SOC ANALYST | PROJECT: CHECKER

שם הסטודנט: אלעד יהושע
שם הסקריפט: HMagen773629.s13.NX220.sh
קוד סטודנט: S13
קוד תוכנית: NX220

Introduction

The script runs a network scan (nmap), shows discovered IPs to the user, offers 3 built-in attacks (Brute Force, MITM, DoS) plus a random option, supports manual/selection/random target choices, executes the chosen attack, and logs every event to /var/log/attack_log.txt. It also performs dependency checks, cleanup, and supports testability (DEFAULT_TARGET, TEST_RANDOM_SEED, /tmp/soc_chosen).

Step 1: Permissions and preparation (general requirement)

What runs: At the beginning of main(), the script calls check_tool() to verify that all dependencies (nmap, hydra, hping3, arpspoof, etc.) are installed, and enforces running as root.
How it meets the requirement: Ensures required tools are available before running attacks, prevents unexpected failures, and enforces proper privileges.

Step 2: Network interface selection (requirement: display IPs)

What runs: choose_network_interface() lists active interfaces (ip -o -4 addr show) and their IPs; the user selects by number. Variables chosen_iface and ntip are saved.
How it meets the requirement: Fulfills **2.1 Display all available IP addresses** by showing active IPs to define the scanning scope.

Step 3: Scan type selection and running nmap (requirement: scanning & host discovery)

What runs: main() asks for scan type (1=Fast, 2=Full, 3=Vuln) and calls run_nmap_scan(scan_type). This function runs nmap with appropriate options, saves outputs in /var/log/nmap_scans/scan-..., and extracts hosts from .gnmap into .hosts.txt and DETECTED_HOSTS.
How it meets the requirement: Detects and displays all discovered IPs (requirement 2.1) and saves scan outputs under NMAP_DIR.

Step 4: Attack list and descriptions (requirements: attacks + description)

What runs: attack_menu() shows four options: 1) Brute Force, 2) MITM, 3) DoS, 4) Random. Each attack function begins with show_attack_description(title, desc) to print a clear explanation.
How it meets the requirement: Implements three distinct attack functions (1.1) and shows a descriptive explanation before execution (1.2).

Step 5: Attack choice or random (requirement: specific/random choice)

What runs: The user selects 1- 4. If option 4 (random) is chosen:

- If no DETECTED_HOSTS exist, random defaults to Brute Force.
- Otherwise, a random number selects one of the three attacks.
  The script also supports TEST_RANDOM_SEED and writes the selection to /tmp/soc_chosen for automated testing.
  How it meets the requirement: Supports both specific and random selection (2.3), and provides deterministic testing support.

Step 6: Target selection (requirement: 2.5)

What runs: choose_target() allows:

1. Manual IP entry,
2. Selection from detected list by number,
3. Randomly chosen detected host.
   If no hosts were detected and DEFAULT_TARGET is defined, that target is used automatically.
   How it meets the requirement: Fully implements requirement **2.5**, supporting manual, listed, and random target selection, plus auto-target for testing.

Step 7: Attack execution (requirement: 1.1)

What runs:

- **Brute Force:** Builds a temporary user list, asks for service/port (via ask_port_default()), runs hydra, captures success lines, and logs BruteForce-SUCCESS.
- **MITM:** Detects gateway, asks for confirmation, enables IP forwarding, runs two arpspoof processes (target→gateway, gateway→target), and stops on Enter.

- **DoS:** Prompts for port, asks for confirmation, requests duration, and runs hping3 with timeout to ensure a limited run.
  How it meets the requirement: Each attack is implemented as a separate function and shows a description before execution (requirement 1.1/1.2).

Step 8: Logging (requirements 3.1 / 3.2)

What runs: log_attack(type, target, info) is called at key stages: attack start, success, end, and when aborted or failed. Entries include timestamp, attack type, target, and details, and are written to /var/log/attack_log.txt.
How it meets the requirement: Logs all attack attempts and results, fulfilling requirements **3.1/3.2**.

Step 9: Cleanup and background processes (general requirement)

What runs: cleanup() kills background processes (BG_PIDS), disables IP forwarding, and is bound to trap EXIT to ensure it always runs.
How it meets the requirement: Guarantees system state is restored and background processes are terminated after execution.

