Meeting Start: **5:32 PM**

- Questions
    - Questions related to HackTheBox Module?
        - Write up posted in Discord
    - Questions related to upcoming competitions?
- Secure Programming
    - Definition: Designing code to prevent vulnerabilities, attacks, etc in an application
    - Specific approaches for every library, package, language, framework, etc
    - Considerations when secure programming
        - Input sanitization / validation
        - Output encoding
        - Threat modeling
        - Improper password requirements
        - Lack of proper access control
        - Lack of proper error handling / logging
    - Auditing insecure programs
        - Stack Buffer Overflow Demo
            - C program using GNU debugger
            - ‘gets’ method insecure
            - Long string doesn’t get checked / input not sanitized - results in registers being overridden
            - Full demo on presentation in Discord
    - Today’s Exercise (SQL Injection)
        - PicoCTF Gym
            - Challenges
                - SQLiLite
                - More SQLi (need Burp Suite for this)
                - SQL Direct (Uses PostgreSQL)

Meeting End: **6:00 PM**

IUP Cybersecurity Club

Meeting 7:11/13/2024

Housekeeping

- Any questions on the previous HTB module?
- Any general questions regarding
  - HTB Binary Badlands competition
  - Upcoming NCAE competition

Secure Programming

An Overview

What is Secure Programming?

- Designing code specifically to prevent vulnerabilities, attacks, and other threats within an application
- “Security first” approach (Security by Design)
- There are specific approaches for every library, package, language, framework imaginable

Sounds Simple, Right?

- I really only have to look out for…
  - Buffer overflow
  - Path traversal
  - Injection
- And all that is pretty much inherently handled by standard issue programming languages/frameworks right?

Of Course Not!

- As a developer/auditor, you should look for…
  - **Password requirements**
    - Entropy, MFA, disallow reusing old passwords, password change rate
      - What kind of MFA is acceptable security-wise?
  - **Access control**
    - Admin
      - Managing admin accounts for applications separately from regular accounts
      - Stricter security for admin accounts, more control/monitoring
      - Admins will have their own applications/pages/apis etc.
    - Preventing access (Zero-Trust principles)
      - Require validation at every layer
      - Using tokens to validate API calls application side
        - Can prevent unauthorized access to data
  - **Error handling/logging**
    - Replace default error logging
      - Default errors give indication of what software you are using (maybe even the version)
    - Avoid giving useful information in error logs
      - Ex. Return an error saying, “Please don’t insert a really long string into this input, it will break this awful validator I’m using on this outdated software version that is also exploitable.”

More Considerations

- **Input sanitization/validation**
  - Sanitizing/validating input at every layer possible
- **Output encoding**
  - Encoding any outgoing information
    - Can give information about the system if not properly configured
    - Could otherwise be intercepted outright and stolen
  - Server → Client
  - Client → Server
- **Threat modeling**
  - Document
    - Document all possible vulnerabilities within the scope of your investigation
  - Locate
    - Locate the specific points of access/files/etc. that are vulnerable
  - Address
    - Fix the actual vulnerabilities with implementation of best security practices
  - Validate
    - Test the results to ensure the vulnerabilities are patched

Examples of Insecure Programming

- **Improper Password Requirements** (2015)
  - Ashley Madison site users had their data stolen
  - Inadequate password requirements
    - Short, no special characters/number requirement
  - Poorly stored passwords
    - Hashed poorly or sometimes not at all
  - How to fix?
    - Hash passwords, require frequent password changes (~1/year minimum)
    - Implement MFA
    - Harsher password requirements
- **Lack of proper access control** (2016)
  - Uber's cloud servers were illegally accessed
  - An access token was exposed in Github
    - 57 million user records stolen
  - How to fix?
    - Access tokens should have expiration dates
    - Obviously should NOT be uploaded to a Github repo ever (whether it is public or private)
    - Can set up detection of keys and other sensitive info, but can also implement security checks in the application pipeline at every step to ensure that these leaks don't happen

Insecure Programming cont.

- **Lack of proper error handling/logging (2017)**
  - Cloudflare data was leaked by a buffer overflow error (Cloudbleed)
    - Lack of error handling – errors where sensitive information was being returned and cached in random users browsers was not detected by Cloudflare's error handling systems at all
    - Was discovered by Google's Project Zero team and reported to Cloudflare
    - "It turned out that the underlying bug that caused the memory leak had been present in our Ragel-based parser for many years but no memory was leaked because of the way the internal NGINX buffers were used. Introducing cf-html subtly changed the buffering which enabled the leakage even though there were no problems in cf-html itself." – Cloudflare CTO
    - 1 in 3.3 million requests potentially resulted in memory leakage
    - How to avoid?

Auditing Insecure Programs (Buffer Overflow)

- Computer Memory
  - Storage of instructions and data
    - “Von Neumann architecture”
- Running process is stored in memory
  - Allocated memory is accessed in however the way the program is intended
  - Can be manipulated by the program itself
- Demo of stack buffer overflow in C using GNU Debugger (gdb)

For Today's exercise (SQL Injection)

- Injecting SQL into forms in HTML in order to execute SQL on the client side

- By adding ' OR '1'='‘1 to the end of the input, you can escape the quotes and execute and inject SQL into the existing statement

```php
<?php
if (isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Connect to the 'database' (super secure btw)
    $conn = new mysqli("localhost", "root", "password", "example_db");

    // (No Input Sanitization)
    $query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        echo "Login successful!";
    } else {
        echo "Invalid username or password.";
    }

    $conn->close();
}
?>
```

Now time for the exercise!

- Complete the following challenges in Pico CTF (SQL Injection)
- Go to PicoCTF Gym
  - SQLiLite
  - More SQLi (Need Burp Suite for this)
  - SQL Direct (Uses PostgreSQL)

Meeting Start: **5:31 PM**

- Housekeeping
    - Questions related to HTB module this week?
    - HTB Binary Badlands Competition
        - 13th - 15th December, 2024
    - NCAE Signups (more members?)
        - Interest in particular areas
            - Infrastructure needed
            - CTF?
    - Any general questions?
- NCAE Minihack
    - Full guide on Discord resources
        - Complete the minimum to make all lights green for minihack
        - Discuss full competition
            - What to do w/ environment we created
            - Backups, DNS, protecting services, VMs, etc
        - For those interested in CTFs / Infrastructure. . .
            - Watch NCAE YouTube tutorial
            - https://www.youtube.com/playlist?list=PLqux0fXsj7x3WYm6ZWuJnGC1rXQZ1018M
- Any suggestions / requests / interests for topics to cover at the club?
    - Let us know!

Meeting End: **5:42 PM**

IUP Cybersecurity Club

Meeting 6:11/6/2024

Housekeeping

- Questions about HTB module this week?
- HTB Binary Badlands competition
  - 13th - 15th December, 2024
- NCAE signups (more members?!)
- Any general questions?

What are we doing today?

- **NCAE Minihack** (for real this time)
- Check discord for the **FULL GUIDE**
  - Complete the minimum to make all the green lights for the Minihack
  - Then we can discuss the full competition
    - *What to do with the environment we just created*
    - *Backups, DNS, protecting the services/VMs etc.*
  - If you signed up for the competition and want to do infrastructure
    - Watch the NCAE Youtube tutorial available through their website as well
- We have to divide the team for CTFs/Infrastructure
  - There are CTF labs as well on the NCAE site

Final Thoughts

- If you have any suggestions/requests for topics to cover at club…
  - Let us know RIGHT NOW
  - Or later

Meeting Start: **5:31 PM**

- NCAE Cybergames
    - Slots filled for Team 1 but if there's enough interest we can have an additional Team 2
    - For those signed up:
        - Interest in focus group to do mini-hack and bring everyone up to speed
        - Ask questions
        - Learn VIM Basics
- Additional CTF Competitions
    - Interest?
    - HTB University CTF
        - December 13th-15th
        - Price money?
    - PicoCTF
        - ~ Mid-March
- Fuzzing
    - Introduction
        - Method of testing via automated software that provides invalid, unexpected, or random data as inputs to a program
            - Inputs could be randomly generated or pulled from a wordlist
        - Purpose:
            - Finding unexpected behavior or hidden functionality
                - Could be exploited
    - Web Fuzzing
        - Used to uncover:
            - Vhosts and subdirectories
            - APIs
            - GET and POST parameters/values
        - Uses wordlists typically
        - Uncovers attack vectors for exploitation
    - How to fuzz the web
        - Open-Source Tools
            - Ffuf
            - Dirbuster
            - Gobuster
        - **Step-by-Step explanation on slideshow**
    - Practical Example
        - **DO NOT ATTEMPT ON REAL / PUBLIC URL'S, ESPECIALLY WITHOUT GIVEN PERMISSION**
    - Any further questions, reach out to club officers
- Natas OTW - Continued
- If interested in signing up for any upcoming competitions, reach out to Thomas McGillan

Meeting End: **6:00 PM**

IUP Cybersecurity Club

Meeting 8: 10/30/2024

General Housekeeping

- NCAE Cybergames
  - One more slot available on Team1 (can still have an additional Team2)
  - For those signed up already:
    - Anyone interested in a focused group to do mini-hack and bring everyone up to speed?
      - If not, please do the mini hack on your own
    - It is okay to ask questions!
    - LEARN VIM BASICS PLEASE
- Additional CTF Competitions
  - Anyone interested?
  - HTB Uni CTF
    - 13th - 15th December, 2024
  - PicoCTF
    - Sometime mid March

Fuzzing

- What is fuzzing?
  - A method of testing via automated software that provides invalid, unexpected, or random data as inputs to a program
    - These inputs could be randomly generated or pulled from a wordlist.
- What is it used for?
  - Finding unexpected behavior or hidden functionality
    - This behavior/functionality could then be leveraged for exploitation

Web Fuzzing

- Similar to regular fuzzing
  - Seeks to uncover things such as:
    - Vhosts and subdirectories
    - APIs
    - GET and POST parameters/values
  - Typically uses wordlists to accomplish tasks
    - https://github.com/danielmiessler/SecLists/
  - Uncovers attack vectors for exploitation

How the ffuf do I Fuzz the Web?

- Various open-source tools available:
  - ffuf
  - dirbuster
  - gobuster
- Order of operations:
  - Find target site
  - Fuzz for extensions on index (.html, .php, etc)
    - Almost all sites will have a page titled index
  - Fuzz for vhosts/subdomains
    - Ex: https://FUZZ.iup.edu
  - Fuzz for subdirectories/pages of each vhost/subdomain
  - Fuzz for GET and POST parameters
  - Fuzz for corresponding parameter values
  - PROFIT (maybe)

Using ffuf for Fuzz Time

- A ffuf command will have the following:
  - A wordlist, specified with –w
  - A target site, specified with –u
- Optional useful parameters include:
  - -fs
    - Filters out certain size of response
  - -fc
    - Filters out certain response codes (200, 301, etc)
  - -e
    - Specifies extension for pages (.html, .php, etc)
  - -t
    - Specifies number of threads (speeds up request rate)
  - -recursion
    - Enables recursive fuzzing
  - -recursion-depth
    - Specifies depth of recursion

Fuzzing for Extensions

- ffuf –w /PATH/TO/seclists/Discovery/Web-Content/web-extensions.txt:FUZZ -u http://SERVER_IP:PORT/indexFUZZ
- -w specifies wordlist
  - web-extensions.txt
- -u specifies target

```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.1.0-git
------------------------------------------------

 :: Method           : GET
 :: URL              : http://SERVER_IP:PORT/blog/indexFUZZ
 :: Wordlist         : FUZZ: /opt/useful/seclists/Discovery/Web-Content/web-extensions.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 5
 :: Matcher          : Response status: 200,204,301,302,307,401,403
------------------------------------------------

.php                    [Status: 200, Size: 0, Words: 1, Lines: 1]
.phps                   [Status: 403, Size: 283, Words: 20, Lines: 10]
:: Progress: [39/39] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

Fuzzing for Vhosts/Subdomains

- Subdomain:
  - ffuf –w /PATH/TO/seclists/Discovery/Web-Content/subdomains-top1million-5000.txt:FUZZ –u http://FUZZ.SERVER_IP:PORT/
- Vhost:
  - After adding target ip and url to /etc/hosts
  - ffuf –w /PATH/TO/seclists/Discovery/Web-Content/subdomains-top1million-5000.txt:FUZZ –u http://TARGET.URL/ -H ‘Host: FUZZ.TARGET.URL’
  - -H is used to specify certain headers in requests
    - In this case the Host header

