VIEH GROUP
Think Secure, Think VIEH

Heist Reconnaisance

- Bash Heists

OUR TEAM

Bash Heists members with their respective roles

Siddharth

LEADER

Leading the project and helped in making the core of scan.sh script

Kartik

TEAM MEMBER

Helped in maintenance script for our tool, namely "Run-me-first.sh" script.

Abhishek

TEAM MEMBER

Helped in creating the pitch presentation for the project & also helped in the making of scan.sh script.

BRIEFING OF OUR

Project Idea

This Reconnaissance script named as Heist Recon script is whole based on the concept of enumeration stage of Pentesting.

Mission One - Subdomain enumeration

We have used subfinder and shuffledns to enumerate subdomains of the given root target.

Mission Two - DNS resolving

For DNS resolving, The favourite tool of hackers is Puredns and so is used in this script for the same.

Mission Three - Port scanning

Here, Nmap is used for Portscanning along with httpx for filtering the responses and Ports

Mission Four - Crawling and js enumeration

For crawling we have used Gospider and through the crawled pages we will enumerate js files with a simple one liner.

Resources Used

This Reconnaissance script named as Heist Recon script is whole based on the concept of enumeration stage of Pentesting.

- Project discovery's Tools
- Some medium blogs + Project discovery's doc
- Stackoverflow
- Googling
- Took 4 days

