picoCTF & dichotomize

Anish G. & Yubo C.

CS Club Cybsersecurity

September 22, 2022

CS Club Discord

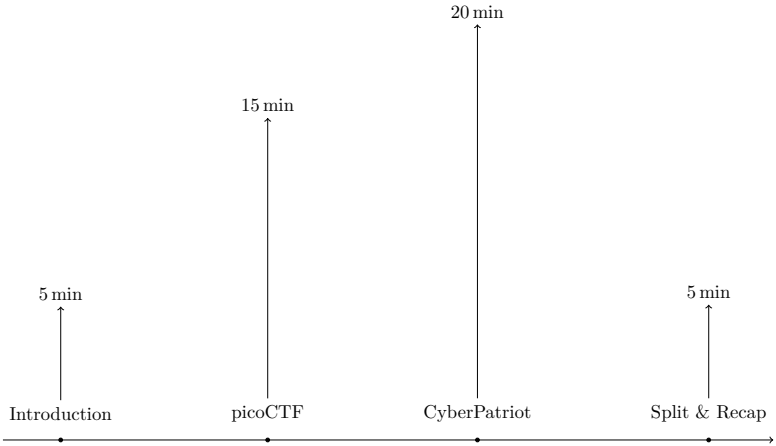
Powered by LaTeX 2_ε

Welcome back!
We are glad that you are here.
Re-introduce your self to the surrounding people as necessary.

Table of Contents

Agenda & Goal

Agenda


5 min
15 min
20 min
5 min
Introduction
picoCTF
CyberPatriot
Split & Recap

Goal

Introduction and general idea of picoCTF.

Solid foundation on CyberPatriot.

Dichotomize and choose picoCTF, CyberPatriot, or both.

Timeline

Timeline

picoCTF

picoCTF Timeline

2023/02/01 Registration opens

2023/03/14 CTF opens

2023/03/28 CTF closes

picoCTF Timeline

In essence, picoCTF won't happen until 2023. This activity/competition will happen in the next semester.

Timeline

CyberPatriot

CyberPatriot Timeline

2022/10/14–16 & 2022/10/20–22 Round 1. All teams may compete.
Because of fall break, the team may choose any one of those windows to compete.

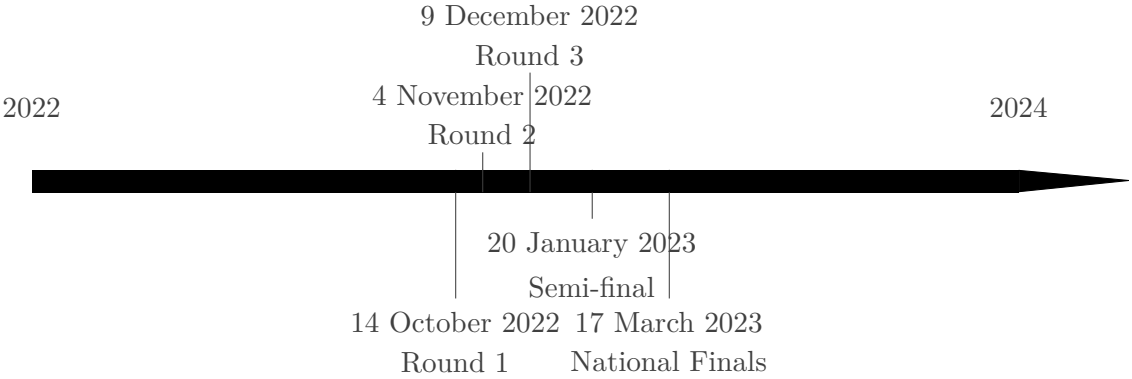
2022/11/4–6 Round 2. All teams may compete.

2022/12/9–11 Round 3. All teams may compete.
The results from rounds 1 & 2 will not count.

2023/1/20–21 Semi-final

2023/3/17-21 National Finals

CyberPatriot Timeline



CyberPatriot Application

Written application and interest form closes 10/1, 11:59 PM. There will be practice images for you, thence, no prior experience is required.