```
mail2                   [Status: 200, Size: 900, Words: 423, Lines: 56]
dns2                    [Status: 200, Size: 900, Words: 423, Lines: 56]
ns3                     [Status: 200, Size: 900, Words: 423, Lines: 56]
dns1                    [Status: 200, Size: 900, Words: 423, Lines: 56]
lists                   [Status: 200, Size: 900, Words: 423, Lines: 56]
webmail                 [Status: 200, Size: 900, Words: 423, Lines: 56]
static                  [Status: 200, Size: 900, Words: 423, Lines: 56]
web                     [Status: 200, Size: 900, Words: 423, Lines: 56]
www1                    [Status: 200, Size: 900, Words: 423, Lines: 56]
<...SNIP...>
```

Fuzzing for Parameters (GET/POST)

- GET
  - ffuf –w /PATH/TO/seclists/Discovery/Web-Content/burp-parameter-names.txt:FUZZ -u http://SERVER_IP:PORT/TARGET_PAGE?FUZZ=key
- POST
  - ffuf -w /PATH/TO/seclists/Discovery/Web-Content/burp-parameter-names.txt:FUZZ -u http://SERVER_IP:PORT/TARGET_PAGE -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded'
- -X specifies HTTP Method
- -d represents POST data

```
 :: Method           : POST
 :: URL              : http://admin.academy.htb:PORT/admin/admin.php
 :: Wordlist         : FUZZ: /opt/useful/seclists/Discovery/Web-Content/burp-parameter-names.txt
 :: Header           : Content-Type: application/x-www-form-urlencoded
 :: Data             : FUZZ=key
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
 :: Filter           : Response size: xxx
------------------------------------------------

id                      [Status: xxx, Size: xxx, Words: xxx, Lines: xxx]
```

Fuzzing for Parameter Values (GET/POST)

- GET
  - ffuf –w /PATH/TO/VALUE_WORDLIST:FUZZ -u http://SERVER_IP:PORT/TARGET_PAGE?=key
- POST
  - ffuf -w /PATH/TO/VALUE_WORDLIST :FUZZ -u http://SERVER_IP:PORT/TARGET_PAGE -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded’
- -X specifies HTTP Method
- -d represents POST data

Questions?

- If not and time permits, go and do more Natas OTW
- See me for event sign-ups if interested
  - NCAE Cybergames
    - February 17th, 2024
  - HTB Uni CTF
    - December 13th - 15th, 2024
  - PicoCTF
    - TBD (Sometime mid March)

Meeting Start: **5:32 PM**

-
- Web Vulnerabilities
    - ***Full Slideshow w/ detailed descriptions + examples in Discord***
    - Broken Authentication / Access Control
        - Definition(s)
            - Broken Authentication - allows attackers to bypass authentication
            - Broken Access Control - allows attackers to access things they shouldn't
        - Prevention
            - Deny by default
            - Disable web server directory listing
            - Log access control failures
            - Implement stateful session identifiers + invalidate them on logout
        - Example
    - Command Injection
        - Definition
            - Execution of arbitrary commands on web server via web app
        - Prevention
            - Filter + Sanitize user input
            - Escape / filter special characters
        - Example
    - Cross-Site Scripting (XSS)
        - Definition(s) + Types
            - Reflected - injected script reflected off web server
            - Stored - injected script stored on target server
        - Prevention
            - Sanitize user input
            - Encode data on output
            - Use appropriate response headers
        - Example
- OverTheWire Natas Exercises
    - Focuses on Server-side Web Security
- NCAE Competition
    - If interested, communicate with President Thomas
    - February 17th, 2025

Meeting End: **6:00 PM**

IUP Cybersecurity Club

Meeting 7: 10/16/2024

Understanding Web Vulnerabilities

- What is a web vulnerability?
  - A weakness in a web application that attackers can exploit
- Why does web security matter?
  - Web applications can host pieces of sensitive information
    - Ex: SSN, DoB, Addresses, etc.
- Potential impact of web vulnerabilities
  - Data breaches and financial losses
  - Service downtime
  - Loss of trust
  - Legal consequences

Broken Authentication/Access Control

Broken Authentication

- Allows for attackers to bypass authentication functions
- Can allow:
  - Normal users becoming an admin
  - Attacker logging in without valid credentials

Broken Access Control

- Allows for attackers to access things that they shouldn’t
- Ranked #1 on OWASP Top 10
- Ex: Normal users gaining access to an admin panel

Broken Authentication/Access Control cont.

Prevention

- Deny by default except for public resources
- Disable web server directory listing
- Log access control failures
- Implement stateful session identifiers and invalidate them on logout

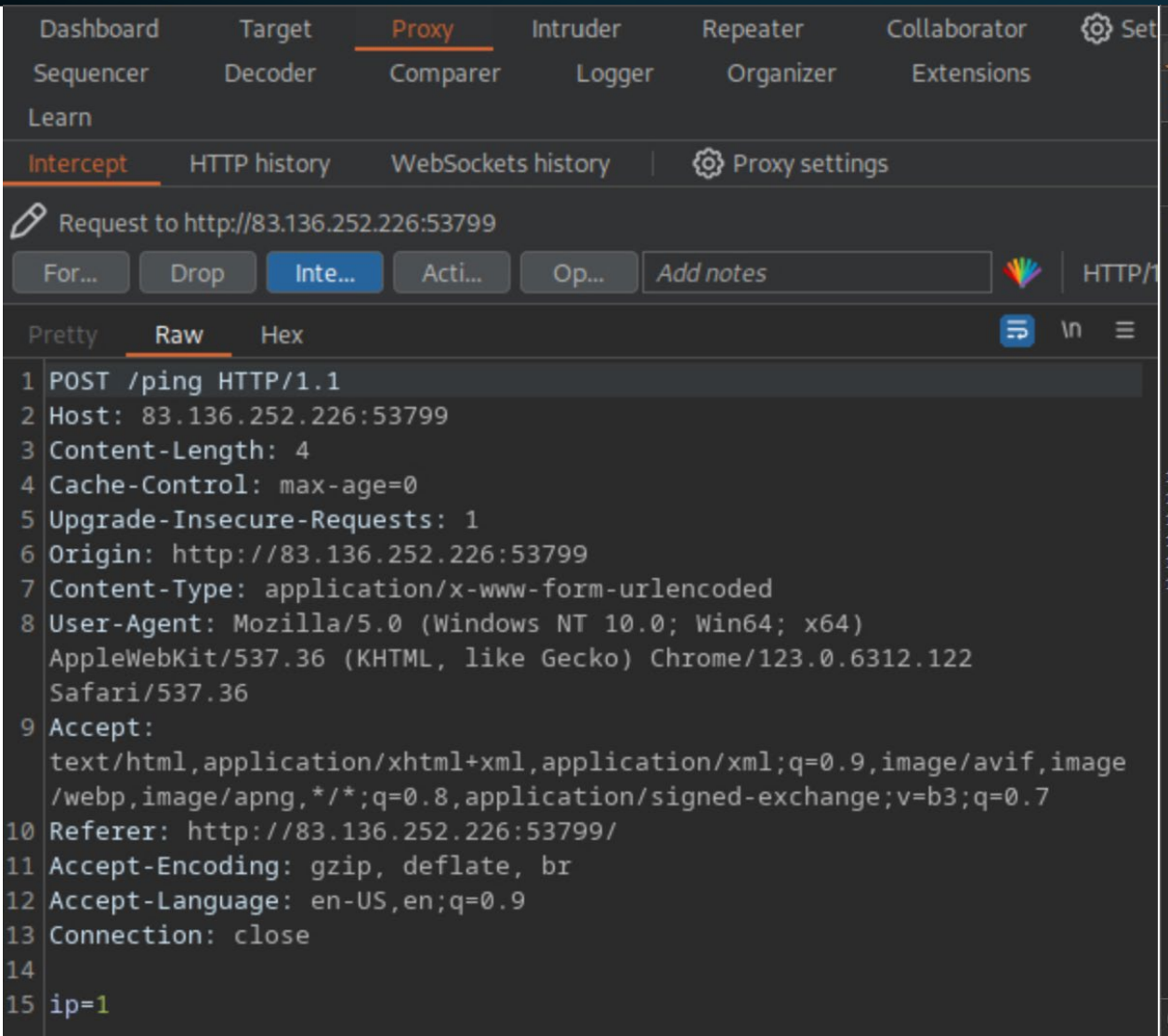
Example Scenario

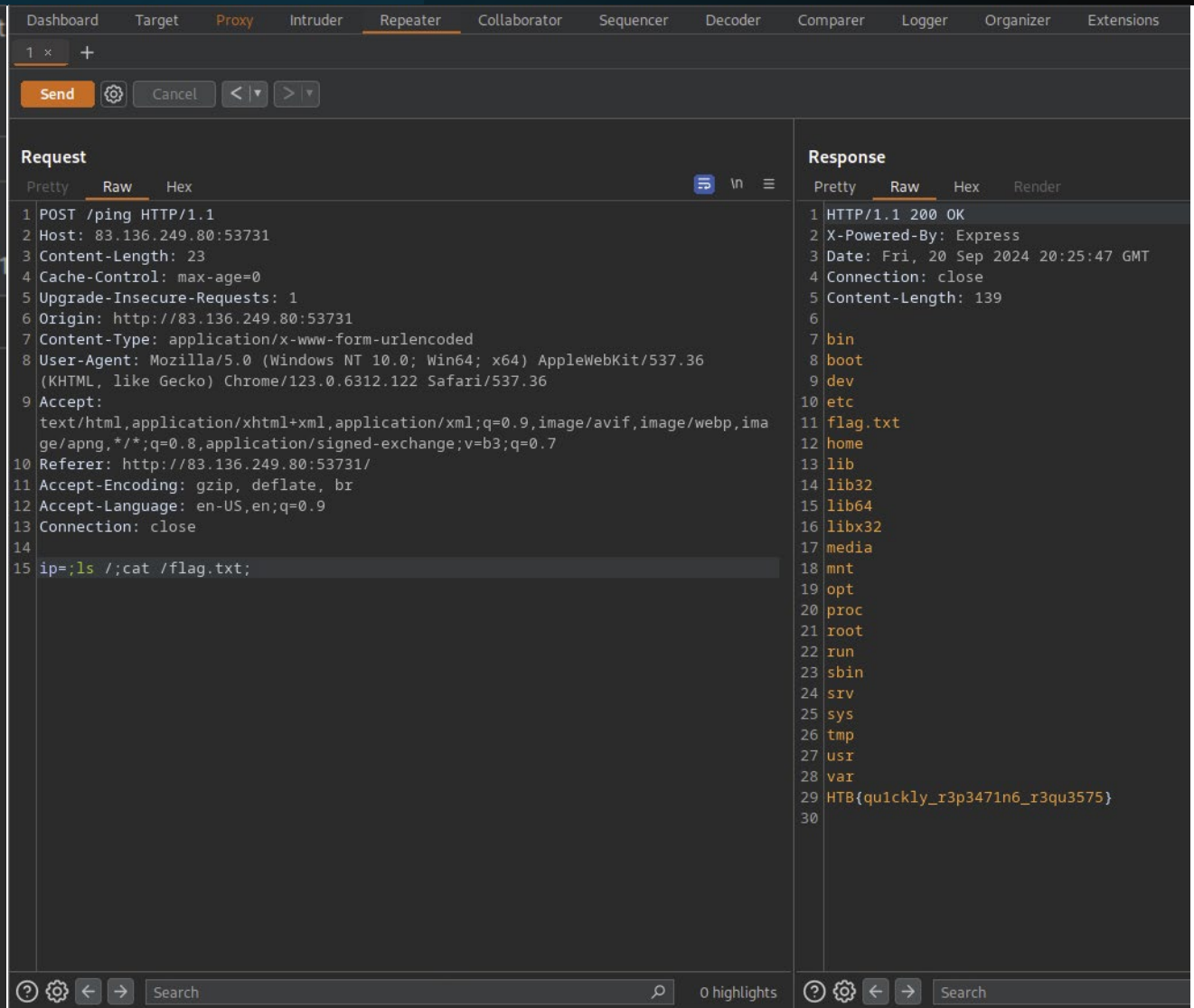
- An attacker simply modifies the browser's 'acct' parameter to send whatever account number they want. If not correctly verified, the attacker can access any user's account.
  - https://example.com/app/accountInfo?acct=notmyacct

Command Injection

- Execution of arbitrary commands on the web server via the web application
- Occurs when applications pass unsafe user input to a system shell
  - Commands are executed with the privileges of the application
- Prevention
  - Filter and sanitize user input
  - Escape special characters (. , ‘ “ \ & ;)

Command Injection Example





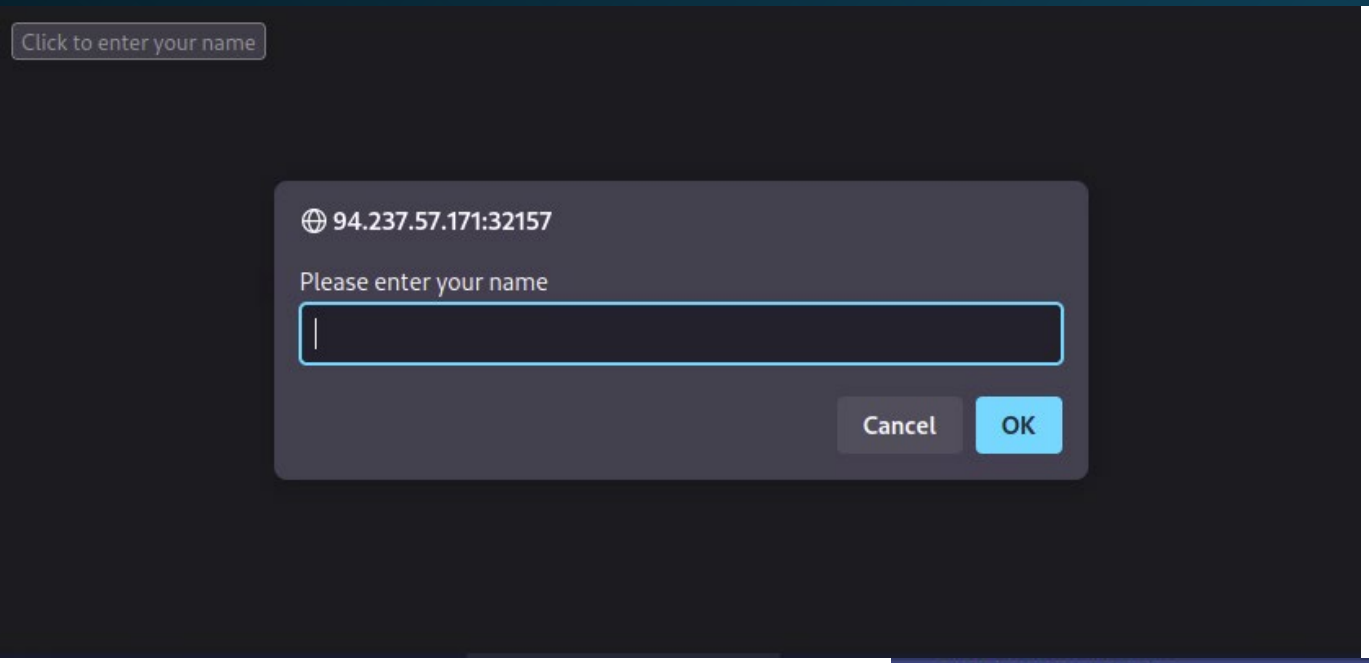
Cross-Site Scripting

- A type of injection in which malicious scripts are injected into otherwise benign and trusted websites
- Occurs when an attacker uses a web application to send malicious code to a different end user
  - Typically in the form of a browser side script i.e. Javascript
- Types of XSS
  - Reflected
    - Injected script is reflected off web server
      - Delivered via malicious link
        - https://vulnerable-website.com/search?q=<script>alert('XSS')</script>
  - Stored
    - Injected script is stored on the target server
      - In database, messages, visitor log, comments, etc.

Cross-Site Scripting cont.

- Prevention
  - Sanitize user input
  - Encode data on output
    - Prevents input from being interpreted as active content
  - Use appropriate response headers
    - If response should not contain HTML or JavaScript, make sure the Content-Type and X-Content-Type-Options headers are set to reflect that

Cross-Site Scripting Example



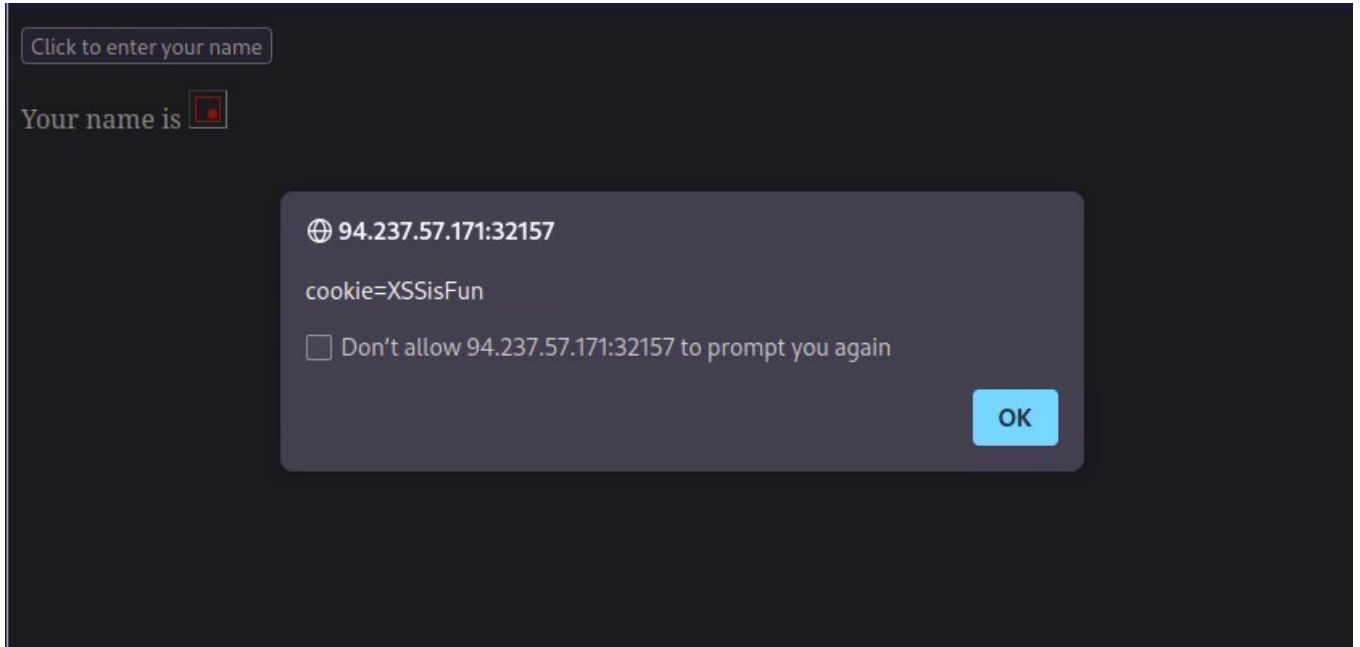
← Vulnerable Website

Vulnerable JavaScript →