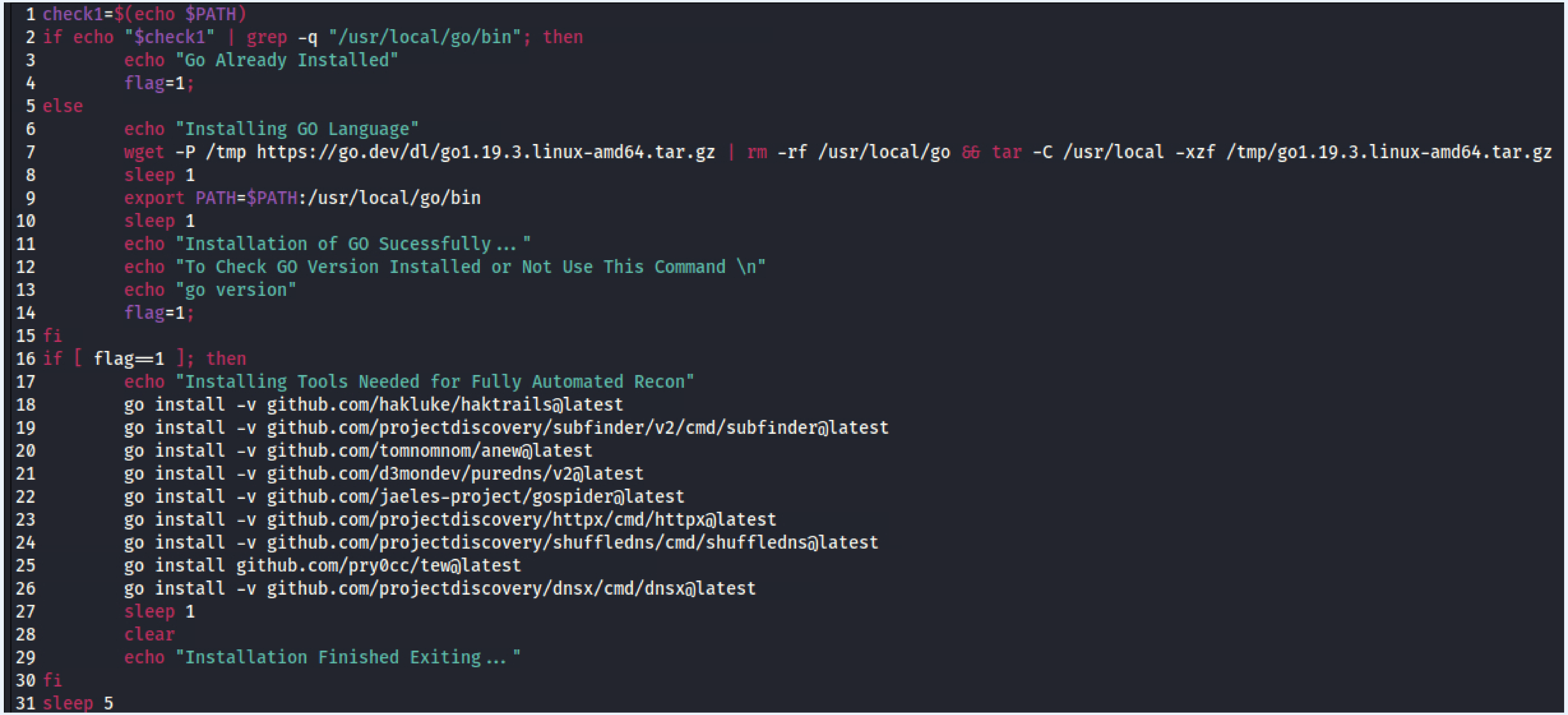
WORK FLOW OF THE PROJECT

TEAM BASH HEIST

Run-me-first.sh

RUN-ME-First.sh

This script cross checks for the requirements needed for the whole framework to run with and install the required files and tools for the script to run with no errors

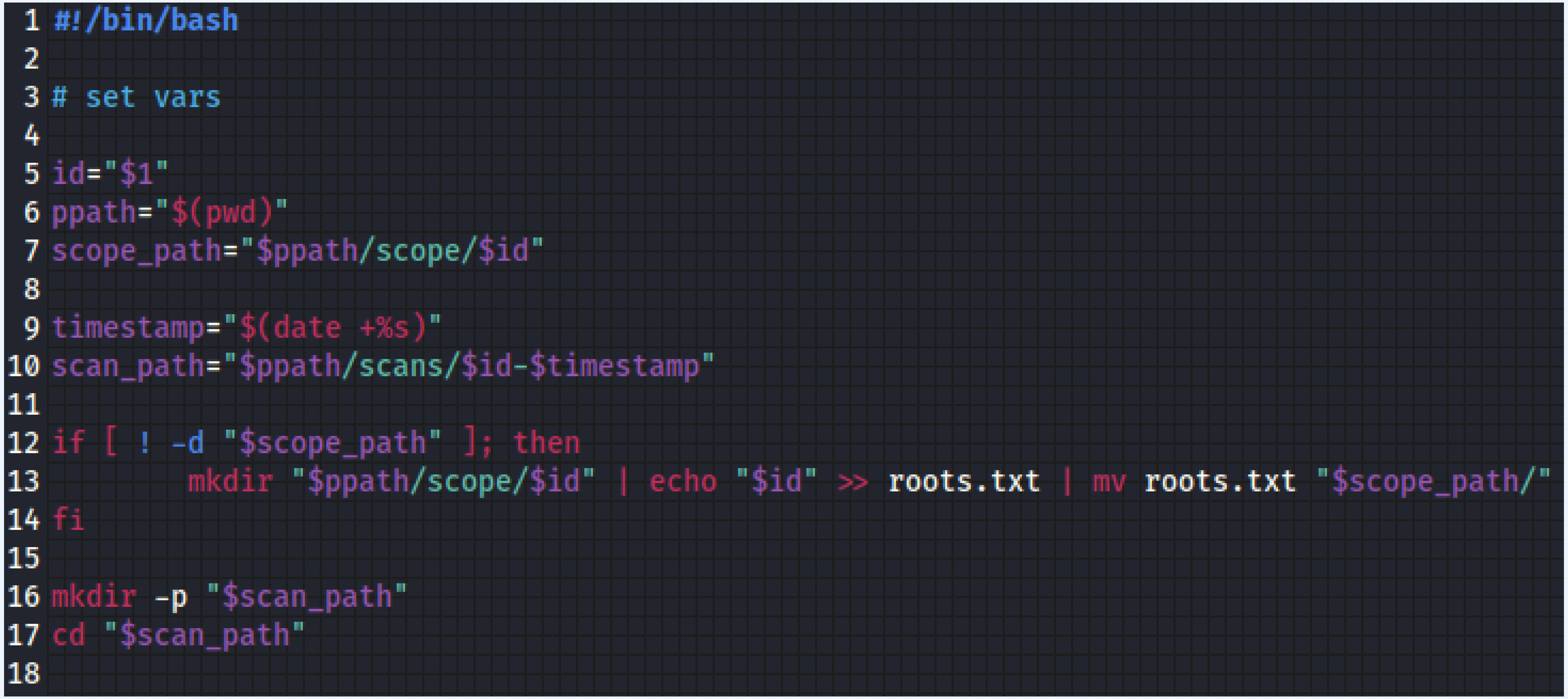


```bash
check1=$(echo $PATH)
if echo "$check1" | grep -q "/usr/local/go/bin"; then
        echo "Go Already Installed"
        flag=1;
else
        echo "Installing GO Language"
        wget -P /tmp https://go.dev/dl/go1.19.3.linux-amd64.tar.gz | rm -rf /usr/local/go && tar -C /usr/local -xzf /tmp/go1.19.3.linux-amd64.tar.gz
        sleep 1
        export PATH=$PATH:/usr/local/go/bin
        sleep 1
        echo "Installation of GO Sucessfully ... "
        echo "To Check GO Version Installed or Not Use This Command \n"
        echo "go version"
        flag=1;
fi
if [ flag==1 ]; then
        echo "Installing Tools Needed for Fully Automated Recon"
        go install -v github.com/hakluke/haktrails@latest
        go install -v github.com/projectdiscovery/subfinder/v2/cmd/subfinder@latest
        go install -v github.com/tomnomnom/anew@latest
        go install -v github.com/d3mondev/puredns/v2@latest
        go install -v github.com/jaeles-project/gospider@latest
        go install -v github.com/projectdiscovery/httpx/cmd/httpx@latest
        go install -v github.com/projectdiscovery/shuffledns/cmd/shuffledns@latest
        go install github.com/pry0cc/tew@latest
        go install -v github.com/projectdiscovery/dnsx/cmd/dnsx@latest
        sleep 1
        clear
        echo "Installation Finished Exiting ... "
fi
sleep 5
```