As we are running short on the team member slots, apply for the CyberPatriot competition here. We promise it won't be stressful, despite we can't select all of you all.

Figure: GSMST CyberPatriot Application: `https://tinyurl.com/GSMSTcp2022`

picoCTF

picoCTF

Introduction

picoCTF Introduction

picoCTF is a hacking competition for high school students. It is sponsored by Carnegie Mellon University. Participants learn to overcome sets of challenges from six domains of cybersecurity including general skills, cryptography, web exploitation, forensics, etc. The challenges are all set up with the intent of being hacked, making it an excellent, legal way to get hands-on experience.

picoCTF

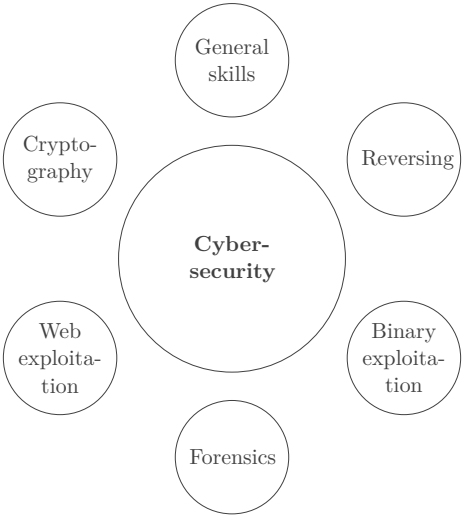


Figure: 6 Components in picoCTF

picoCTF website

Here is the url of this website `https://play.picoctf.org/login`. You should register an account for participating in the problem solving. After registration, you should be able to go to the `https://play.picoctf.org/practice` and practice some problems.

picoCTF

Question 1

Sample Question 1

Mod 26

| 10 points

Tags: picoCTF 2021 Cryptography

AUTHOR: PANDU

Description

Cryptography can be easy, do you know what ROT13 is?

cvpbPGS{arkg_gvzr_V'yy_gel_2_ebhaqf_bs_ebg13_jdJBFOXJ}

Hints

1

Caesar Cipher

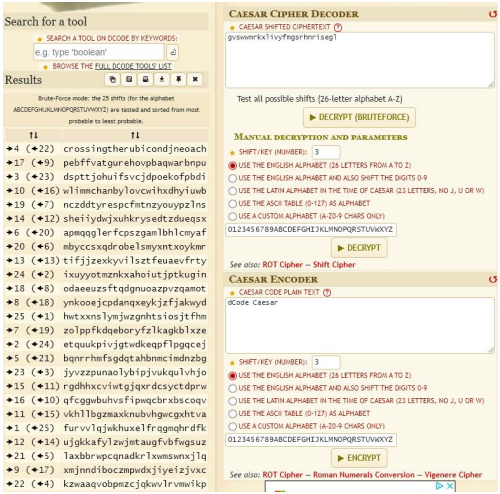
A caesar cipher is a substitution cipher where each letter in the alphabets is shifted several characters after, in alphabetic order. For example, `HELLO` would shifted to `IFMMP`, by shifting every letter one position after. Knowing this rule, this problem essentially asks to shift every letter by 13. To quickly finish this challenge, consider this code:

Solution to Question 1

```
1 "".join(result if (result :=
↪   ascii_lowercase[(ascii_lowercase.find(char.lower()) -
↪   ascii_lowercase.find("a") + 13) % len(ascii_lowercase)])
↪   and char.islower() else result.upper() if char.lower() in
↪   (ascii_lowercase := "abcdefghijklmnopqrstuvwxyz") else char
↪   for char in
↪   "cvpbPGS{arkg_gvzr_V'yy_gel_2_ebhaqf_bs_ebg13_jdJBFOXJ}")
```

This quick one-liner solves this problem. I hope you appreciate the elegance of Python a little bit after that too!