```
<!DOCTYPE html>
<html data-darkreader-mode="dynamic" data-darkreader-scheme="dark"> event
▶ <head> ··· </head>
▼ <body>
    <button onclick="inputFunction()">Click to enter your name</button> event
    <p id="output"></p>
  ▼ <script>
      function inputFunction() { var input = prompt("Please enter your name", ""); if (input != null) {
      document.getElementById("output").innerHTML = "Your name is " + input; } }
    </script>
  </body>
</html>
```

Cross-Site Scripting Example cont.

- When <img src="/" onerror="alert(document.cookie)"> is inputted:
  - Website triggers an alert with the user's cookie value
  - This alert function could be substituted for more malicious functions:
    - Used to steal authentication and session tokens
    - Used to redirect to attacker's malicious site
    - Used to execute malicious scripts on target system



Conclusion

- Sanitize and don’t trust user input
  - Prevents numerous issues/vulnerabilities
- Many more vulnerabilities than presented
  - Presented were some common vulnerabilities
  - Check out owasp.org for listing and guides on vulnerabilities
- Questions?

- Proceed with OverTheWire Natas Exercises
  - Focuses on server-side web security
- Talk to Thomas if you want to participate in NCAE Competition
  - Need your school email to sign up

References

- https://owasp.org/www-community/vulnerabilities/
- https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- https://owasp.org/Top10/A03_2021-Injection/
- https://owasp.org/www-community/attacks/Command_Injection
- https://portswigger.net/web-security/cross-site-scripting
- https://www.rapid7.com/fundamentals/web-application-vulnerabilities/
- https://academy.hackthebox.com/module/75/section/764

Meeting Start: **5:32 PM**

- Vim/Neovim Tutorial (***Full slides in discord***)
    - Brief history of Vim & Neovim
    - About Vim/Neovim
        - Modal Text Editor
        - Works in buffer - Files not created until saved
        - Vim Script
            - Create macros
    - Modes
        - Normal - default
            - Used to navigate document + move to other modes
        - Insert - “i” key
            - Used to insert text into file
        - Visual - “v” key
            - Highlight text using motions
            - Copy/Cut/Paste
        - Command - start commands with “:”
            - Type commands in Nvim (ex: ‘w’ to save)
    - Reasons to use Neovim
        - Added plugins
        - Personalized, customizable environment
        - “It’s cool” - Thomas McGillan
- HackTheBox
    - Module 3 Using Web Proxies - Write-up in discord
    - Module 4 Information Gathering - Web Edition due next week
- NCAE Minihack Practice - Continued

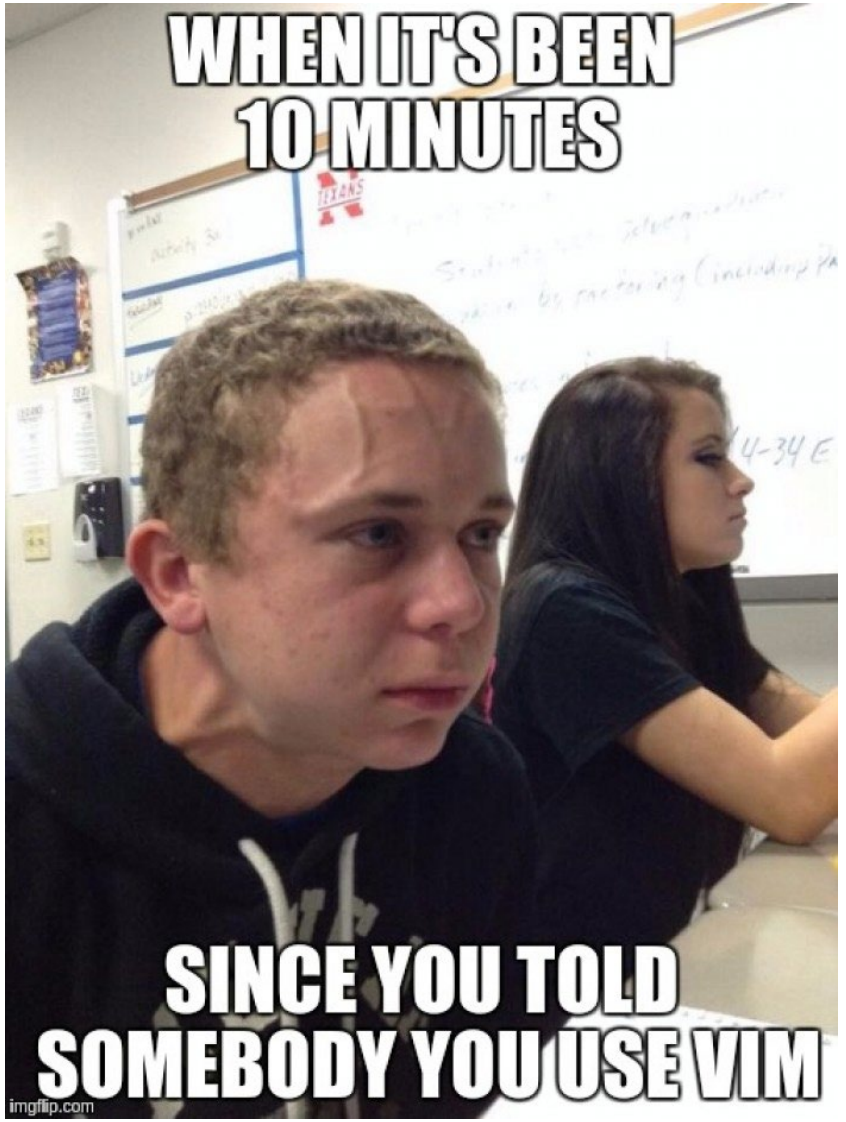
Meeting End: **6:30 PM**

IUP Cybersecurity Club

Meeting 6: 10/08/2024

NEOVIM

HOW & WHY YOU SHOULD USE IT

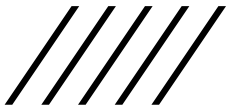


Origins

- Vi or Visual (1976)
  - The Original Editor
  - Terminal Based
    - Lightweight/Minimal
- Vim or Vi IMproved (1991)
  - Improved upon Vi with UI capabilities
  - Introduced new functions
    - Multi-Level Redo/Undo
    - Split-Screen Editing
- Neovim (2014)
  - Wide User Base
  - Open Source = Many Devs
  - Diverse Assortment of Plugins

About Vim/Neovim

- Modal Text Editor
  - Actions are divided into different modes
- Works in the buffer
  - Files are not created until you save the file
- Customizable with Vim Script
  - Create macros
  - Define custom commands
  - Configure Vim’s behavior (Neovim extends functionality w/ Lua)
- Use Vim "motions" to scroll through the doc
  - h j k l (left, down, up, right)



Modes

- Normal
  - The default mode when you open a file with nvim
  - Move to other modes from here, navigate the document
- Insert
  - Accessed by hitting the "i" key in normal mode
  - Used to insert text into a file
    - i.e. Writing code
- Visual
  - Accessed by hitting the "v" key in normal mode
  - Highlight text using motions
  - Copy/Cut/Paste
    - "yank" (y)/"delete" (d)/"put" (p)
- Command
  - Type commands in Nvim (ex. ':w' to save)

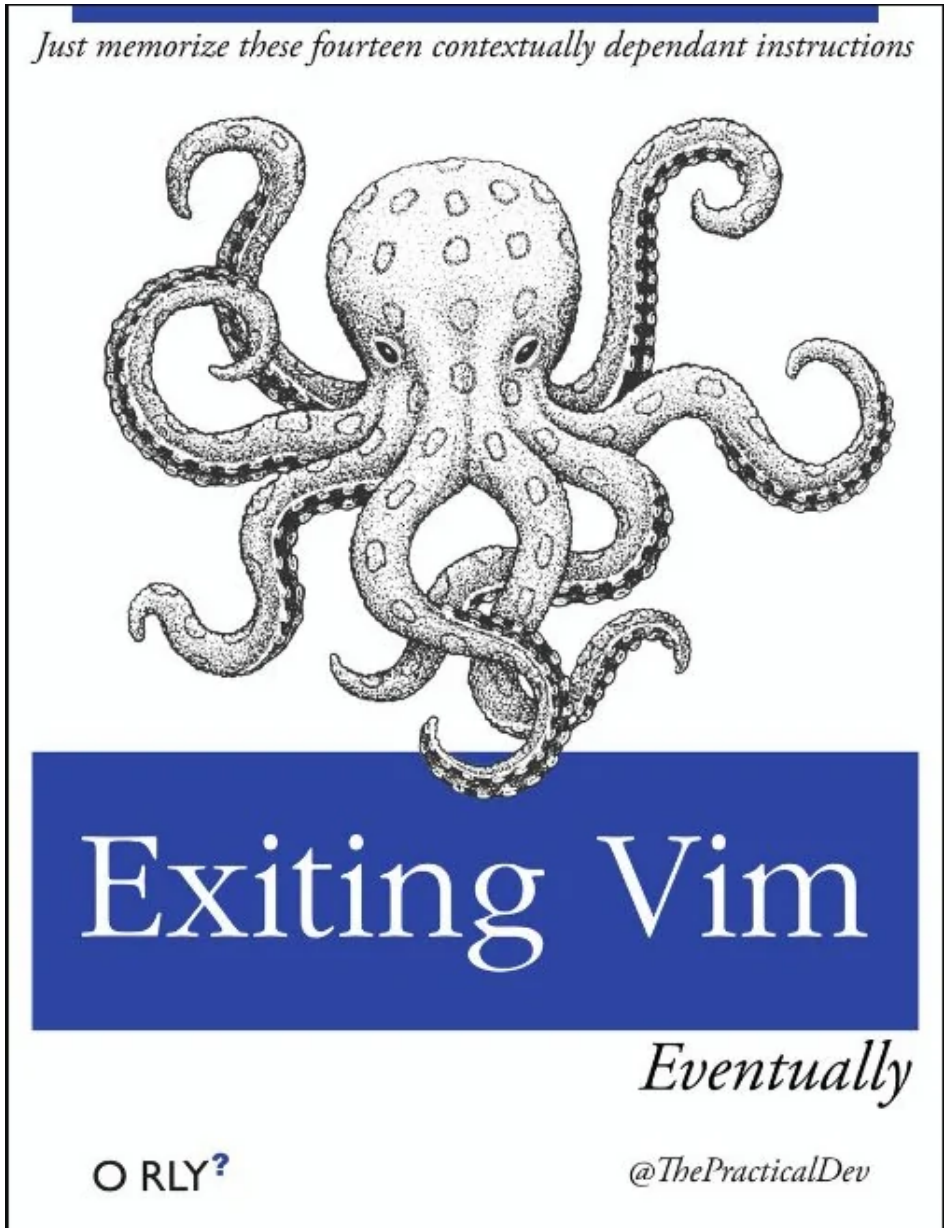
- Each mode can be exited with either Ctrl + c or Esc

Why Neovim?

- Inherits all the good qualities of Vim
- Use a package manager to manage your plugins (like packer)
- Added Plugins
  - Undo Tree – Track changes, Browse/Switch between branches (Vim Plugin)
  - Telescope – Fuzzy Finder, Search files for terms
  - LSPZero – Language server protocol for Nvim
    - Used for code suggestions
  - Vim Fugitive/Rhubarb – Git support for Nvim
  - Harpoon – Quick file toggling
  - Nvim Treesitter – Syntax Highlighting and Code navigation using the tree-sitter parser
- Personalized Development Environment (PDE)
  - Completely customizable
  - Theme, Shortcuts, Aliases, UI, everything
- It's cool

Pop Quiz

- How do you exit Vim?
  - Power off your machine



Meeting Start: **5:32 PM**

- HackTheBox
    - Giving another week to finish Module 3
    - Reach out if you have any questions or need help
- NCAE Minihack
    - Checklist in #Resources channel
    - Full Guide in #Resources channel

Meeting End: **6:32 PM**

IUP Cybersecurity Club

Meeting 5:10/2/2024

OpenPrinting Common Unix Printing System (CUPS) Vulnerability

- Lack of validation/sanitization of network data allows the user to install a malicious driver
- Print job then gets sent to the printer that executes malicious code
- Realistically these CVEs are probably not a large-scale issue
  - CVE chain will not work on systems with default configuration for CUPS
  - Users on a network would have to print to a printer they do not recognize for the RCE to become possible
- However, the CVEs are not currently patched, so it IS a problem

- CVE-2024-47176 - cups-browsed <= 2.0.1 binds on UDP INADDR_ANY:631 trusting any packet from any source to trigger a Get-Printer-Attributes IPP request to an attacker-controlled URL
- CVE-2024-47076 - libcupsfilters <= 2.1b1 cfGetPrinterAttributes5 does not validate or sanitize the IPP attributes returned from an IPP server, providing attacker-controlled data to the rest of the CUPS system
- CVE-2024-47175 - libppd <= 2.1b1 ppdCreatePPDFromIPP2 does not validate or sanitize the IPP attributes when writing them to a temporary PPD file, allowing the injection of attacker-controlled data in the resulting PPD
- CVE-2024-47177 - cups-filters <= 2.0.1 foomatic-rip allows arbitrary command execution via the FoomaticRIPCommandLine PPD parameter

NCAE Minihack

We will be doing the NCAE Minihack today!

- Go to NCAE Sandbox, and launch/deploy the minihack environment
  - https://ui.sandbox.ncaecybergames.org/challenges
- Work with a group of people around you
