system can be exploited without any interaction from the user.
- Scope (S)
 - Unchanged (U) has been selected as the only resources affected were those specific aspects in the level, no other components were interfered with nor did any of the vulnerabilities have a way to escalate privileges and consequently the vulnerability into other components of the system.
- Confidentiality (C)
 - Low (L) has been selected as only access to some restricted information is obtained, but it is very specific and the attacker does not have control on which information is gained as there is no way to escalate the vulnerability.
- Integrity (I)
 - None (N) has been selected as there is zero loss of integrity within the impacted component.
- Availability (A)
 - None (N) has been selected as there was no reduced performance or interruptions caused by any of the vulnerabilities.

Base Score

Attack Vector (AV)
 Network (N) | Adjacent (A) | Local (L) | Physical (P)

Attack Complexity (AC)
 Low (L) | High (H)

Privileges Required (PR)
 None (N) | Low (L) | High (H)

User Interaction (UI)
 None (N) | Required (R)

Scope (S)
 Unchanged (U) | Changed (C)

Confidentiality (C)
 None (N) | Low (L) | High (H)

Integrity (I)
 None (N) | Low (L) | High (H)

Availability (A)
 None (N) | Low (L) | High (H)

5.3 (Medium)

Figure 5. The resulting CVSS score from the calculator.

The resultant CVSS score from the calculator is 5.3. Which I believe to be accurate as the vulnerabilities encountered and exploited within the CTF only led to very specific data breaches and not any privilege escalation or any integrity destroying exploits. The main reason for the score not being even lower would have to do with the simplicity of the attacks as they are due to bad implementation of the system, although this means they should be patched relatively easily and quickly.

4 Appendix

Exhibit 1. Base64 String returned in Flag 3 - Attribution

```
iVBORw0KGgoAAAANSUheUgAAAG8AAABvAQMAAADYCWwjAAAABlBMVEUAAAD///+l2Z/dAAAAAnRS TIP//8i138cAAAAJcEhZcwAACxIAAAASAdLdfvwAAAEISURBVDiN1dS9rcQgDABgRxR0lwWQWIOO lZIFLskCyUruWCPSLQAdBTo/J/enazDF05OeFUX5CgvbBIC+Av4HI8AUbEIDYEUmKmMoHZmJGojn Ry6DbuKgAXQrr36fsmkilcEZ8OVdZIXc7xgMP+/2K+S4Q+nzZ7AVRjDgaPZqdQ30txQUIdmRabM Y9/7QFGTSAq0keKKZi/zHLu5gFpBZnRqo3M5tCIJOdEugeZHVXWSXbD0qGbXwn3M3OyNghWZUK0a rs50uYFUetq7wMMnmVkl4pbV/blHVSLwqQEN3SO3So50vGx0VuTxx2a7oRlfh71CbuHqypSfuQl5 C2n9FClw8Mda6ytXoN67fOyUTL5zeJKoopP J/Q7eXDwtQeYv3rF/xB+D4DxaO8Q/qgAAAABJRu5E rkJggg==
```

References

- [1] ExtraHop. “Telnet Protocol: Definition How It Works | Protocol Support Library | ExtraHop”. Extrahop.com, 2019. URL: <https://www.extrahop.com/resources/protocols/telnet/> (page 1).
- [2] “Linux - Network Information Service”. GeeksforGeeks, Nov. 2020. URL: <https://www.geeksforgeeks.org/linux-network-information-service/> (page 3).
- [3] Jamie Cameron. “NIS Client and Server”. Webmin, Sept. 2023. URL: <https://webmin.com/docs/modules/nis-client-and-server/#intro> (visited on 04/20/2024) (page 3).
- [4] “Working with pipes”. www.ibm.com. URL: <https://www.ibm.com/docs/de/aix/7.2?topic=io-working-pipes> (visited on 04/21/2024) (page 4).
- [5] “Linux Processes and Signals- 2020”. www.bogotobogo.com. URL: https://www.bogotobogo.com/Linux/linux_process_and_signals.php (page 4).
- [6] “File permissions and attributes - ArchWiki”. wiki.archlinux.org. URL: https://wiki.archlinux.org/title/File_permissions_and_attributes (page 5).
- [7] “chattr(1) - Linux manual page”. man7.org. URL: <https://man7.org/linux/man-pages/man1/chattr.1.html> (page 6).
- [8] “getfattr(1) - Linux manual page”. man7.org. URL: <https://man7.org/linux/man-pages/man1/getfattr.1.html> (visited on 04/21/2024) (page 6).
- [9] Shivam Verma. “What is a Process in Linux/Unix?” Scaler Topics, Sept. 2023. URL: <https://www.scaler.com/topics/linux-process/> (visited on 04/21/2024) (page 7).
- [10] “Appendix E. The proc File System Red Hat Enterprise Linux 6 | Red Hat Customer Portal”. access.redhat.com. URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/ch-proc (visited on 04/21/2024) (page 7).
- [11] “telnet(1): user interface to TELNET protocol - Linux man page”. linux.die.net. URL: <https://linux.die.net/man/1/telnet> (page 8).

- [12] Missouri University of Science and Technology. “Secure Telnet Connection”. Missouri ST. URL: <https://it.mst.edu/policies/secure-telnet/> (page 8).
- [13] “Common Vulnerability Scoring System Version 3.0 Calculator”. FIRST — Forum of Incident Response and Security Teams. URL: <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N> (visited on 04/21/2024) (page 9).