Solution to Question 1



...or you can just use a Caesar Cipher decoder from the Internet.

picoCTF

Question 2

Question 2

Insp3ct0r

| 50 points

Tags: picoCTF 2019 Web Exploitation

AUTHOR: ZARATEC/DANNY

Description

Kishor Balan tipped us off that the following code may need inspection: `https://jupiter.challenges.picoctf.org/problem/41511/` (link) or http://jupiter.challenges.picoctf.org:41511

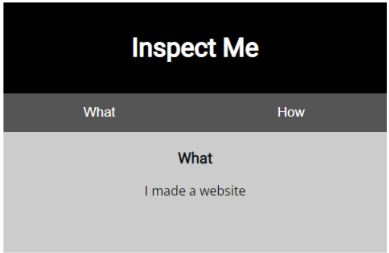
Hints

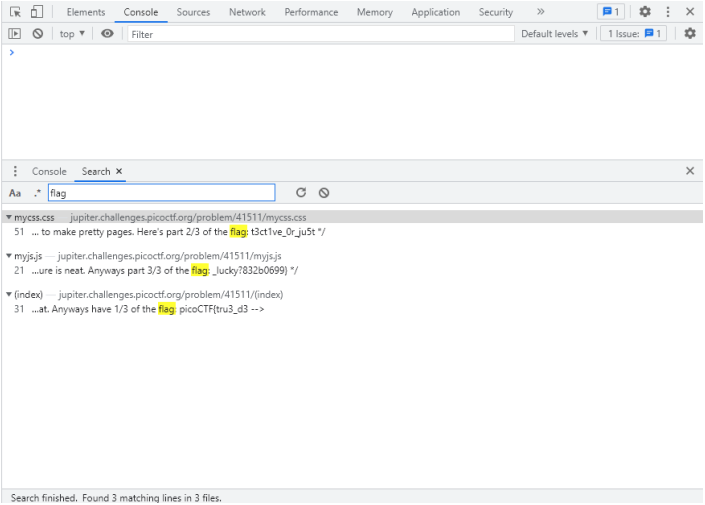
1 2

Solution to Question 2

This question simply need some inspection. Go to the website the question provided, `https://jupiter.challenges.picoctf.org/problem/41511/`. After that, Ctrl+Shift+I to open the development console, and then Ctrl+Shift+F to search the content in the JavaScript, HTML, and similar.
Type `flag` to search for the flag, and collect 3 pieces of flags in this website and submit — and the flag is obtained.

Inspect Picture in Browser





picoCTF

Question 3

Question 3

plumbing

| 200 points

Tags: picoCTF 2019 General Skills

AUTHOR: ALEX FULTON/DANNY TUNITIS

Description

Sometimes you need to handle process data outside of a file. Can you find a way to keep the output from this program and search for the flag? Connect to `jupiter.challenges.picoctf.org 7480`.

Hints

1 2

Solution to Question 3

This question asks for Linux skills, which are also necessary during the CyberPatriot competition. Essentially, when you connect to **`jupiter.challenges.picoctf.org:7480`**, the server shall response with plain text content that contains the flag.

This one-liner will do it:

```
nc jupiter.challenges.picoctf.org 7480 | grep 'pico'
```

CyberPatriot

As the competition of this year is coming for CyberPatriot, the plan for today is to talk more about it.

CyberPatriot

Shell

The kernel of a Linux system is in charge of allocating and scheduling hardware resources, which is critical to the system's regular operation. However, in order to interact with it, we need a wrapper as handling the Linux kernel directly is really difficult.

What is Bash?

`Bash` (bourne-again shell), is a program that take commands from the keyboard and gives them to the operating system to perform. Other competiors exists, such as `zsh`, `fish`, and `powershell`. `bash` offers one benefit that set it apart from its competitors — *industrial standard*. Many Linux distributions installed this out of box, and knowing this gives you ability to work on a variety of other platforms.