  - Can use one computer to build the sandbox
  - Will **most likely** have to do research!
- There is a ‘guide’ in the discord #resources channel
- Please ask questions!
  - It will not be “easy” or straightforward
  - Use the guide as a checklist
- We will post a full guide to the solution AFTER the meeting

Things to consider before starting

- Every machine will have different *services*
  - CentOS does not have nano, has vi
    - Using vi is completely different from using nano
- Every OS has a different ‘network’ service to control the network interfaces
  - Ex. Of restarting network service in each OS
    - `sudo systemctl restart network` (CentOS)
    - `sudo systemctl restart netplan` (Ubuntu)
      - ‘sudo netplan apply` ALSO works
    - `sudo systemctl restart networking` (Kali)
- You will have to restart relevant services after applying changes to actually *see* the changes

Meeting Start: **5:31 PM**

- HackTheBox
    - **'Intro to Web Applications' Write Up in Discord**
    - Topics covered:
        - Basic HTML
        - CSS
        - XSS (Cross Site Scripting)
        - Common Web vulnerabilities
    - Next Module due next week (10/2)
        - Using Web Proxies
        - Lots of work - start sooner rather than later
        - Ask for help in Discord if needed
- Using NCAE Minihack to explain…
    - Network Topology
    - Defining / Configuring
        - Services
        - Router
        - Web server
        - Database
        - DNS
        - Backup Servers
    - **Full slideshow in Discord**
- For next week: Minihack exercise
    - Create account on NCAE Cyber Games site: https://ncasecybergames.org
    - Start quick sandbox instance of minihack
    - Familiarize w/ Linux commands
        - How to use Nano/Vi text editors
        - Systemctl, ping, cd, nano, vi, sudo commands
        - Officers available to help
- PicoCTF Challenges
    - Introduce tools/concepts that will be used in CTF portion of NCAE Cyber Games
    - Solve following challenges
        - Commitment Issues
        - First Grep
        - Bases
    - Additional challenges
        - Get aHEAD
        - IntroToBurp
        - LocalAuthority

Meeting End: **6:27 PM**

IUP Cybersecurity Club

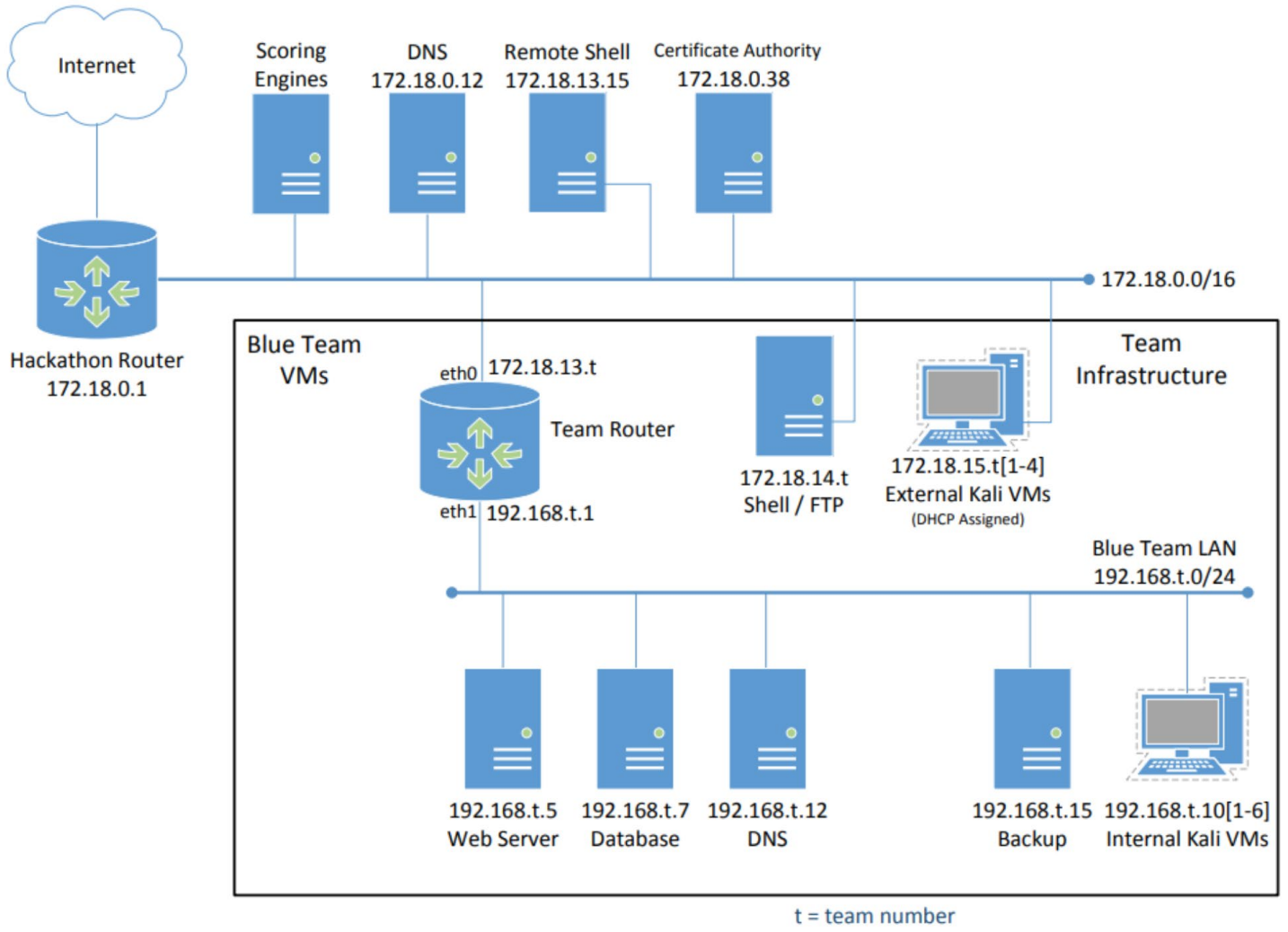
Meeting 3: 9/18/2024

Overview

- We’ll be using the NCAE Minihack to explain the following
  - How to read a Network Topology Map
  - What is/How to configure:
    - Services
    - Router
    - Web Server
    - Database
    - DNS
    - Backup Servers
- **Plus some fun CTFs at the end!**

NCAE Topology Ex.

- Must set up the following
  - Router
  - Web Server
  - Internal/External Clients
    - (Kali machines)
- Various Considerations
  - External access?
  - Layered security
  - Access Control



What are we working with?

- Hardware – The Router/Web Server/Database *machine*
  - The physical machine that these services operate on
- Operating System on the hardware
  - Windows, MacOS, Ubuntu, CentOS, Kali Linux, ParrotOS, etc.
- **Services** within the machine
  - Ex.
    - Apache HTTP Server (httpd) – Web Server
    - MySQL – Relational DB management service
    - OpenSSH (sshd) – Handles remote login and SFTP through SSH
    - DNSMasq – Lightweight DNS forwarder and DHCP server
    - Systemd – System and Service manager for Linux operating systems
  - We will use a shell (terminal) to configure various required services on these machines

Basic Process to Understanding a Service

1. What does the service do?
    1. What is the purpose of the service
    2. How does it **function**
    3. What are other services that require this service?
    4. What other services does this require if any?
2. Where are the ‘important’ files?
    1. Locate the configuration files
        1. Level of configuration matters – By user or system-wide
3. Does the service require a restart?
    1. More often than not services require a restart when configuration files are modified
    2. Is it running by default?
    3. Did the service stopping require the restart of any other services

Ex. Gunicorn on a Web Server

- Gunicorn (Green Unicorn) is a Python WSGI HTTP server
- 1. What does the service do?
  - Handles HTTP requests from clients (web browsers)
  - Dispatches them to the Python application using the WSGI interface
  - Normally used with Django or Flask (Python Web Dev libraries)
- 2. Where are the important files?
  - Log files : /var/log/gunicorn
    - Sometimes /logs
  - Systemd service file
    - /etc/system/system/gunicorn.service
  - Socket file – if using Gunicorn with a socket
    - /run/gunicorn.sock
- Does the service require a restart?
  - Yes – after making config changes or redeploying new version of python application
    - Sudo systemctl restart gunicorn
  - You may also have to restart the web server (nginx, apache, etc.) after restarting gunicorn

Now We’ll Look at the Machines

What are these devices and what services do they use? How are they configured?

Understanding IP addresses and netmasks

- Public/Private IP
  - Public IP:
    - Your public IP is a globally unique ip address assigned to your device by your ISP
    - “regional internet registries” (RIRs) manage allocating the addresses
  - Ex.
    - When you visit google.com, your device connects to google’s **public** ip address
  - Ex. 2
    - Home/Office networks have a router that is assigned a single **public IP** address
      - All devices behind that router share this public ip address when communicating with the rest of the world (google.com for example).
  - Private IP
    - Used within a local network
    - Certain ranges are reserved for local networks only
      - 10.0.0.0 to 10.255.255.255 (Class A)
      - 172.16.0.0 to 172.31.255.255 (Class B)
      - 192.168.0.0 to 192.168.255.255 (Class C)

Understanding IP addresses and netmasks

Common Netmasks and Their CIDR Notation

- 255.0.0.0 or **/8** (Class A):
  - Network: First 8 bits (11111111 00000000 00000000 00000000)
  - Hosts: Last 24 bits
  - Example: 10.0.0.0 with a /8 allows up to 16.7 million host IP addresses on that network.
- 255.255.0.0 or **/16** (Class B):
  - Network: First 16 bits (11111111 11111111 00000000 00000000)
  - Hosts: Last 16 bits
  - Example: 172.16.0.0 with a /16 allows for 65,534 hosts on the network.
- 255.255.255.0 or **/24** (Class C):
  - Network: First 24 bits (11111111 11111111 11111111 00000000)
  - Hosts: Last 8 bits
  - Example: 192.168.1.0 with a /24 allows for 254 hosts on the network.
- 255.255.255.255 or **/32**:
  - Network: All 32 bits (11111111 11111111 11111111 11111111)
  - This is a **single IP** address, often used in routing for specifying a single host.

Understanding IP addresses and netmasks cont.

