Project 5: Enumeration and Penetration automated tool

Sian Bin Chan

CFC 020823

Trainer: Kar Wei

Table of contents

Project 5: Vulnerability and Penetration testing

Introduction

This report presents a Bash script designed for network penetration testing, emphasizing its significance in detecting vulnerabilities and weak points within a network infrastructure. The script prompts users for network information, conducts scanning operations, and logs the results for analysis. It offers two distinct scan modes: Basic, which includes TCP/UDP port scanning and weak password detection, and Full, incorporating advanced vulnerability analysis. This report details the methodologies, discussions, and recommendations concerning the script's functionality and effectiveness.

Methodology

The script initiates by prompting the user for a network (IP address) to scan and a directory to save the results. Employing Bash regex, it validates the input IP address and searches for a relevant directory or suggests alternatives based on the user's choice. The save_to_log() function stores the scan results in the chosen directory, aiding in log management.

For weak credential detection, the script utilizes hydra, allowing users to input username lists and password lists. Additionally, it conducts nmap and masscan scans to identify TCP/UDP ports and service versions.

Two scan modes, 'Basic' and 'Full,' enable users to select the level of assessment desired. The 'Basic' mode encompasses TCP/UDP port scans and weak password brute-forcing, while the 'Full' mode integrates Nmap Scripting Engine (NSE) and vulnerability analysis.

Upon completion of scans, the script generates log files named 'basicscan_log.txt' and 'fullscan_log.txt' for 'Basic' and 'Full' scans, respectively, within the chosen directory.

Project 5: Vulnerability and Penetration testing

Discussion

User Input Acquisition

The script initiates by gathering user input for network scanning and result storage directories. It utilizes Bash's read command for user input capturing and performs input validation using Bash regex (=~ operator) to ensure the accuracy of the provided IP address [1].

```
##1.1 Get from the user a network to scan.
ipaddr=""
while true; do
    #Requests for the ip address input to scan
    read -p 'Welcome Pentester, please enter a network (IP address) to scan: ' ipaddr_input

    #Validation check with bash regex to ensure that only IP addressess are entered
    #Checks for 4 occuring 3 sequential digits separated by a dot (Creds: https://ioflood.com/blog/bash-regex/)
    if ! [[ $ipaddr_input =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$ ]]; then
        echo -e 'Invalid IP address entered. Please try again!\n'
    else
        ipaddr=$ipaddr_input
        break
    fi
done
```

Log File Management

A pivotal function, save_to_log(), has been crafted to save output into log files. It employs local variables and utilizes the echo command with redirection (>>) to append data to the specified log file within the chosen directory [2].

```
#Function to save output to log file
function save_to_log() {
    local log_file="$1"
    shift
    echo "$@" >> "${chosen_dir}/${log_file}"
}
```

Weak Credential Detection (hydrascan() Function)

This function prompts users to input a username list and either choose a built-in password list or provide their password list. It integrates the hydra tool to perform brute-force attacks against SSH, RDP, FTP, and Telnet services [3].

The internal password list, hardcoded within the script, offers common weak passwords. The user can choose either the internal list or provide a custom one, enhancing flexibility and enabling tailored assessments [4].

```
#2. Weak Credentials
function hydrascan()
##2.1 Look for weak passwords used in the network for login services.
{
    while true; do
        #Display prompt and gather username list
        read -p "Enter the path to the username list: " username_list

        #Validate if the file exists and has a compatible extension (.lst or .txt)
        if [ -f "$username_list" ] && [[ "$username_list" == *.lst || "$username_list" == *.txt ]]; then
            break  # Valid file, exit the loop
        else
            echo "Invalid username list. File must exist and have a .lst or .txt extension."
        fi
    done

###2.1.1 Have a built-in password.lst to check for weak passwords.
###2.1.2 Allow the user to supply their own password list.
    #Check if user wants to use a built-in password list
    internal_passwords=("123456" "12345678" "123456789" "12345" "1234567" "password" "1password" "abc123" "qwerty

    while true; do
        read -p "Do you want to use an internal password list? (yes/no): " use_internal_list
        if [ "$use_internal_list" == "yes" ]; then
            password_list=("${internal_passwords[@]}")
```