Figure: Logo of Bash

Common Bash Shortcuts

Key	Function
⇥	complete command/direction path
Ctrl + c	terminate current process
Ctrl + shift + c	copy selection area
Ctrl + d	terminate keyboard input by sending `SIGINT`
Ctrl + l	clear terminal content. scroll up if you want to see old output
Ctrl + r	search your history
↑, ↓	previous executed command, next executed command
Ctrl + w	delete word
Ctrl + u	delete line

CyberPatriot

Commands

Commands

The general format of a command is denoted as follows. Usually, `optional_parameters` are quoted with brackets, where `positional_parameters` are quoted with angled brackets or nothing.

```
executable_name [optional_parameters] positional_parameters
```

Two formats of the parameters exists:

Short-format `-\w`. There is only one hyphen before the parameter, followed by a character, e.g., `sort -n -u` and `sort -nu`
Short format have a short hand — you can mix them together without type `-` multiple times.s

Long-format `--\w+`. There is two hyphen before the parameter, followed by a word, e.g., `sort -u` and `sort --unique`.

CyberPatriot

Files

pwd

Welcome to the first bash command you have typed in your life (hopefully). **pwd** prints the current working directory.

```
[root@yuck ~]# pwd
/root
```

print the current directory and resolve symbol links

```
[root@yuck ~/Lecture/Symb]# pwd
/root/Lecture/Symb
[root@yuck ~/Lecture/Symb]# pwd -P
/root/Lecture/Test
```

ls

List files in the directory (notice that directory is also a file).

```
ls [option] [filename]
```

option	function
`-a`, `--all`	show hidden files (list all files)
`-F`	append a character to each file to indicate the file type
`-1`	one file per line
`-h`	display file size in human readable format
`-d`	display directory, but not their content
`-l`/`--format=verbose`	show name, number of hard links, file type, file mode, owner name and group, size, and time stamp, etc.

ls (cont.)

```
[root@yuck ~/Lecture]# ls -l
total 12
prw-r--r-- 1 root root    0 Jul  3 09:14 fifo
-rw------- 1 root root   37 Jul  3 08:55 history
lrwxrwxrwx 1 root root    4 Jul  3 09:01 Symb → Test
drwxr-xr-x 2 root root 4096 Jul  3 09:01 Test
-rwxr--r-- 1 root root  125 Jul  2 20:45 test.py
```

The first character in the first column represents the file type.

- `b` is block
- `d` is directory
- `c` is character device (find some in `/dev/`)
- `p` is pipe
- `l` is link
- `s` is socket
- `-` is regular file

ls (cont.)

```
[root@yuck ~/Lecture]# ls -a
.   ..   fifo   history   Symb   Test   test.py
```

F append a character to each file to indicate the file type.

```
[root@yuck ~/Lecture]# ls -1F
fifo|
history
Symb@
Test/
test.py*
```

- * executable
- / directory
- @ symbolic link
- = sockets
- **nothing** regular
- | FIFO/named pipe

find I

Find file according to certain predicates.

```
find [location] predicates
```

option	function
`-name`	Match name pattern
`-path`	Match path pattern
`-perm`	Match permissions (append a `-` to make permission included)
`-user`	Match the owner
`-group`	Matches all groups
`-mtime -n +n`, `-atime -n +n`, `-ctime -n +n`	Mod time, access time, and create time. (-n refers to n days, +n refers to n days ago)
`-nouser`, `-nogroup`	Matches files that have no owner/groups

find II

`-type b/d/c/p/l/f`	Match file type (followed by subtitle letters for block device, directory, character device, pipe, link file, text file)
`-size`	Match file size (+50KB for finding files over 50KB, and -50KB for finding files smaller than 50KB)
`-exec ……` `{}\;`	Processing each matching result concurrently. `{}` is replaced/expanded by actual file name.
`-not`, `-and`, `-or`	Combine predicates

find EXAMPLES

Find all files that start with **host**

```
[root@yuck /etc]# find -name "host*"
./host.conf
./apparmor.d/abstractions/hosts_access
./hostname
./hosts.allow
./avahi/hosts
./hosts
./hosts.deny
./hostid
```

-exec

most important stuff, `-exec`.

start from `-exec` token, every parameter here after is consider as executable, or parameters to such executable. the first encounter of `\;` is considered the end of the command. `{}` serves as the placeholder of the filename that are currently under examination.

since the execution is concurrent, so thousands of millions of files can be quickly handled by this. compare this with `Executors.ThreadPoolExecutor.submit()`, you would see the usefulness of this command immediately.