```
┌──(sandbox㉿kali-sandbox)-[~]
└─$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 172.20.128.100
        netmask 255.255.0.0
```

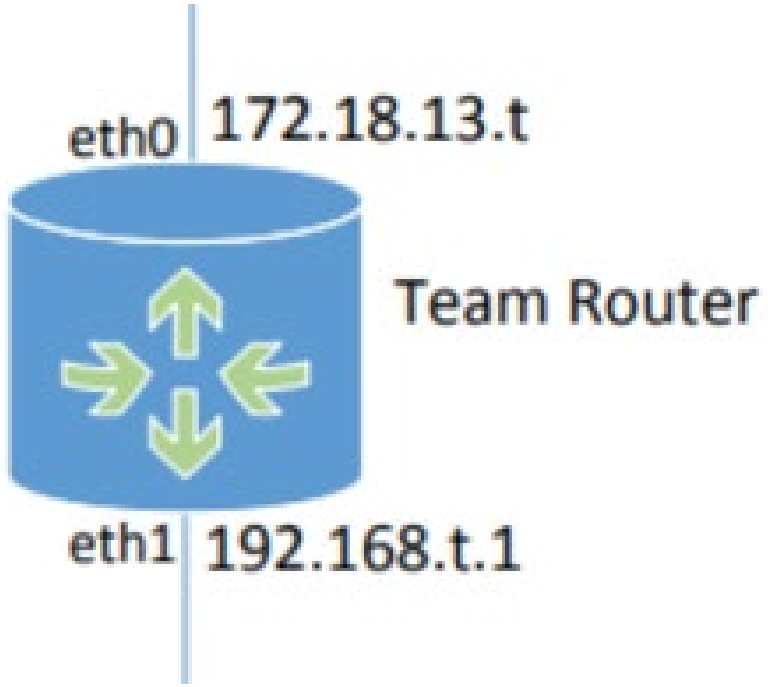
- The “netmask” section tells us that the first 16 bits are the network and the last 16 are the host
- This device will be able to connect to other devices that are 172.20.x.x

Brief on IPv4 vs IPv6

- IPv6 provides an **enormous address space** compared to IPv4:
  - **IPv4** uses a 32-bit address space, offering about 4.3 billion possible IP addresses.
  - **IPv6** uses a 128-bit address space, offering **2^128** (about **340 undecillion** addresses)
    - This effectively eliminates the risk of running out of IP addresses.
- To put it in perspective:
  - IPv4: ~4.3 billion addresses. *2^32*
    - 4,294,967,296
  - IPv6: ~340 undecillion addresses. *2^128*
    - *340,282,366,920,938,463,463,374,607,431,768,211,456*

Router: How does it work?

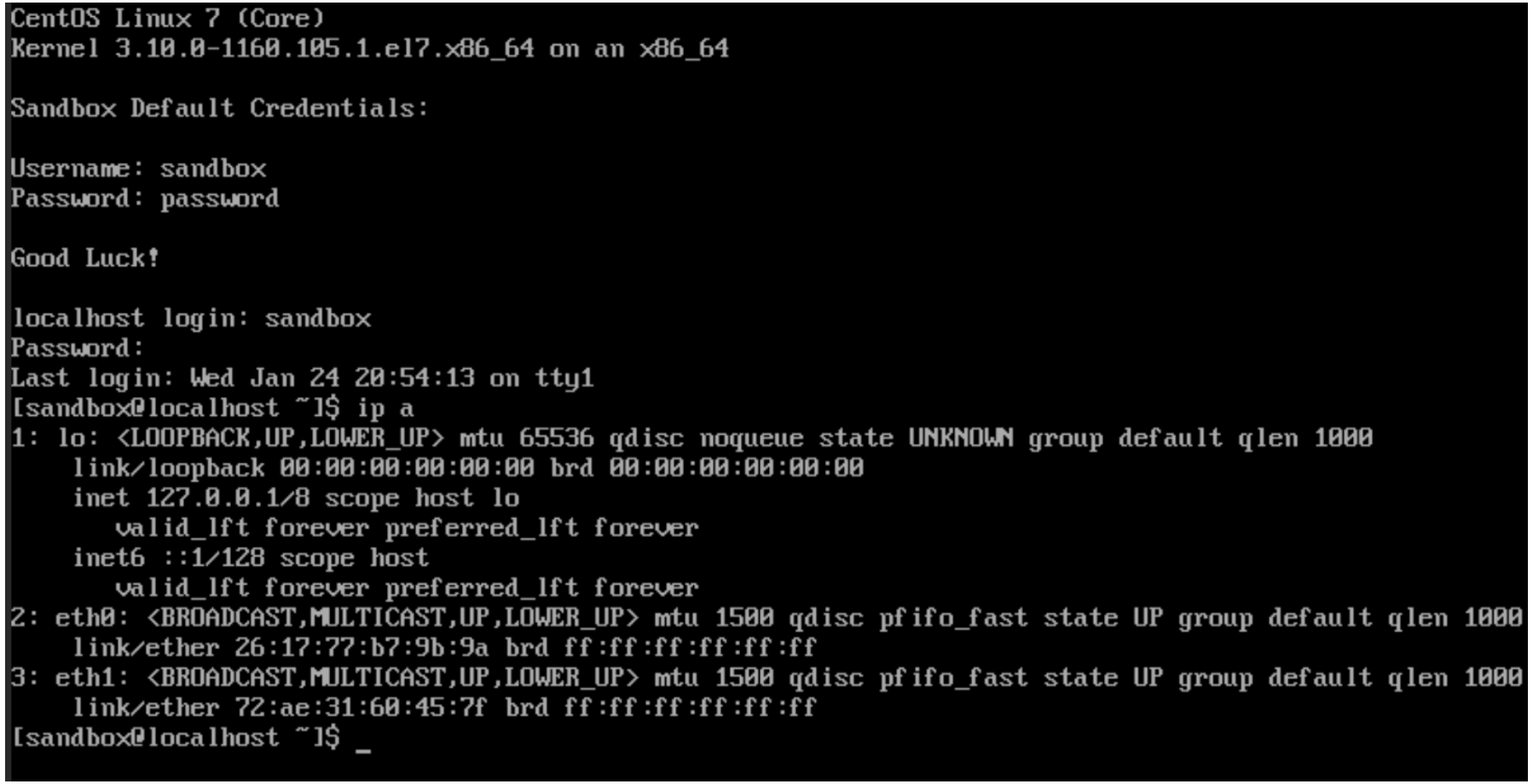
- Routing – forwarding IP packets
- packets – have a header/payload
  - Header contains info about the packet
    - Origin and Destination IP address
  - Payload
    - The actual data
- But how do we configure one?



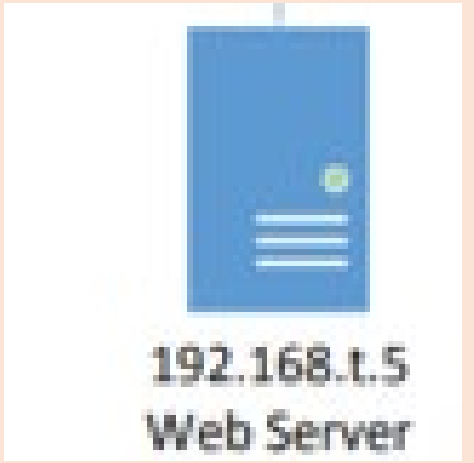
How to configure a Router

- Routers must have two different **network interfaces**
  - “network interfaces” – a way to interact with a network
  - Two are required because each has to be configured differently
  - Traffic has to pass **THROUGH** the router, therefore…
    - It must have an interface to receive external traffic (packets) and send it internally to where it needs to go
    - and one to receive internal traffic (packets) and send it externally to where it needs to go

Cent OS `ip a` example



Web Server: How does it work?



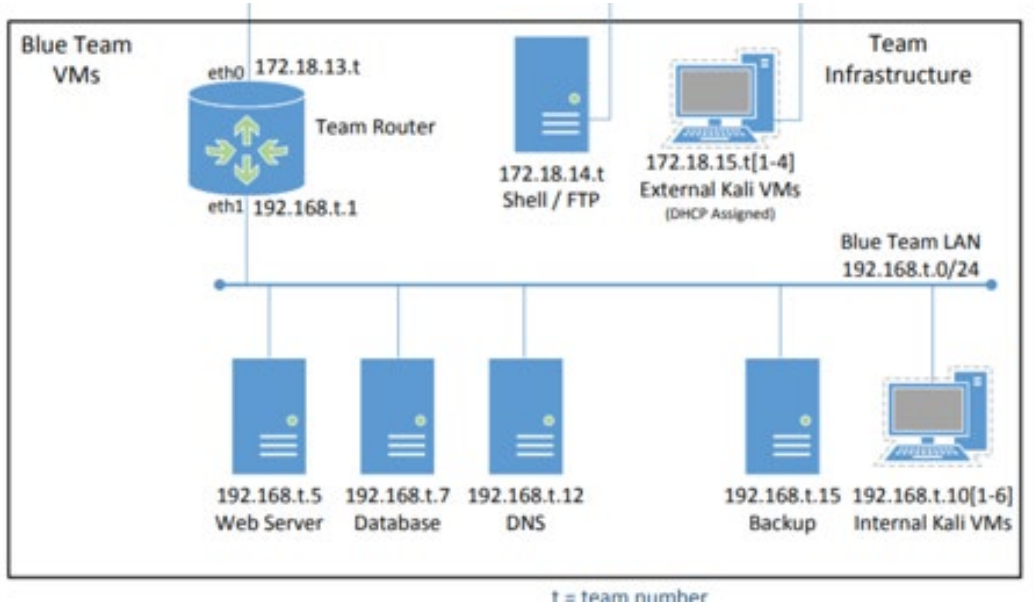
- A web server is anything that can run a web service and have some form of open port for web access
  - Could be sent through a router or exposed to the **internet directly (not recommended)**

How to configure a web server: ex w/ apache2

- Start apache2 service
  - `sudo systemctl enable apache2` (this enables it to start when the machine starts – for future convenience)
  - `sudo systemctl start apache2`
  - Run `systemctl status apache2` and verify it is running
- Navigate to the /var/www/html and view the index.html file
  - This is the static file being served by the apache web server
    - Standard location for the index file in Apache
- Another web server like nginx is very similar in its' setup process

HOWEVER

- Getting a web server to the outside world in this scenario is more complicated
  - Data has to be requested through the router, sent to the web server, and sent back through the router to be displayed on the public side
- It's important to enable firewalls on your router to support this as well– external and internal zones
- Port forwarding would also have to be configured



For next week: a minihack exercise!

- Sign up for an account at the ncae cyber games site
  - https://ncaecybergames.org
- Start up a quick sandbox instance of the minihack
- Read the instructions on how to use it
- Familiarize yourself with some linux commands
  - How to use Nano/Vi text editors
  - How to ‘cd’ across directories
  - systemctl, ping, cd, nano, vi, and of course sudo
- Review this ppt before the meeting for a refresher
- It’s not that bad! We will be here to help out

PicoCTF Challenges

- Work together with the people next to you to try and solve the following challenges:
  - Commitment Issues
  - First Grep
  - Bases
- These challenges will introduce some tools/concepts that will be used in the CTF portion of the NCAE Cyber Games
- If those were too easy, try:
  - GET aHEAD
  - IntroToBurp
  - LocalAuthority
- These challenges will give exposure to the concepts covered in the next HTB module

Meeting Start: **5:30 PM**

- If you haven't gotten an account set up with HackTheBox or ParrotOS downloaded, get in touch with a club officer on discord or after the meeting
- HackTheBox
    - Web Requests Write up can be found in Discord with step-by-step explanations
- Intro to C Programming **(slides posted in Discord)**
    - What is C?
    - Using C
        - 'Hello World' program example
        - Variables
        - Data types
        - Operators
        - Logic
            - Conditional logic (If / Else)
            - Comparison operators
            - Logical operators
        - Loops
            - While loops
            - For loops
            - Break / Continue
        - How to run a C program
            - Compile + Run
    - **Leetcode link, Challenge, and their solutions posted in discord meeting minutes + resources channel**
        - Leetcode 412. Fizz Buzz *(link posted in resources)*
        - Challenge 1
        - Challenge 2
- HackTheBox - Introduction to Web Applications due next week

Meeting End: **6:16 PM**

Intro to C Programming

Meeting 3: 9/18/2024

General Housekeeping

- For those who haven't signed up for the following, please do so:
  - Club Discord
  - HTB Academy
  - Club Crimson Connect
- Another active member survey is posted
  - ONLY FOR THOSE WHO DIDN'T RESPOND TO THE PREVIOUS ONE
- Review of Module 1: Web Requests
  - Quickly review solutions for interactive activities
  - Write up is posted in "htb-write-ups" channel in Discord

Challenges

- For those who haven't programmed before:
  - Ask questions!
    - Everyone knows that you are trying to learn
- For those who have programmed before:
  - Challenge is posted in "resources" channel in Discord
  - See if you can uncover the message!
- For those who have programmed before and have taken/know DSA:
  - Try to write a singly linked list from scratch
    - Hint: this involves the use of structs and void pointers
    - Try without ChatGPT!
  - Requirements:
    - Must be type generic
    - Must have delete and add functionality
      - Ensure handling of edge cases
        - Ex: Proper bounds checking

What is C?

- C is a general-purpose programming language
  - Created by Dennis Ritchie at Bell Laboratories in 1972.
- It is a very popular language, despite being old
  - Fundamental language in the field of computer science.
- C is strongly associated with UNIX
  - Developed to write the UNIX operating system

Statements

- An instruction for the computer to execute
  - Ex: printf(“Hello World”);
    - This will print the text “Hello World” to the terminal without the quotes
  - ALWAYS end statements with a semicolon
  - Can have many statements in a singular .c file

Hello World