Scan.sh

# Setting Up Variables

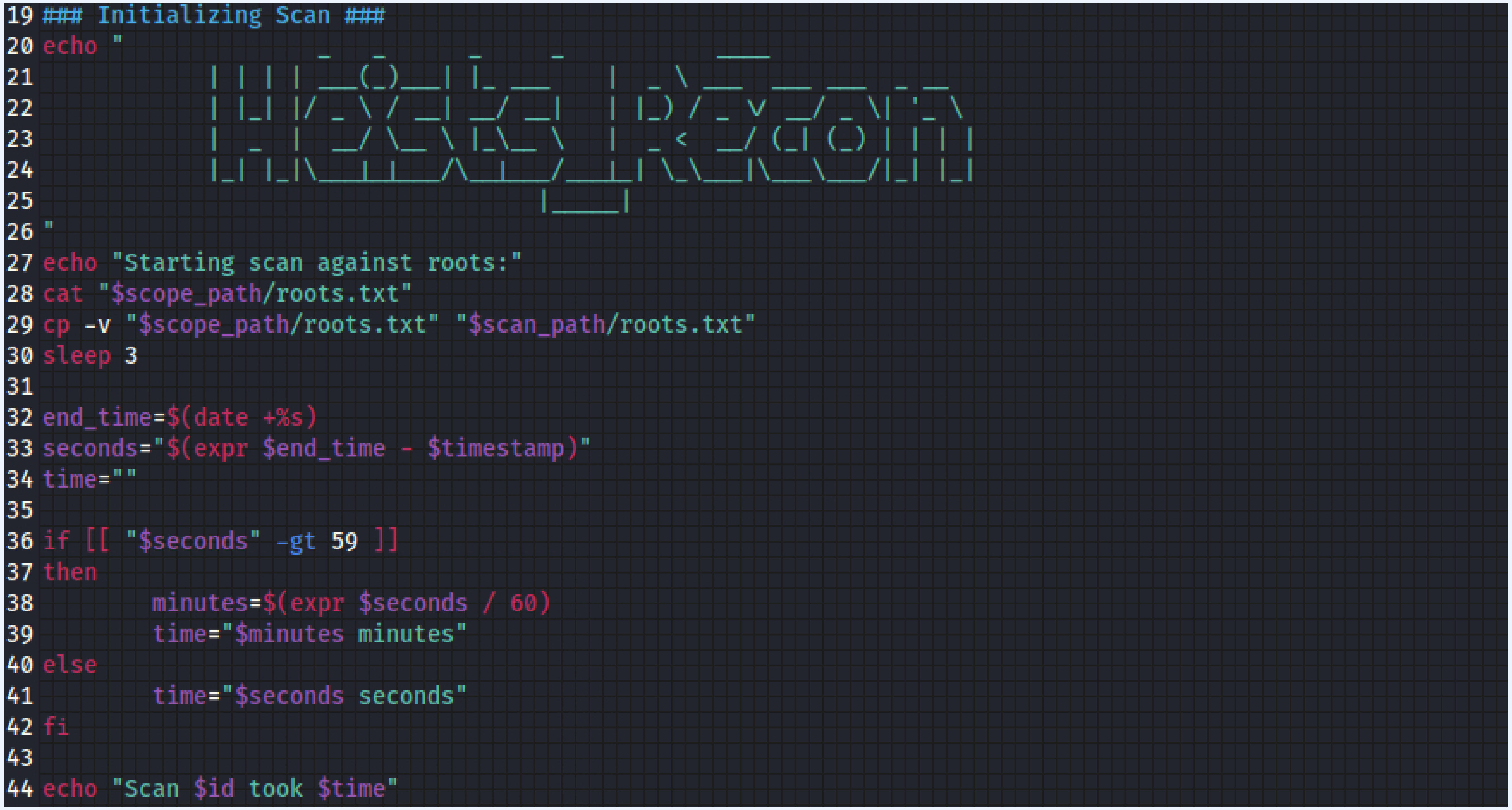
This script cross checks for the requirements needed for the whole framework to run with and install the required files and tools for the script to run with no errors



```bash
#!/bin/bash

# set vars

id="$1"
ppath="$(pwd)"
scope_path="$ppath/scope/$id"

timestamp="$(date +%s)"
scan_path="$ppath/scans/$id-$timestamp"

if [ ! -d "$scope_path" ]; then
        mkdir "$ppath/scope/$id" | echo "$id" >> roots.txt | mv roots.txt "$scope_path/"
fi

mkdir -p "$scan_path"
cd "$scan_path"

```

Initializing Scan Process

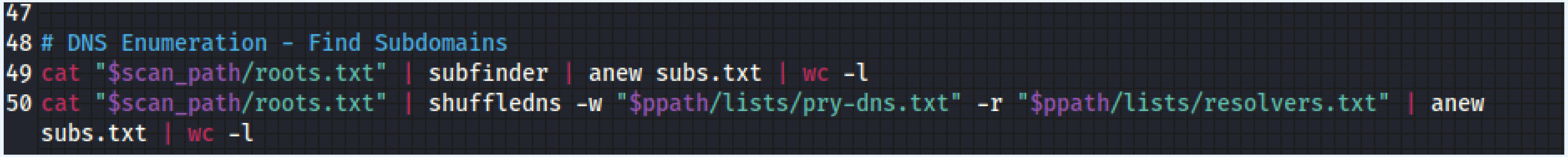
At the very starting of the Scanning part, this script initializes the directories and the result folders to arrange the whole scan in a systemised way.



```bash
### Initializing Scan ###
echo "
                     _   _      _     _         ____
            | | | | ___(_)___| |_ ___  |  _ \ ___  ___ ___  _ __
            | |_| |/ _ \ / __| __/ __| | |_) / _ v __/ _ \| '_ \
            |  _  |  __/ \__ \ |_\__ \ |  _ <  __/ (_| (_) | | | |
            |_| |_|\___|_|___/\__|___/ |_| \_\___|\___\___/|_| |_|
                                |____|
"
echo "Starting scan against roots:"
cat "$scope_path/roots.txt"
cp -v "$scope_path/roots.txt" "$scan_path/roots.txt"
sleep 3

end_time=$(date +%s)
seconds="$(expr $end_time - $timestamp)"
time=""

if [[ "$seconds" -gt 59 ]]
then
        minutes=$(expr $seconds / 60)
        time="$minutes minutes"
else
        time="$seconds seconds"
fi

echo "Scan $id took $time"
```

