Scripting for Cyber Defense Competitions

What is a script?

- Lots of commands, such that you don’t need to re-type them out
- Speeds up what you want to do

I’ll give this in the context of CCDC, but it applies to NCAE

Pros and Cons

Pros:

- Drastic speed up, especially the longer your script is
- Red team has scripts, so scripting is one of the few ways to beat them

Cons:

- Scripting takes WORK and KNOWLEDGE
- You *probably* need to cover every edge case
- ^^^ scoping

Python and Bash

These are two of the best languages for most of what you will want to do

Ease of use + functionality is the sweet spot

More advanced projects, C/C++ might be more useful, maybe Perl? But that's *significantly* more advanced

I'll put our focus on bash because it's better than Python for this

Bash

What is bash?

- Bourne Again SHell (sh is the Bourne SHell)
- `echo $SHELL` (bash, zsh, fish?)
- CLI + Programming Language

- Builtins vs commands, `$PATH`, tab complete, pipes, substitution, arithmetic, and all kinds of other cool features

Crash Course

shebang

Every file should start with a shebang:

`#!<interpreter>`

`#!/bin/bash`

`#!/usr/bin/env bash` - this is viewed as “best”

Variables

You can declare and use variables in bash

- `export VAR=1` - usable during the *whole* session
- `VAR=1` - just during that process
- `echo $VAR` - use the variable

Data types:

- `VAR=14`
- `VAR="some string"`
- `VAR=( "this" "is" "an" "array")`

Spaces *matter*

Functions

- Defined with `function`
- Local variables are possible
- Called using just the function name (no parenthesis)
- Arguments space separated

```
function get_input_string {
    read -r -p "$1" input
    echo "$input"
}
function print_banner {
    echo “#### Welcome! ###”
}

print_banner
input=$(get_input_string "Enter input: ")
echo $input
```

Logic Flow

- Remember, spaces matter
- Conditions are in brackets
- Must wrap in `do`:`done`, `if`:`fi`, `case`:`esac`, etc

```
while [ $done != “true” ]; do
    input=$(get_input_string “> “)
    if [ $input == “” ]; then
        done=“true”
    elif [ $input == “train” ]; then
        sl
    else
        echo -n “You said: ”
        echo $input
    fi
done
```

Useful tricks

- `$netid=$(cat /home/blackteam/users.txt | grep $username | cut -d' ' -f1)`
- Wildcards: `?`, `*`, `[ ]` (range), `{}` (set), `##`
- Regex with `grep` (learn it, it's useful I promise)
- `$_` last argument of last single command run in foreground after expansion
- Arguments:
  - `$@` either each argument as a string (in quotes), or arguments split by IFS
  - `$*` either a string of ALL arguments, IFS separated (in quotes), or same as above
  - `$1` argument number 1, etc
- `${}` - auto-expand variables inside
- `> <` - output and input redirects: `2>/dev/null`, `fd` are also useful to know
- `env`s are useful
- `sudo !!` - reruns the last command with sudo

Learn by doing

If you want to *learn* bash, not just use it, go write a couple things with it.

You no longer need it for NCAE, but what could improve your QoL? Update your `.bashrc`?

https://github.com/BYU-CCDC/public-ccdc-resources/blob/main/linux