```
#include <stdio.h>                  // Header file containing functions for stdin/out


int main() {                        // Function main declaration
        printf("Hello World!");     // Prints "Hello World to terminal
        return 0;                   // Returns 0 for main function, indicating success
}
```

Variables

- Variables are used to store data values
- Variable names must be unique
  - Cannot be DECLARED more than once
- Declared by using a single =
  - Syntax: type variableName = value;
  - Ex:
    - int variable1 = 10;
      - This declares a variable named “variable1” and assigns it a value of 10
- The value variables hold can be changed using another statement
  - Ex:
    - int variable1 = 10;
    - variable1 = 5;
      - Notice that there is not a type declared with this statement
      - Variable reassignment cannot have a type declaration
      - The compiler will think you are trying to declare another variable with the same name

Data Types

- int
  - Used to store integer values
  - Can be 2 to 4 bytes depending on compiler, most likely 4
- float
  - Used for storing decimals up to 6-7 digits
- double
  - Also used for storing decimals, but with more precision
    - Stores 15 decimal digits
- char
  - Stores a single character/letter/number, or ASCII values

Operators

+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	x++
--	Decrement	Decreases the value of a variable by 1	x--

If/Else If/Else Statements

- Used to execute code if a condition(s) are true
- If multiple exclusive checks are needed, use an else if statement
- Else is used as a catch all if none of the set conditions are true
- Format
  - if (condition) {
    statement to be executed if condition is true
    } else if (condtion2) {
    other statement to be executed if condition2 is true
    } else {
    statement that will be executed if none of the conditions are true
    }

Comparison Operators

==	Equal to	x == y	Returns 1 if the values are equal
!=	Not equal	x != y	Returns 1 if the values are not equal
>	Greater than	x > y	Returns 1 if the first value is greater than the second value
<	Less than	x < y	Returns 1 if the first value is less than the second value
>=	Greater than or equal to	x >= y	Returns 1 if the first value is greater than, or equal to, the second value
<=	Less than or equal to	x <= y	Returns 1 if the first value is less than, or equal to, the second value

Logical Operators

&&	AND	x < 5 && x < 10	Returns 1 if both statements are true
||	OR	x < 5 || x < 4	Returns 1 if one of the statements is true
!	NOT	!(x < 5 && x < 10)	Reverse the result, returns 0 if the result is 1

While Loops

- Loops until a certain condition is met
  - Will check condition prior to execution of statements inside of loop
  - Do while loops will execute statements first then check condition
- Format
  - while (condition) {

        statement that will loop while condition is true

    }
  - do {

        statement that will loop while condition is true

    } while (condition)

For Loop

- Loops for a specified number of iterations
- Format
  - for (expression1; expression2; expression3) {
    statement to be executed
    }
  - Expression 1 is executed (one time) before the execution of the code block.
  - Expression 2 defines the condition for executing the code block.
  - Expression 3 is executed (every time) after the code block has been executed.
- Example
  - for (int i = 0; i < 5; i++) {
    printf("%d\n", i);
    }
  - This will print numbers 0 through 4 separated by a newline

Break/Continue

- Important for manipulation of loops
- Break will “break out” of a loop Example
  - for (int i = 0; i < 5; i++) {
    if (i == 2) {
    break;
    }
    printf(“%d\n”, i);
    }
  - This will print numbers 0 through 1 and
- Continue will break a single iteration of a loop
  - for (int i = 0; i < 5; i++) {
    if (i == 2) {
    continue;
    }
    printf(“%d\n”, i);
    }
  - This will print numbers 0 through 4, but it will skip the number 2

How to Run a C Program

- Compile your C file
  - gcc nameOfFile.c –o nameOfProgram
- Run your C file
  - ./nameOfProgram

Challenge: Leetcode 412. Fizz Buzz

- Given an integer n, print answer(s) where:
  - answer[i] == "FizzBuzz" if i is divisible by 3 and 5.
  - answer[i] == "Fizz" if i is divisible by 3.
  - answer[i] == "Buzz" if i is divisible by 5.
  - answer[i] == i (as a string) if none of the above conditions are true.
- Example 1:
  - Input: n = 3
  - Output: 1, 2, Fizz
- Example 2:
  - Input: n = 5
  - Output: 1, 2, Fizz, 4, Buzz
- Example 3:
  - Input: n = 15
  - Output: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz

```c
#include <stdio.h>

int main() {
    int n = 15;

    for (int i = 1; i <= n; i++) {
        if (i % 3 == 0 && i % 5 == 0) {
            printf("FizzBuzz");
        } else if (i % 5 == 0) {
            printf("Buzz");
        } else if (i % 3 == 0) {
            printf("Fizz");
        } else {
            printf("%d", i);
        }

        if (i != 15) {
            printf(", ");
        } else {
            printf("\n");
        }
    }
}
```

Fizz Buzz Solution

Optional Challenges

- Challenge 1: Write a program to reverse an input string
  - Ex.
    - In: Cyber
    - Out: rebyC
- Challenge 2: Write a program to return the number of 1's in binary of a given string's characters
  - Ex.
    - In: CyberClub
    - Out: 34
  - Convert all characters to binary, then parse out the "1's" and add to the total
  - Finally, print the total number of 1's
    - *Hint* Every Ascii letter is going to be 8 bits in length. Use a for loop to bitwise shift through each bit of each individual bits of a character

```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];  // Array to store the input string
    int length;  // Variable to store the length of the string

    // Ask the user to enter a string
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove newline "\n" character if fgets adds it
    str[strcspn(str, "\n")] = 0;

    // Get length of the string
    length = strlen(str);