Performing Scans - Finding Subdomains

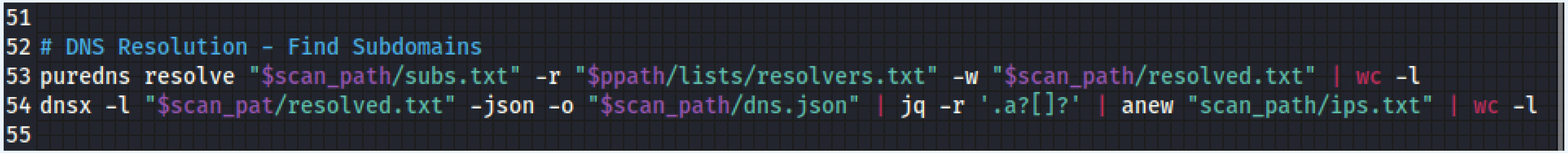
This part of script uses tools like Subfinder & Shuffledns to enumerate subdomains while Shuffledns also resolve those subdomains.



```

# DNS Enumeration - Find Subdomains
cat "$scan_path/roots.txt" | subfinder | anew subs.txt | wc -l
cat "$scan_path/roots.txt" | shuffledns -w "$ppath/lists/pry-dns.txt" -r "$ppath/lists/resolvers.txt" | anew subs.txt | wc -l
```

Performing Scans - Resolving the subdomains

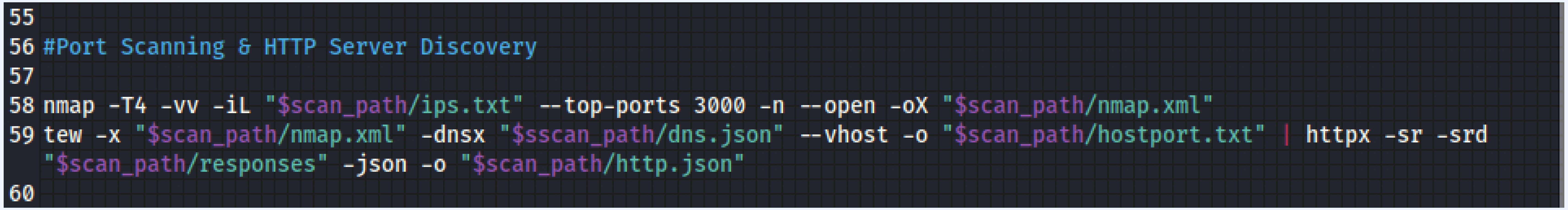
Here in this part, we have used Puredns tool to accurately filter out the wildcard subdomains, and also have added dnsx as a passive resource to get more results and filter the ips from the same.



```
# DNS Resolution - Find Subdomains
puredns resolve "$scan_path/subs.txt" -r "$ppath/lists/resolvers.txt" -w "$scan_path/resolved.txt" | wc -l
dnsx -l "$scan_pat/resolved.txt" -json -o "$scan_path/dns.json" | jq -r '.a?[]?' | anew "scan_path/ips.txt" | wc -l
```