Project 5: Vulnerability and Penetration testing

Scan Modes: 'Basic' and 'Full' (basicscan() and fullscan() Functions)

Two distinct scan modes are provided: 'Basic' and 'Full', each encapsulated within the basicscan() and fullscan() functions, respectively

1. Basic Scan (basicscan())

   This function initiates a TCP/UDP port scan (nmap and masscan) along with weak password checks using hydrascan(). It collates the results and hydrascan() outputs, appending them into a specialized log file [5].

```
function basicscan()
{
    echo -e 'Running Basic scan... Please wait as this will take a while.\n'
    nmap_results=$(sudo nmap -p- -sV "$ipaddr" -oG -)
    echo -e "$nmap_results/n"
    masscan_results=$(sudo masscan -pU:0-65535 $ipaddr)
    echo -e "$masscan_results/n"
    hydrascan_results=$(hydrascan)
    echo -e '\nBasic scan complete.'

    #Creating a log file
    basicscan_log="basicscan_log.txt"
    touch "$basicscan_log"

    #Redirect scan output to log file
    save_to_log "$basicscan_log" "$nmap_results"
    save_to_log "$basicscan_log" "$masscan_results"
    save_to_log "$basicscan_log" "$hydrascan_results"

    #Display completion message
    echo -e '\nBasic scan complete. Results saved to basicscan_log.txt.'
```

2. Full Scan (fullscan())

   Extending the Basic Scan functionalities, the Full Scan encompasses NSE (nmap -sV --script vulners --script-args mincvss=5.0) to identify vulnerabilities [6]. It constructs comprehensive log files containing port scans, vulnerability analysis, and outputs from weak credential checks.ull Scan (fullscan()):

```
function fullscan()
{
    echo -e 'Running Full scan... Please wait as this will take a while.\n'
    nmap_results=$(sudo nmap -p- -sV "$ipaddr" -oG -)
    echo -e "$nmap_results/n"
    masscan_results=$(sudo masscan -pU:0-65535 $ipaddr)
    echo -e "$masscan_results/n"
    hydrascan_results=$(hydrascan)
    echo -e "$hydrascan_results/n"
```

Result Presentation and Search

The script allows users to view and search through the generated log files. The display_logs() function presents log contents, while search_results() enables keyword-based searches within the logs. These functions utilize cat for file content display and grep for search functionality [7].

```
function search_results() {
    local log_file="$1"
    echo -e "\nSearch inside $log_file:"
    read -p "Enter search keyword: " search_keyword

    if grep -qi "$search_keyword" "$log_file"; then
        grep -i "$search_keyword" "$log_file"
    else
        echo "No results found for '$search_keyword'."
    fi
}
    #Asks user if they want to search inside the results
while true; do
    read -p 'Do you want to search inside the results? (yes/no): ' search_option

    if [ "$search_option" = "yes" ]; then
        search_results "$basicscan_log"
        search_results "$fullscan_log"

    elif [ "$search_option" = "no" ]; then
        break

    else
        echo -e 'Invalid option. Please try again.\n'
    fi
done
```

Log Compression (zip_logs() Function)

Lastly, the script provides an option to compress log files into a ZIP archive using the zip command [8].

Conclusion

The Bash script effectively handles user input validation, directory selection, and executes network scanning operations. It facilitates flexible password list usage and provides comprehensive scanning modes to suit different assessment requirements. However, the script could benefit from additional error handling and user guidance during scan processes.

References

[1] "bash(1): GNU Bourne-Again SHell - Linux man page," *linux.die.net*. https://linux.die.net/man/1/bash

[2] "Regular expressions," *tldp.org*. https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_04_01.html (accessed Jan. 03, 2024).

[3] vanhauser-thc, "vanhauser-thc/thc-hydra," *GitHub*, Nov. 20, 2019. https://github.com/vanhauser-thc/thc-hydra

[4] Internal Passwords List - Script Implementation.

[5] Nmap, "Nmap," *Nmap.org*, 2017. https://nmap.org/

[6] "Chapter 9. Nmap Scripting Engine | Nmap Network Scanning," *Nmap.org*, 2010. https://nmap.org/book/nse.html

[7] Bash Manual Pages - cat, grep commands.

[8] Zip Command - Script Implementation.