Step 10: Security, comments, and documentation (general requirement)

What runs: Student header with name/code/lecturer at the top, root check, warnings for lab-only use, and explanatory comments throughout the code.
How it meets the requirement: Provides proper documentation, student identification, and warnings, ensuring clarity and compliance.

Step 11: Testability (optional, not required)

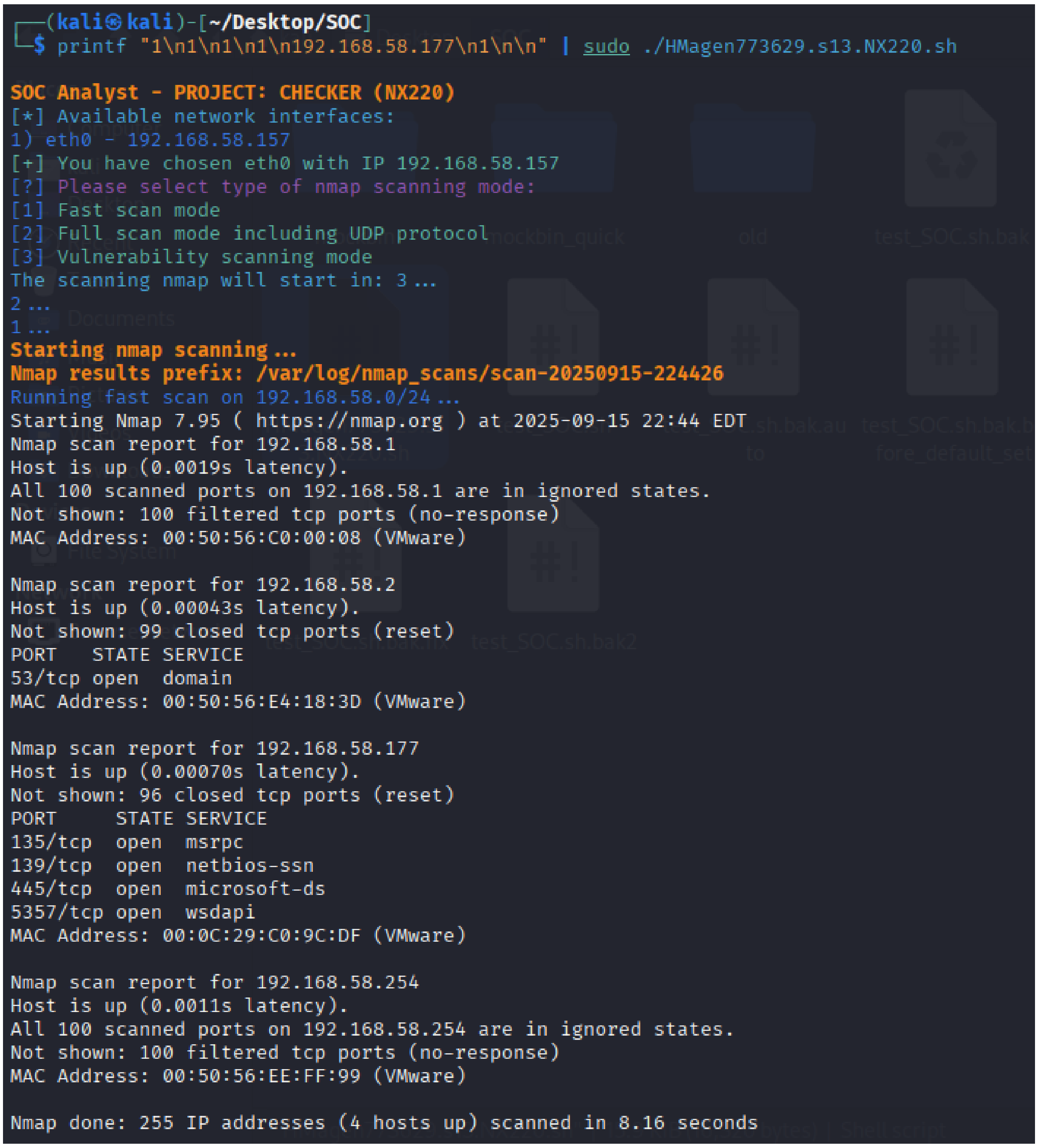
What runs: Supports TEST_RANDOM_SEED for deterministic random choices, writes selection to /tmp/soc_chosen, and allows DEFAULT_TARGET for headless testing. A separate test_SOC.sh script with mocks was created for automated checks.
How it meets the requirement: Not required by the project, but allows instructors to test the script safely and reproducibly.