Performing Scans - Port Scanning & HTTP discovery

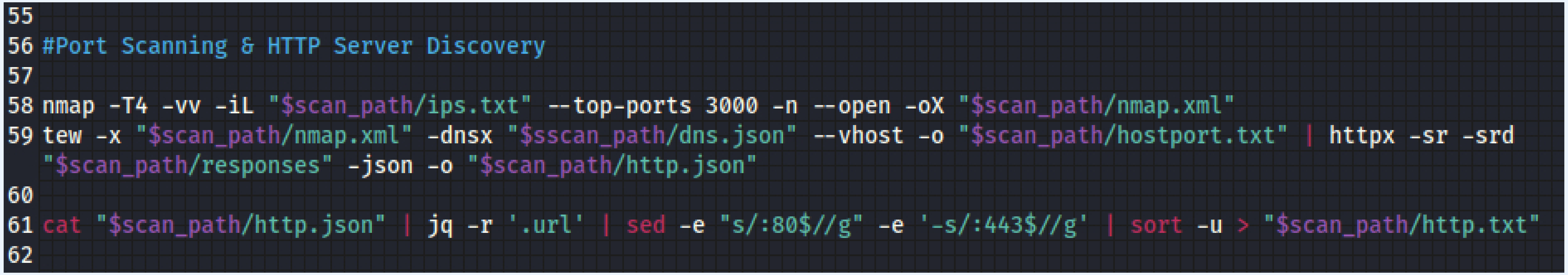
Running the Nmap scan on collected IP addresses and using tool named tew for beautifying the nmap result and filtering it out with httpx.



```
#Port Scanning & HTTP Server Discovery

nmap -T4 -vv -iL "$scan_path/ips.txt" --top-ports 3000 -n --open -oX "$scan_path/nmap.xml"
tew -x "$scan_path/nmap.xml" -dnsx "$sscan_path/dns.json" --vhost -o "$scan_path/hostport.txt" | httpx -sr -srd
"$scan_path/responses" -json -o "$scan_path/http.json"

```

Performing Scans - Port Scanning & HTTP discovery

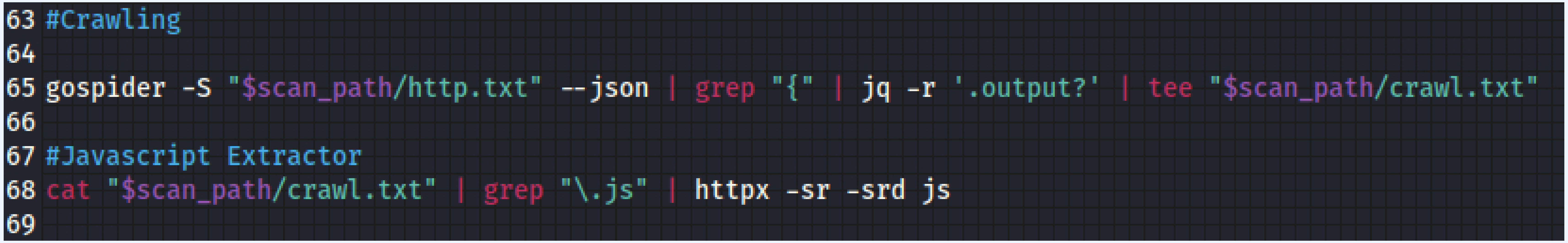
Running the Nmap scan on collected IP addresses and using tool named tew for beautifying the nmap result and filtering it out with httpx for port 80 and 443.



```

#Port Scanning & HTTP Server Discovery

nmap -T4 -vv -iL "$scan_path/ips.txt" --top-ports 3000 -n --open -oX "$scan_path/nmap.xml"
tew -x "$scan_path/nmap.xml" -dnsx "$sscan_path/dns.json" --vhost -o "$scan_path/hostport.txt" | httpx -sr -srd "$scan_path/responses" -json -o "$scan_path/http.json"

cat "$scan_path/http.json" | jq -r '.url' | sed -e "s/:80$//g" -e '-s/:443$//g' | sort -u > "$scan_path/http.txt"

```

Performing Scans - Crawling & Js Files Enumerate

At last but definitely not the least, We have used tool named Gospider for crawling throughout the web page and then used the one liner to extract the js endpoints through httpx.



```
#Crawling

gospider -S "$scan_path/http.txt" --json | grep "{" | jq -r '.output?' | tee "$scan_path/crawl.txt"

#Javascript Extractor
cat "$scan_path/crawl.txt" | grep "\.js" | httpx -sr -srd js

```

BASH

HEIST

Thank You