    //Print the string in reverse order using a for loop
    printf("Reversed string: ");
    for (int i = length - 1; i >= 0; i--) {
        putchar(str[i]);
    }
    printf("\n");

    return 0;
}
```

Challenge 1 Solution

```c
#include <stdio.h>
#include <string.h>

// Function to print binary representation of the ASCII value and count 1's
int countAsciiOnes(char ch) {
    int count = 0;
    int ascii_value = (int) ch;  // Convert the character to its ASCII value
    printf("Binary representation of '%c': ", ch);

    // Print and count the binary representation (8 bits)
    for (int i = 7; i >= 0; i--) {
        int bit = (ascii_value >> i) & 1;  // Get the i-th bit (from left to right)
        printf("%d", bit);  // Print the bit
        if (bit == 1) {
            count++;  // Count the 1's
        }
    }

    printf(" (Number of 1's: %d)\n", count);  // Show the count of 1's

    return count;
}

int main() {
    char str[100];  // Array to store the input string
    int total_sum = 0;

    // Ask the user to enter a string
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove newline character if fgets adds it
    str[strcspn(str, "\n")] = 0;

    // Iterate over each character in the string
    for (int i = 0; str[i] != '\0'; i++) {
        total_sum += countAsciiOnes(str[i]);  // Add the number of 1's for each character
    }

    // Output the total sum of 1 bits
    printf("\nTotal sum of 1 bits in the string: %d\n", total_sum);

    return 0;
}
```

Challenge 2 Solution

```
evan@Tablet:/mnt/c/Users/ecroo/School/cyber-club$ gcc string-add.c -o string-add
evan@Tablet:/mnt/c/Users/ecroo/School/cyber-club$ ./string-add
Enter a string: CyberClub
Binary representation of 'C': 01000011 (Number of 1's: 3)
Binary representation of 'y': 01111001 (Number of 1's: 5)
Binary representation of 'b': 01100010 (Number of 1's: 3)
Binary representation of 'e': 01100101 (Number of 1's: 4)
Binary representation of 'r': 01110010 (Number of 1's: 4)
Binary representation of 'C': 01000011 (Number of 1's: 3)
Binary representation of 'l': 01101100 (Number of 1's: 4)
Binary representation of 'u': 01110101 (Number of 1's: 5)
Binary representation of 'b': 01100010 (Number of 1's: 3)

Total sum of 1 bits in the string: 34
```

Challenge 2 Output

Meeting Start: 5:30 PM

- National Public Data Data breach
    - 270 Million Americans affected
    - Check if you're affected: https://npd.pentester.com/
- Using Linux Presentation
    - Getting Started
        - Installing + setting up Parrot OS VM
        - Differences between Windows and Linux
        - Linux File Permissions
        - Why use Linux?
    - Navigation and Essentials
        - Installing WSL / Configuring User Accounts
        - Changing user passwords
        - Basic navigation and file structure
        - Common Commands
            - Server Usage Examples
        - Basic Cyber Resiliency
    - Practice
        - Bandit Labs
        - Hack The Box - Linux Fundamentals
    - Bandit Labs 0-5
        - Description
        - Solutions

Meeting End: 6:30 PM

IUP Cybersecurity Club

Meeting 2: 9/11/2024

National Public Data

- Who are they/what do they do?
  - Data broker company that performs employee background checks
  - Primary service is collecting information from public data sources, including criminal records, addresses, and employment history, and offering that information for sale
- Data breach first occurred in April
  - Affected over 270 million Americans
  - Includes names, addresses, and social security numbers
- Does this affect me?
  - Probably…
  - https://npd.pentester.com/
- What to do?
  - Freeze your credit at the three major credit reporting agencies
    - https://www.experian.com/freeze/center.html
    - https://www.equifax.com/personal/credit-report-services/credit-freeze/
    - https://www.transunion.com/credit-freeze

Using Linux

Basic Commands and Server Navigation

Installing Parrot OS VM

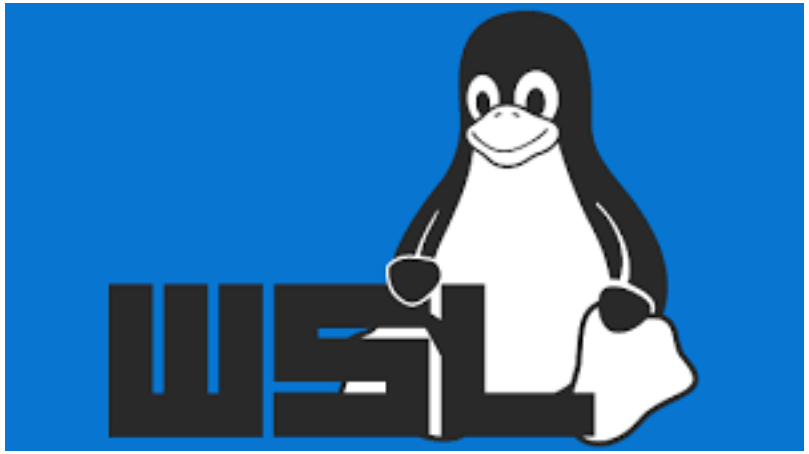
- Open VMware Workstation Pro
- File > New Virtual Machine
- Select “Typical” install
- Select “Installer disc image file (iso)”
  - Browse to Parrot OS iso
- For Guest operating system, select “Linux”
  - Version: “Other Linux 6.x kernel 64-bit”
- Choose a name and location of VM
  - Ex: “C:\Users\<Your User>\.vmware\Parrot OS VM”
- Choose disk capacity (16 GB or more)

Installing Parrot OS VM (cont.)

- Click “Customize Hardware…”
  - Change memory allocation to at least 6 GB (for install, can be tweaked later)
  - Change processors to have 2 cores per processor (if possible)
- Click “Finish”
- Power on VM
  - Select “Try/Install”
- Do these next steps AFTER club
- Once at Desktop, click “Install Parrot” icon on Desktop
  - Make sure all Time zones/Layouts are correct
  - Once in “Partitions” section, select “Erase disk” option
  - Create User account
  - Click “Install” (will take a while)
- Once done installing, restart VM
  - If prompted after reboot, install updates

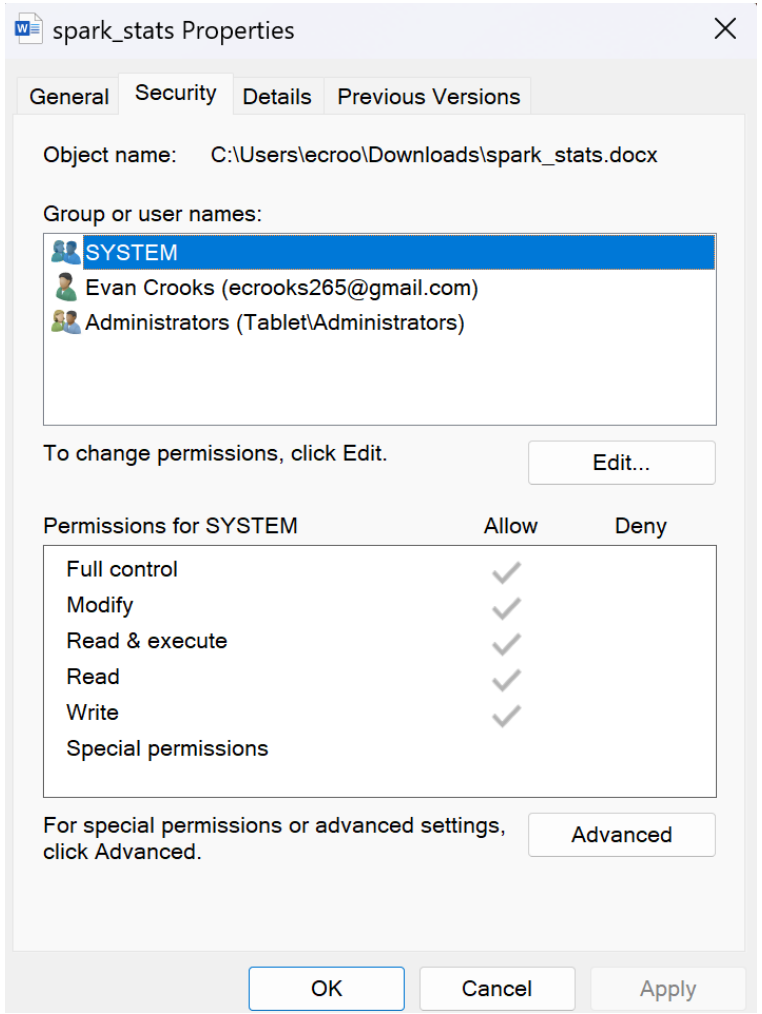
Installing WSL/Configuring User Accounts (Optional)

- Install WSL
  - In Powershell `wsl –install`
  - follow prompts to create initial user
- Launch wsl with `wsl` in powershell
- Add User
  - `sudo su` (switches to root user)
    - `sudo su – username` to change to selected user
- Assign User to Group
  - `usermod -aG sudo username` (adds ‘username’ user to sudo user group)



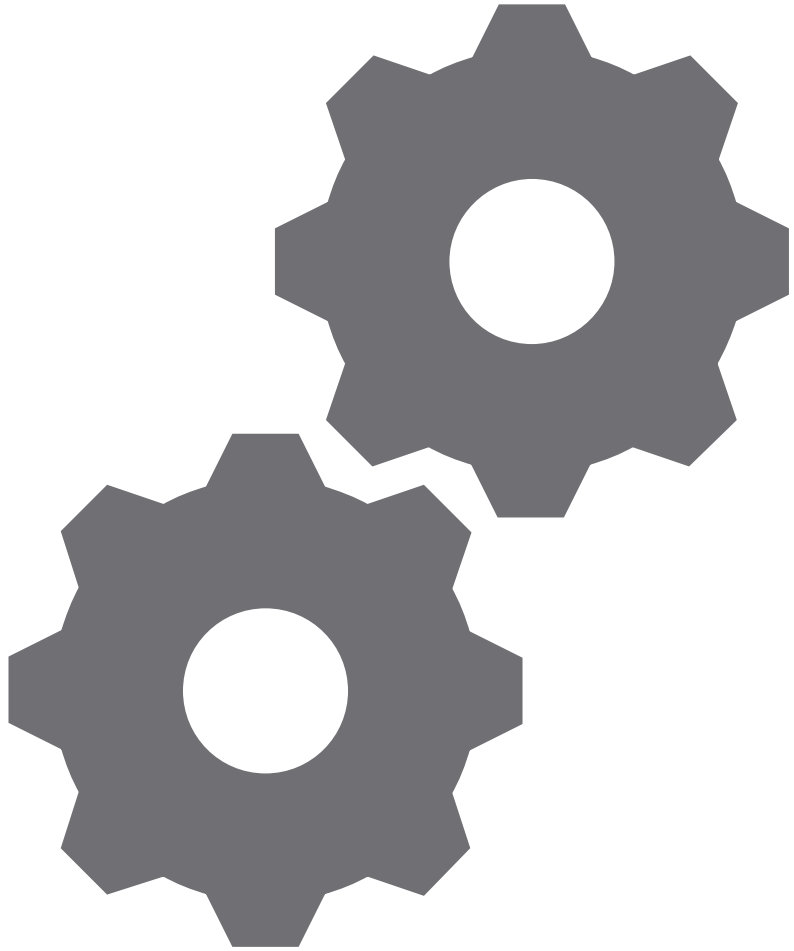
Windows and Linux differences

- File Access
  - Windows has:
    - Read, Write, Execute, Delete, Modify, Full Control, Special Permissions
  - Linux has:
    - Read, Write, Execute
- User Groups
  - Windows has:
    - Users and Groups
    - Permissions are inherited from parent folders
  - Linux has:
    - Users, Groups, and Others
    - Does NOT have permission inheritance



Linux File Permissions

- Read
  - Files: Grants the ability to view or read the contents of the file
    - If user has read permission on a file, they can open it and see its data.
  - Directories: Allows the user to list the contents of the directory
    - Does not allow opening the files inside unless the user also has execute permission on the directory
- Write
  - Files: Grants the ability to modify, alter, or delete the file's contents
    - If a user has write permission on a file, they can edit and save changes to it
  - Directories: Allows the user to create, delete, or rename files within the directory
    - To perform these actions, execute permission on the directory is also usually needed
- Execute
  - Files: Grants the ability to run or execute the file as a program or script.
    - For running executable files or scripts that contain code
  - Directories: Allows the user to cd into the directory and access its files and subdirectories
    - Without execute permission, even if a user can read the directory's contents, they cannot navigate into it or access its files directly



So Why Use Linux?

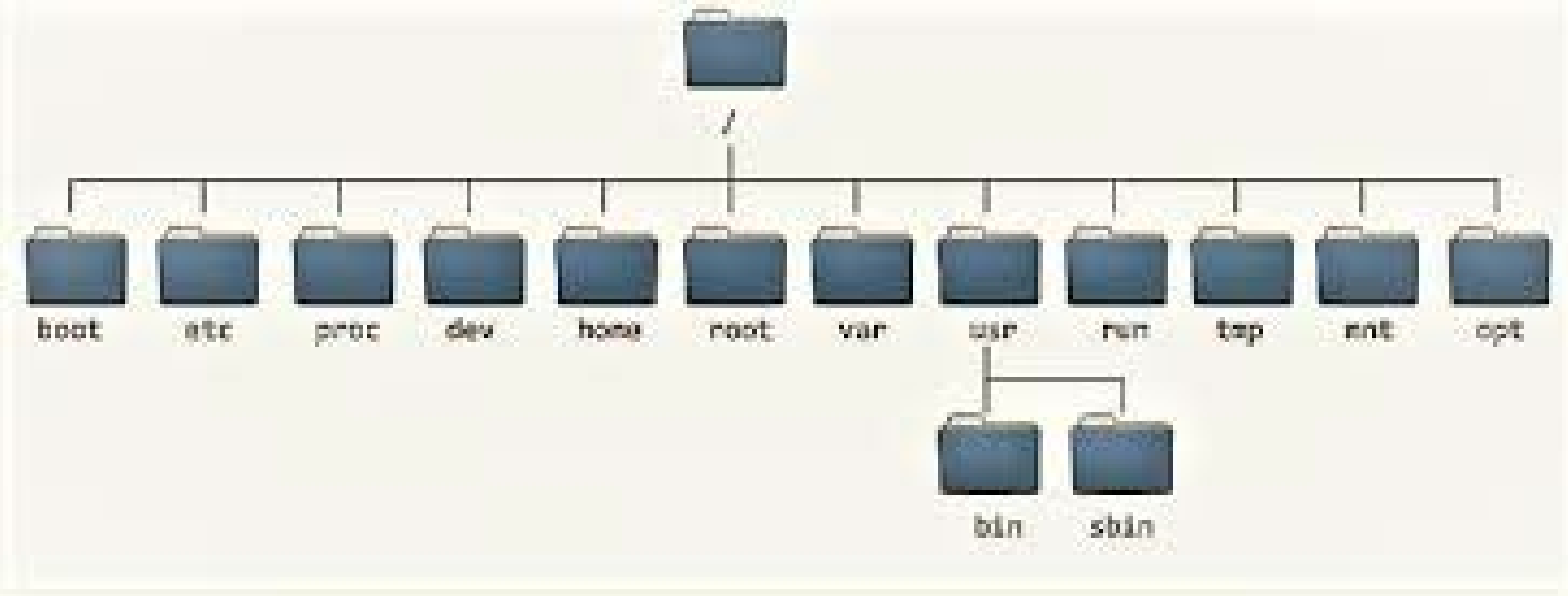
- Reliability
  - Uptime is typically higher in unix-based systems
    - Not as many updates, required reboots
    - Typically well-optimized systems
- Cost
  - Often free and open source tools
- Security
  - File permissions allow for better security
  - Drawback is open source is also open to bad actors

Changing User Passwords in Linux

`sudo passwd username` to change password of user ‘username’

Important not to use default passwords EVER

Show all users on a server - `cat /etc/passwd`



Basic Navigation and File Structure

- Cd – change directory
  - `cd ..` go up a directory
  - `cd ./mnt/c` open the mnt/c directory from your current location
- Pwd – print working directory
- **Use `man command_name` to see options/description of a command**

Common Commands

cd – change directory

mkdir – make directory

pwd – print working directory

ls – list directory

touch – changes 'last modified' timestamp (creates file if not exist)

cat – list contents of specified file

man – lists manual for given command

rm – remove file or directory

cp – copy file/directory from >> to

head/tail – prints top or bottom part of file

chmod – changes file permissions or directory permissions

chown – change owner of a file

chgrp – change group owner for file

tar/unzip – zip files and unzip files

ps – display actively running processes

grep – search plaintext using regex

Example Usage on Servers

- crontab – regulate backups, scans, logging, etc.
- nano/vi – edit config files on a server, write scripts, etc.
- firewall-cmd & iptables – configure firewall options on server
- Nginx & Apache – common web server tools
- Ping, nslookup, dig – resolve various DNS issues

Cyber Resiliency—What to Do

- Backdoors
  - SSH keys
    - Check .ssh directory for keys that look suspicious – remove them
    - Common SSH Exploitations
  - Crontab
    - Check the crontab for unknown scripts/suspicious scripts
    - Check the scripts that you DO know as well
      - Could be modified
  - Unknown Users/Unauthorized Users in User Groups
    - Can be used as backdoors
      - Users with elevated privileges will have access to more within the system
    - Would have to change passwords, check last login status, last commands ran etc.
      - Ideally would have logs being created and saved to track this

Practice

- Bandit Labs
- Hack The Box – Linux Fundamentals