Summary - Requirements mapping

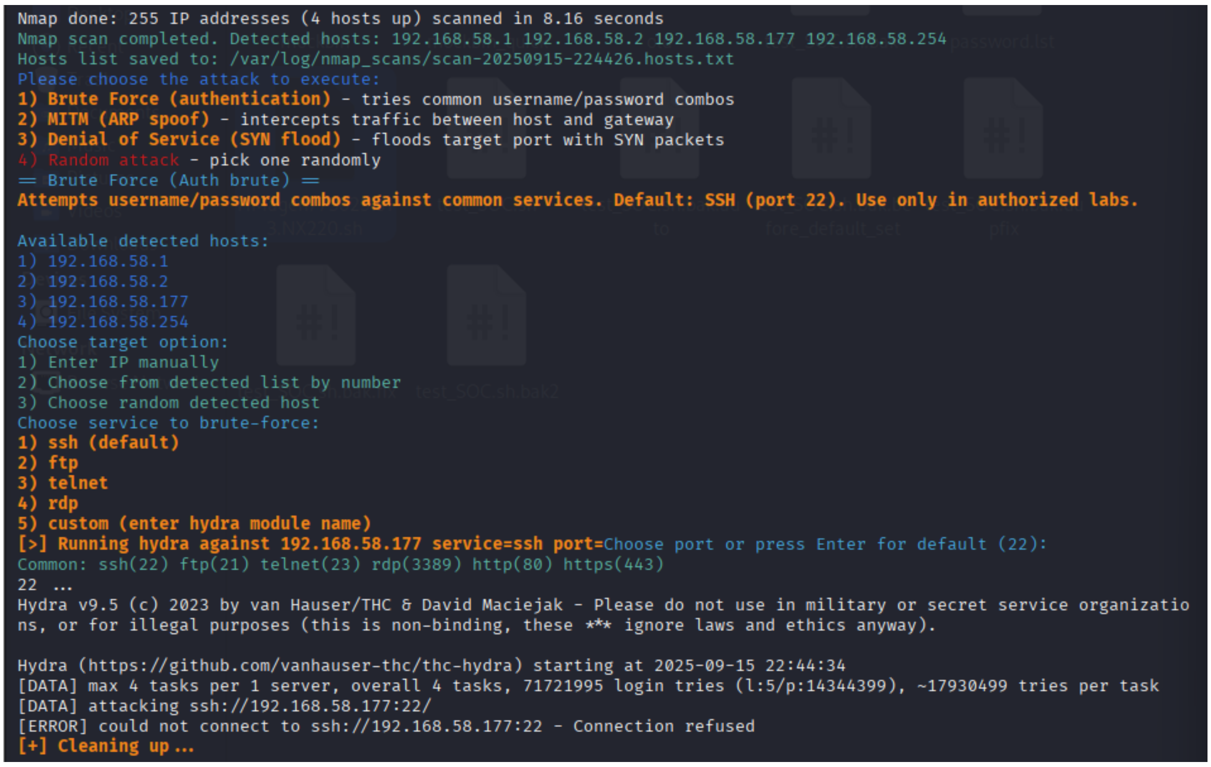
- **Req 1.1 / 1.2:** Implemented - three attack functions with descriptions.
- **Req 2.1–2.5:** Implemented - IP display, attack menu, manual/random choices, input validation, and target selection.

- **Req 3.1 / 3.2:** Implemented - attack logging with type, timestamp, target, and info.
- **General requirements:** Implemented - functions, documentation, student header, cleanup.