EXAMPLE OF exec

find text file that is not empty and print its path and content.

```
[root@yuck /]# find ~/Lecture/ -name "*.txt" -exec bash -c 'if
↪  [ "`cat {}`" ≠ "" ]; then echo {}; cat {}; fi' \;
/root/Lecture/Test/2/c/test.txt
Hello World!
```

touch

Change a file access and modification times (atime, mtime).

```
touch [option] path
```

option	function
`-a`	modify atime
`-m`	modify mtime
`-d`	modify both atime and mtime

touch EXAMPLE

create a file called **test**

```
[root@yuck /h/s/L/create]# touch test
```

change its mtime and atime

```
[root@yuck /h/s/L/create]# touch -d '2022-06-04
↪  13:29:23.809509936' test
[root@yuck /h/s/L/create]# stat test | rg Modify
Modify: 2022-06-04 13:29:23.809509936 -0400
```

mkdir

create directory

```
mkdir [option] path ...
```

option	function
`-p`	recursively create (i.e., if parent directory does not exists, create one)

mkdir EXAMPLE

create directory called **tdir**

```
[root@yuck /h/s/L/create]# mkdir tdir
```

create multiple directory, **adir/adir,adir/bdir,bdir/adir,bdir/bdir**

```
[root@yuck /h/s/L/create]# mkdir -p {a,b}dir/{a,b}dir
[root@yuck /h/s/L/create]# tree
.
├── adir
│   ├── adir
│   └── bdir
└── bdir
    ├── adir
    └── bdir

6 directories, 0 files
```

cp

copy files and directories.

```
cp [option] src ... dst
```

if **dst** exists and is a directory, **src** will be copied to **dst**

if **dst** exists and is a file, **cp** will ask you if you want to replace them

otherwise, new file is created

`cp` OPTIONS

option	function
`-d`	copy symbolic links rather than resolve them
`-i`	prompt whether to overwrite
`-L`, `--dereference`	resolve symbol links and make sure copies are regular file
`-p`	preserve all attributes of the original files
`-r`	recursive copy, must use this with directories
`-a`	archive, `-pdr`
`-f`	force

mv

Move or rename files and directories.

```
mv [option] src ... dst
```

option	function
`-n`	don't overwrite
`-u`	update files in dst, i.e., don't modify file with same or newer timestamp
`-i`	interactive

mv EXAMPLES

```
[root@yuck /h/s/L/create]# mv test{,bak}
[root@yuck /h/s/L/create]# ls
testbak
```

If you feel like some thing is suspicious during competition, try `sudo mv ^^I^^Ifilename{,.bak}` rather than delete them, so that, when scoring script punish you, you can `mv filename{.bak,}` to get your score back.

rm

Remove file or directory

```
rm [option] path
```

option	function
`-f`	no confirm
`-i`	ask before deletion (i.e., interative)
`-r`	delete the directory
`-v`	display the process (i.e., verbose)

```
[root@yuck /h/s/L/create]# rm testbak
[root@yuck /h/s/L/create]# ls
```

CyberPatriot

User management

Linux is a multi-user, multi-task operation system with high stability and security. Some configuration to limit the user permission is thus necessary in order to enjoy that. This part, we will introduce how to limit the user from read, write, execute, and delete the file. In addition, we will also introduce SUID, SGID, and sticky bit. Some hidden permissions will also be introduced.

User Types

`root` user has the highest permission in Linux. It can do anything. Therefore, it is a good practice to use `root` user to do the configuration — *that comes with huge responsibility!*. `su` and `sudo` command can be used to switch to `root` user.
The `root` user is determine by UID (user identification) rather than the name of the user. In Linux system, users are grouped by UID as the following categories.

