- IDS 시스템의 정상적 운영 및 탐지 확인

본인의 실습 환경에 따 3 개의 임의의 룰을 작성하여 테스트하고 확인

```
alert icmp 192.168.5.5 any -> $HOME_NET any (msg:"WINDOWS PING";sid:1000000;rev:
1;)
alert http 192.168.5.5 any -> $HOME_NET 80 (msg:"WINDOWS HTTP";sid:1000001;rev:1
;)
alert tcp 192.168.5.5 any -> $HOME_NET 22 (msg:"WINDOWS SSH";sid:1000002;rev:1;)
```

Suricata rules

```
02/26/2025-20:38:19.762937  [**] [1:1000000:1] WINDOWS PING [**] [Classification: (nul
l)] [Priority: 3] {ICMP} 192.168.5.5:8 -> 192.168.5.111:0
```

```
02/26/2025-20:28:50.307480  [**] [1:1000001:1] WINDOWS HTTP [**] [Classification
: (null)] [Priority: 3] {TCP} 192.168.5.5:58364 -> 192.168.5.111:80
```

```
02/26/2025-20:31:07.652031  [**] [1:1000002:1] WINDOWS SSH [**] [Classification: (null
)] [Priority: 3] {TCP} 192.168.5.5:57958 -> 192.168.5.111:22
```

룰에 따른 PING, HTTP(80), SSH(22) 감지 확인

- Firewalld or IPTables 룰 작성 및 테스트

본인의 실습 환경에 따라 3 개의 임의의 룰을 작성하여 테스트하고 확인

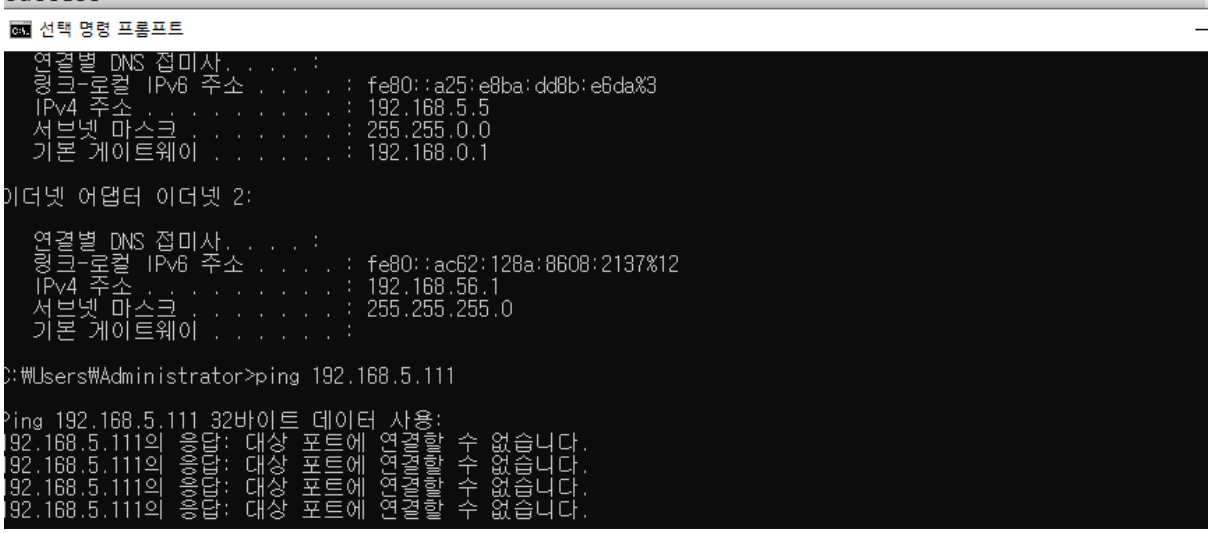
```
  rich rules:
        rule family="ipv4" source address="192.168.5.20" destination address="192.168.
5.111" port port="80" protocol="tcp" reject
        rule family="ipv4" source address="192.168.5.3" destination address="192.168.5
.111" port port="80" protocol="tcp" reject
        rule family="ipv4" source address="192.168.5.20" port port="80" protocol="tcp"
 reject
        rule family="ipv4" source address="192.168.5.5" destination address="192.168.5
.111" port port="80" protocol="tcp" reject
        rule family="ipv4" source address="192.168.5.3" destination address="192.168.5
.111" port port="22" protocol="tcp" reject
        rule family="ipv4" source address="192.168.5.5" destination address="192.168.5
.111" protocol value="icmp" reject
        rule family="ipv4" source address="192.168.5.5" port port="80" protocol="tcp"
```

여러 룰 추가

```
[root@localhost ~]# firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source
address='192.168.5.5' destination address='192.168.5.111' port protocol='tcp' port='80
' reject"
```