Execution example:



```
┌──(kali㉿kali)-[~/Desktop/SOC]
└─$ printf "1\n1\n1\n1\n192.168.58.177\n1\n\n" | sudo ./HMagen773629.s13.NX220.sh

SOC Analyst - PROJECT: CHECKER (NX220)
[*] Available network interfaces:
1) eth0 - 192.168.58.157
[+] You have chosen eth0 with IP 192.168.58.157
[?] Please select type of nmap scanning mode:
[1] Fast scan mode
[2] Full scan mode including UDP protocol
[3] Vulnerability scanning mode
The scanning nmap will start in: 3 ...
2 ...
1 ...
Starting nmap scanning ...
Nmap results prefix: /var/log/nmap_scans/scan-20250915-224426
Running fast scan on 192.168.58.0/24 ...
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-15 22:44 EDT
Nmap scan report for 192.168.58.1
Host is up (0.0019s latency).
All 100 scanned ports on 192.168.58.1 are in ignored states.
Not shown: 100 filtered tcp ports (no-response)
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap scan report for 192.168.58.2
Host is up (0.00043s latency).
Not shown: 99 closed tcp ports (reset)
PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 00:50:56:E4:18:3D (VMware)

Nmap scan report for 192.168.58.177
Host is up (0.00070s latency).
Not shown: 96 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
5357/tcp open  wsdapi
MAC Address: 00:0C:29:C0:9C:DF (VMware)

Nmap scan report for 192.168.58.254
Host is up (0.0011s latency).
All 100 scanned ports on 192.168.58.254 are in ignored states.
Not shown: 100 filtered tcp ports (no-response)
MAC Address: 00:50:56:EE:FF:99 (VMware)

Nmap done: 255 IP addresses (4 hosts up) scanned in 8.16 seconds
```



```
Nmap done: 255 IP addresses (4 hosts up) scanned in 8.16 seconds
Nmap scan completed. Detected hosts: 192.168.58.1 192.168.58.2 192.168.58.177 192.168.58.254
Hosts list saved to: /var/log/nmap_scans/scan-20250915-224426.hosts.txt
Please choose the attack to execute:
1) Brute Force (authentication) - tries common username/password combos
2) MITM (ARP spoof) - intercepts traffic between host and gateway
3) Denial of Service (SYN flood) - floods target port with SYN packets
4) Random attack - pick one randomly
== Brute Force (Auth brute) ==
Attempts username/password combos against common services. Default: SSH (port 22). Use only in authorized labs.

Available detected hosts:
1) 192.168.58.1
2) 192.168.58.2
3) 192.168.58.177
4) 192.168.58.254
Choose target option:
1) Enter IP manually
2) Choose from detected list by number
3) Choose random detected host
Choose service to brute-force:
1) ssh (default)
2) ftp
3) telnet
4) rdp
5) custom (enter hydra module name)
[>] Running hydra against 192.168.58.177 service=ssh port=Choose port or press Enter for default (22):
Common: ssh(22) ftp(21) telnet(23) rdp(3389) http(80) https(443)
22 ...
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatio
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-15 22:44:34
[DATA] max 4 tasks per 1 server, overall 4 tasks, 71721995 login tries (l:5/p:14344399), ~17930499 tries per task
[DATA] attacking ssh://192.168.58.177:22/
[ERROR] could not connect to ssh://192.168.58.177:22 - Connection refused
[+] Cleaning up ...
```

Explanation:

- **Interface & IP display:** you selected interface eth0 and the script showed its IP - fulfills **2.1** (display available IPs).
- **Nmap scanning & host discovery:** the script ran a fast nmap scan and found 192.168.58.1, .2, .177, .254 and saved scan-*.gnmap/.hosts.txt in /var/log/nmap_scans - fulfills **2.1 / 2.2**.
- **Attack selection menu & descriptions:** you chose “Brute Force”; the script printed the attack title and description before running - fulfills **1.2**.
- **Target selection:** you targeted 192.168.58.177 (via manual input) which the script accepted - fulfills **2.5** (manual/list/random target allowed).
- **Separate attack function:** the brute-force run executed the brute_force_attack() flow (hydra invocation, temporary userlist, port handling) - fulfills **1.1** (each attack implemented as a function).
- **Logging:** the script logged attack start/end and would record successes/failures in /var/log/attack_log.txt (hydra error was produced and cleanup ran) - fulfills **3.1 / 3.2**.
- **Graceful behavior on unreachable service:** hydra showed Connection refused (target has no SSH or denies connection); the script handled this cleanly and continued to cleanup - acceptable behavior, not a script error.

- **Cleanup / safety:** the script ran cleanup ([+] Cleaning up...) and did not leave background state - satisfies the cleanup requirement and lab-safety expectation.

In one sentence: the run exercised the full user flow (interface → scan → attack menu → target selection → attack execution → logging → cleanup) and produced correct outputs/logging and safe termination, so it meets the project requirements.