UID 0 `root` user

UID 1–999 System user

UID 1000–60000 Normal user

id

Show the user ID and group ID of the current user (and other information).

```
id [option] [user]
```

The user argument can be used to specify the user.

option	function
`-u`	show the user ID
`-g`	show the group ID

useradd

Add a new user to the system.

```
useradd [option] username
```

option	function
`-d`	home directory
`-e`	expiration date in form of `YYYY-MM-DD`
`-g`	initial group
`-G`	supplementary group
`-m`	create home directory
`-s`	shell
`-u`	user ID

userdel

Delete a user from the system.

```
userdel [option] username
```

option	function
`-r`	delete home directory (strongly against that, because important data might be removed in such process, and you will be penalized in competiton for that.)
`-f`	force delete

usermod

Modify the user information.

```
usermod [option] username
```

option	function
`-c`	comment
`-d`	home directory, usually used with çmono!-m!
`-m`	move home directory
`-e`	expiration date in form of `YYYY-MM-DD`
`-g`	initial group
`-G`	supplementary group
`-l`	new username
`-s`	shell
`-U`	unlock the user
`-L`	lock the user. in such situation, the user can't login into the system.
`-u`	user ID

passwd

Change the password of the user.

```
passwd [option] username
```

option	function
`-l`	lock the user. in such situation, the user can't login into the system.
`-u`	unlock the user
`--stdin`	read the password from standard input, allow change the password through command line (without `expect`) `echo "C3yb@rP*atr@iot" | password --stdin yubo`
`-d`	delete password, i.e., user can login without password
`-e`	set the password to expire, i.e., force the user to change password upon next login
`-S`	show the password status

su

Switch to another user without logout.

```
su [option] [-] [username]
```

option	function
`-`, `-l`, `--login`	start a login shell as to avoid problem caused by environment variables. Advised to always do that.
`-c`, `--command=cmd`	execute the command and exit.

sudo

Execute a command as another user. Usually used to execute a command as **root** user.

```
sudo [option] [command]
```

option	function
-i	emulate a login shell, so that login specific files will be readed and you can execute command interatively.
-l	list the allowed and denied commands.
-e, **--edit**	edit file as a superuser
-u, **--user**	execute the command as another user.
-g, **--group**	execute the command as another group.

visudo

Change, add, or delete the sudoers file.

```
visudo
```

This command automatically check the syntax of the file and will not allow you to save the file if there is any error. It invokes `VISUAL_EDITOR`, `EDITOR` sequentially if any of them are set to allow you to edit the file. If none of them are set, it will use `vi` as the default editor.
The format of the file is as follows:

```
user host = (runas_user) command
```

visudo EXAMPLE

For example, the following line allows user **yubo** to run **ls** as user **root** without password.

```
yubo ALL = (root) NOPASSWD: ls
```

One may also put **ALL** to represent the highest privilege, i.e., **root**. This configuration allow **anish** to run all command as **root** without password.

```
anish ALL = (ALL) NOPASSWD: ALL
```

CyberPatriot

permissions

Each file in the Linux operating system has information on who owns it, to which group it belongs, and whether or not it is executable, readable, or writable.

Table: Permission

	File Type	
Permission	File	Directory
`r`	readable `cat`	listable `ls`
`w`	writable `vim`	writable `touch`
`x`	executable `./script`	into directory `cd`

The permissions are usually represented as string of `rwx`, where

- `r` means readable
- `w` means writable
- `x` means executable

The permissions are also separated as owner, group, and others. For example, `rwxr-xr--` means the owner has read, write, and execute permission, the group has read and execute permission, and others have read permission.

Octal representation of the permissions is as follows:

Table: Symbolic to octal Conversion

	Owner			Group			Others		
Permission	`r`	`w`	`x`	`r`	`w`	`x`	`r`	`w`	`x`
Octal number	4	2	1	4	2	1	4	2	1

To convert from symbolic to octal, use the above table and add up the numbers. For example, `rwx` is $4 + 2 + 1 = 7$.
To convert from octal to symbolic, use the above table and find the corresponding symbol. For example, **7** is $4 + 2 + 1$ and is `rwx`.