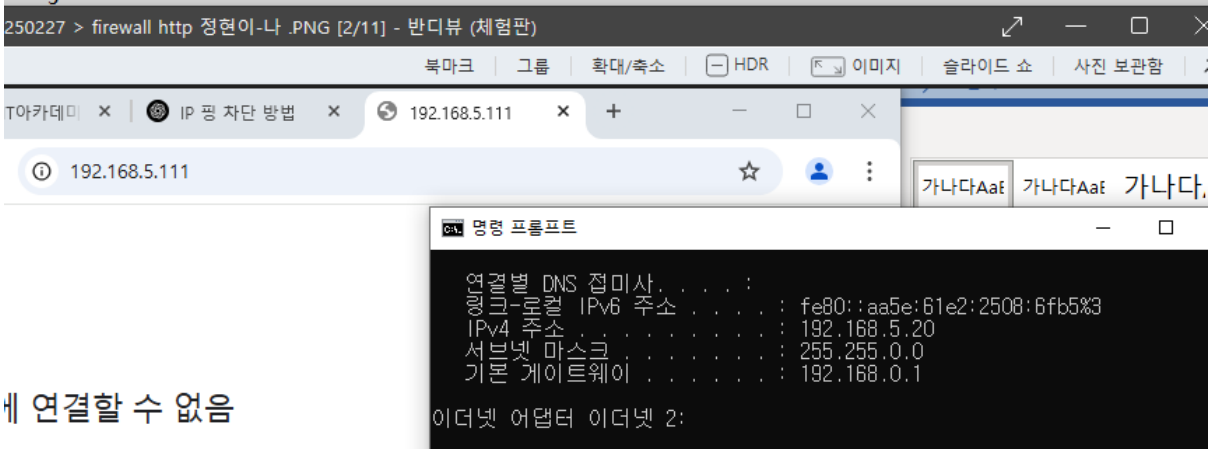
WINDOWS(.5)에서 오는 HTTP(.111) 접속 접속 차단

```
[root@localhost ~]# sudo firewall-cmd --permanent --add-rich-rule="rule family=ipv4 so
urce address='192.168.5.5' destination address='192.168.5.111' protocol value='icmp' r
eject"
success
```



WINDOWS(.5)에서 오는 PING(.111) 차단

```
        rule family="ipv4" source address="192.168.5.20" port port="80" protocol="tcp"
 reject
```



타 컴퓨터(정현 .20)에서 오는 HTTP(.111) 접속 차단

- 시스템에서 suid / sgid 파일을 find 명령어를 통해 확인하고 권한 상승 테스트를 진행

```
[root@localhost ~]# sudo find / -type f -perm -4000 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/umount
/usr/bin/mount
/usr/bin/su
/usr/bin/crontab
/usr/bin/passwd
/usr/bin/sudo
/usr/sbin/unix_chkpwd
/usr/sbin/pam_timestamp_check
/usr/sbin/grub2-set-bootflag
[root@localhost ~]# sudo find / -type f -perm -2000 2>/dev/null
/usr/bin/write
/usr/libexec/utempter/utempter
/usr/libexec/openssh/ssh-keysign
```

find 명령어를 통해 suid/sgid 파일을 확인

```
[jo@localhost /]$ id
uid=1000(jo) gid=1000(jo) groups=1000(jo) context=unconfined_u:unconfined_r:unconfined
_t:s0-s0:c0.c1023
[jo@localhost /]$ ./backdoor
[root@localhost /]# id
uid=0(root) gid=0(root) groups=0(root),1000(jo) context=unconfined_u:unconfined_r:unco
nfined_t:s0-s0:c0.c1023
[root@localhost /]#
```

사용자 jo 를 추가하고 backdoor 를 만들어 실행해 권한이 상승한 결과

- /root/koreait 파일 생성 후 rm koreait 명령어 실행 시 삭제가 되지 않도록 파일의 속성을 변경하고 확인

```
[root@localhost ~]# touch koreait
[root@localhost ~]# chattr +i /root/koreait
[root@localhost ~]# lsattr /root/koreait
----i---------------- /root/koreait
[root@localhost ~]# rm /root/koreait
rm: remove regular empty file '/root/koreait'? y
rm: cannot remove '/root/koreait': Operation not permitted
```