```
SSH(1)                          BSD General Commands Manual                          SSH(1)

NAME
     ssh – OpenSSH remote login client

SYNOPSIS
     ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
         [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file] [-J destination] [-L address]
         [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address]
         [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]] destination [command [argument ...]]

DESCRIPTION
     ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine.
     It is intended to provide secure encrypted communications between two untrusted hosts over an insecure network.
     X11 connections, arbitrary TCP ports and UNIX-domain sockets can also be forwarded over the secure channel.

     ssh connects and logs into the specified destination, which may be specified as either [user@]hostname or a URI
     of the form ssh://[user@]hostname[:port].  The user must prove their identity to the remote machine using one of
     several methods (see below).

     If a command is specified, it will be executed on the remote host instead of a login shell.  A complete command
     line may be specified as command, or it may have additional arguments.  If supplied, the arguments will be ap-
     pended to the command, separated by spaces, before it is sent to the server to be executed.

     The options are as follows:

     -4      Forces ssh to use IPv4 addresses only.
```

Bandit Labs 0-5

- ssh bandit0@bandit.labs.overthewire.org -p 2220
  - After finding the flag for each level, you can ssh into the next one by running the above command, but incrementing the user
    - So level 1 would look like this
      - ssh bandit1@bandit.labs.overthewire.org -p 2220
- Remember to use the man command for commands you are unfamiliar with
  - Ex. `man ssh` will tell you use cases for the ssh command
- Read the description of the bandit problem on the overthewire site before attempting

Bandit 0-5 Solutions

- Bandit 0 – readme file
  - Must output contents of readme file in home directory
- Bandit 1 – '-' file
  - Output contents of '-' file
    - Run the command `cat './-'`
- Bandit 2 – 'spaces in this filename'
  - Output contents of the 'spaces in this filename' file
    - Run the command `cat './spaces in this filename'`
- Bandit 3 – find hidden file
  - `cd inhere`
  - `ls –al`
  - `cat …hiding-from-you`

Bandit 0-5 solutions continued

- Bandit 4
  - `cd inhere`
  - `ls`
  - cat out contents of each file until you find a plaintext password in one of the files
    - `cat ‘./-file00’` etc…
  - could also use the strings command
    - Will only output strings of plaintext
- Bandit 5
  - Search inhere directory ` find ./ -size 1033c`
    - Can also specify –f to only search files
    - 1033c is the specified size, 1033 is the size, and c specifies that it is 1033 bytes
    - Only one file is returned
  - ‘cat './maybehere07/.file2’`

Meeting Start: 5:31 PM

- Opening by Dr. Waleed Farag
    - DoD CSA Scholarship
        - https://www.iup.edu/cybersecurity/grants/dod-cyber-scholarship-program/index.html
        - Benefits
            - Full cost of tuition and fees paid
            - $29,000 stipend to cover room and board
            - Cost of books
            - One-time laptop / computer purchase
        - Requirements
            - US citizen
            - Maintained GPA of 3.2
            - Sophomore / Junior when applying
    - IUP Cybersecurity news / future updates
        - https://www.iup.edu/cybersecurity/news/index.html
        - IUP Cybersecurity Day
- Overview of Cyber Club
    - Weekly meetings Wednesdays @ 5:30pm
    - Discord: https://discord.gg/dXQwuYKJeU
- Topics for the semester
    - NCAE Cyber Games Prep
        - Linux navigation, tools, command-line-interface, etc
        - Networking
        - Server management
    - Learn to use other various software / develop skills
        - Work towards HTB Certified Bug Bounty Hunter Certificate
- Labs and Resources
    - Hack the Box
    - Over the Wire
    - NCAE sandbox labs
    - PICO CTF (Capture the Flag) activities

Meeting End: 6:02 PM

IUP Cybersecurity Club

Meeting 1: 9/2/2024

DoD Cyber Service Academy

- Recruit students to work for DoD agencies upon graduation
- Basic Requirements
  - Entering junior or senior year
  - Maintain cumulative 3.2 GPA
  - Must be US citizen
  - Agree to work for one year for each year of scholarship received
    - Summer internships may also be required
- Benefits
  - Full cost of tuition and all fees
  - $29,000 stipend for undergrad students to cover room and board
  - All required books
  - One-time laptop/computer purchase
- Additional information:
  - https://www.iup.edu/cybersecurity/grants/dod-cyber-scholarship-program/index.html

About Cyber Club

- IUP Cyber Club is a student-led organization
- We hold weekly meetings Wednesdays @ 5:30pm
- Engaging Activities
- Practical Experience
- Competitions (NCAE Cyber Games, CMU's Pico CTF)
- Community/Networking

Officers

- President – Thomas McGillan
- Vice President – Evan Crooks
- Secretary – Dylan Timbrook
- Treasurer – Benjamin Baker

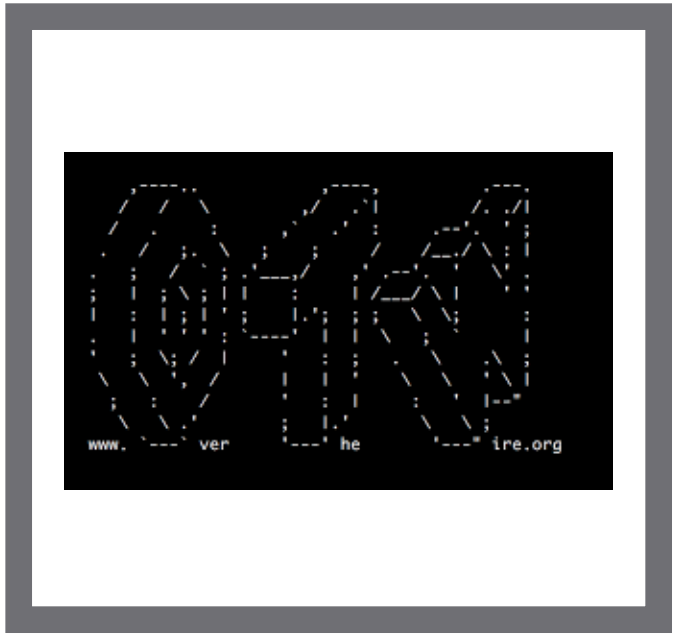
- Join the discord: https://discord.gg/dXQwuYKJeU
  - Meeting Minutes, Resources, Community, etc.

Topics

- Something for everyone!
- Preparing for the NCAE Cyber Games
  - Basic Linux navigation/scripting, command-line-interface
  - Networking
  - Server Management/Configuration (Web Servers, DNS, etc.)
  - Using various Linux tools for many different things
    - firewall-cmd, nslookup, iptables, netcat, etc.
  - Practical cybersecurity on Linux servers
- Learn to use various software/develop skills
  - Git, Cloud Platforms, Wireshark, etc.
  - Earn a certificate (HTB Certified Bug Bounty Hunter)

Labs and Resources

- Many of our meetings will include hands-on labs
  - Group exercises and guidance for solo enrichment
  - Plenty of resources to use to increase your knowledge on your own
- Hack the Box, Over the Wire, NCAE sandbox labs, PICO CTF activities



Meetings

- Collaborative Meetings
- Including:
  - Presentations
  - Labs/Activities
  - Projects
  - Discussion
  - Troubleshooting/Debugging (use us as resources!)
- Held every Wednesday in Stright 112A @ 5:30pm

What do YOU want out of Club?