Table: Octal representation of the permission

Symbolic representation	Octal
`rwx`	7
`rw-`	6
`r-x`	5
`r--`	4
`-wx`	3
`-w-`	2
`--x`	1
`---`	0

chmod

Change the permission of a file or directory.

```
chmod [option] [mode] [file]
```

option	function
`-R`	change the permission recursively

The mode can be either symbolic or octal. For example, `chmod 777` and `chmod ugo+rwx` are equivalent. In addition, one can place a `+` or `-` before the mode to add or remove the permission. For example, `chmod +x` and `chmod ugo+x` are equivalent. Finally, put `u`, `g`, or `o` before the mode to change the permission of the owner, group, or others. For example, `chmod u+x` changes the permission of the owner to be executable.

chown

Change the owner of a file or directory.

```
chown [option] [user] [file]
```

option	function
`-R`	change the owner recursively
`--reference=path`	change the owner to the same as the owner of `path`

Notice the user can be specified by either name or user ID. For example, `chown yubo` and `chown 1000` are equivalent. In addition, by placing a colon before the user, one can change the group of the file. For example, `chown anish:cs` changes the group of the file to `cs` and the owner to `anish`.

SGID

The SGID permission is set by `chmod g+s` and is represented by `s` in the permission string. When a file has the SGID permission, the file is executed as the group of the file, not the group of the user who executes the file. For example, `sudo chmod g+s /dev/ps` makes ps obtain the privilege of `system` user group, thus allowing the user to see all the processes.

Sticky Bit/SBIT

The SBIT permission is set by `chmod o+t` and is represented by `t` in the permission string. When a directory has the SBIT permission, only the owner of the file or the owner of the directory can delete the file. For example, `sudo chmod o+t /tmp` makes the `/tmp` directory to be only deletable by the owner of the directory or the owner of the file.
This is usuaully used in remote file systems to prevent users from deleting each other's files.

Convert Special permissions

Those special permission may also be converted to octal. `SUID` converts to `4` and `SGID` converts to `2` and `SBIT` converts to `1`.

Access Control List

ACL provides a more fine-grained control of the permission. ACL is a list of users and their permissions. For example, `u:rwx` means the user has read, write, and execute permission. The ACL can be set by `setfacl` and `getfacl`.

setfacl

Set the ACL of a file or directory.

```
setfacl [option] [file]
```

option	function
`-m`	modify the ACL
`-M`	read the ACL from a file
`-x`	remove the ACL
`-b`	remove all ACL
`-R`	set the ACL recursively

For example, `setfacl -Rm u:anish:rwx /root` sets the ACL of the root directory to be readable, writable, and executable by the user `anish`.

getfacl

Get the ACL of a file or directory.

```
getfacl [option] [file]
```

option	function
`-R`	get the ACL recursively

getfacl

For eaxmple, get the ACL of `/etc/fstab`.

```
getfacl /etc/fstab
```

It is usuaully suggested to use `getfacl` to backup the ACL of a directory before modifying the ACL. For example, `getfacl -R /etc > /etc/acl` backs up the ACL of the `/etc` directory. After that, `setfacl --restore /etc/acl` restores the ACL.

CyberPatriot

Dichotomize

COMPETITION SCHEDULES/COMMITMENTS

CyberPatriot

defensive cybersecurity
six-hour virtual competitions once a month
application process: 13 people
OPEN TO EVERYONE
first competition October 20th-22nd
our flagship competition

picoCTF

offensive cybersecurity

mid-march, as much as you can in one week

everybody can compete!

little room for trophies, advancement, etc.

Scan this qrcode to determine which side you wish to come (CyberPatriot/picoCTF). However, you can take both sides.

Figure: Choice form: `https://tinyurl.com/GSMSTcybersplit`

Scan this qrcode in order to join discord server.

Figure: Discord Server: `https://discord.com/invite/FFmwPacn`

Thank you.