Immutable 명령어를 사용하여 속성의 변경을 확인하고 삭제가 되는지 확인한 결과 삭제가 되지 않는다

- ‘주요정보통신기반시설_기술적_취약점_분석_평가_방법_상세가이드‘에 따라

U01 ~ U04 까지의 계정관리 취약점을 점검할 수 있는 보안 점검 쉘 스크립트를 작성

```
▌U 01 ▌root 계정 원격접속 제한

root 계정의 원격 접속이 허용되어 있습니다.
차단을 원하시면 /etc/ssh/sshd_config 파일에서 PermitRootLogin 을 no로 수정하십시오.

▌U 02 ▌패스워드 복잡성 설정

패스워드 복잡성 설정 점검 결과 :
패스워드 복잡성 정책이 적용되어 있습니다.
정책 수정을 원하시면 /etc/pam.d/system-auth 파일과 /etc/login.defs 파일을 정책에 맞게 수정하십시오.

▌U 03 ▌계정 잠금 임계값 설정

계정 잠금 임계값 설정 점검 결과 :
로그인 실패 시 계정 잠금 설정이 적용되어 있지 않습니다. (/etc/pam.d/system-auth)
설정을 변경하려면 /etc/pam.d/system-auth 파일을 정책에 맞게  수정하십시오.
로그인 실패 시 계정 잠금 설정이 적용되어 있지 않습니다. (/etc/pam.d/password-auth)
설정을 변경하려면 /etc/pam.d/password-auth 파일을 정책에 맞게  수정하십시오.

로그인 실패 임계값 설정 :

로그인 실패 임계값 설정이 적용되어 있지 않습니다.
설정을 변경하려면 /etc/security/faillock.conf 파일을 정책에 맞게 수정하십시오.

▌U 04 ▌패스워드 파일 보호

/etc/passwd와 /etc/shadow에서 암호화 설정 확인 결과 :
사용자 'root'는 /etc/shadow에 암호화가 되어 있습니다.
[root@localhost ~]#
```

U 01~04 스크립트 실행 결과

```
echo
echo " ▌U 01 ▌root 계정 원격접속 제한"
echo
# sshd_config 파일 경로
sshd_config="/etc/ssh/sshd_config"
# PermitRootLogin 설정 값 확인
permit_root_login=$(grep -i "^PermitRootLogin" $sshd_config | awk '{print $2}')
# PermitRootLogin 설정 확인
if [[ "$permit_root_login" == "no" ]]; then
    echo "root 계정의 원격 접속이 차단되어 있습니다."
    echo "허용을 원하시면 /etc/ssh/sshd_config 파일에서 PermitRootLogin 을 yes로 수정하십시오."
else
    echo "root 계정의 원격 접속이 허용되어 있습니다."
    echo "차단을 원하시면 /etc/ssh/sshd_config 파일에서 PermitRootLogin 을 no로 수정하십시오."
fi
```

U 01 코드

```
echo
echo " ▌ U 02 ▌ 패스워드 복잡성 설정 "
echo
pam_config="/etc/pam.d/system-auth"
login_defs="/etc/login.defs"

check_password_complexity() {
       echo "패스워드 복잡성 설정 점검 결과 :"
if grep -q "pam_pwquality.so" $pam_config || grep -q "pam_cracklib.so" $pam_config;
then
       echo "패스워드 복잡성 정책이 적용되어 있습니다."
       echo "정책 수정을 원하시면 ${pam_config} 파일과 ${login_defs} 파일을 정책에 맞게 수정하십시오."
else
       echo "패스워드 복잡성 정책이 적용되어 있지 않습니다."
       echo "${login_defs} 를  수정하여 8자 이상의 영문, 숫자, 특수문자 조합으로 암호 설정 및 패스워드 복잡성 옵션을 설정하십시오. "
fi
}
check_password_complexity
```

U 02 코드

```
echo
echo " ▌ U 03 ▌ 계정 잠금 임계값 설정 "
echo
pam_config="/etc/pam.d/system-auth"
pam_password_config="/etc/pam.d/password-auth"

check_account_lock_threshold() {
    echo "계정 잠금 임계값 설정 점검 결과 :"

    if grep -q "pam_tally2.so" $pam_config || grep -q "pam_faillock.so" $pam_config; then
        echo "로그인 실패 시 계정 잠금 설정이 적용되어 있습니다. (/etc/pam.d/system-auth)"
    else
        echo "로그인 실패 시 계정 잠금 설정이 적용되어 있지 않습니다. (/etc/pam.d/system-auth)"
        echo "설정을 변경하려면 /etc/pam.d/system-auth 파일을 정책에 맞게  수정하십시오."
    fi

    # /etc/pam.d/password-auth 파일에서 pam_tally2.so 또는 pam_faillock.so 확인
    if grep -q "pam_tally2.so" $pam_password_config || grep -q "pam_faillock.so" $pam_password_config; then
        echo "로그인 실패 시 계정 잠금 설정이 적용되어 있습니다. (/etc/pam.d/password-auth)"
    else
        echo "로그인 실패 시 계정 잠금 설정이 적용되어 있지 않습니다. (/etc/pam.d/password-auth)"
        echo "설정을 변경하려면 /etc/pam.d/password-auth 파일을 정책에 맞게  수정하십시오."
    fi

    echo
    echo "로그인 실패 임계값 설정 :"
    echo

    # /etc/security/faillock.conf
    if [ -f /etc/security/faillock.conf ]; then
        if grep -q "deny=" /etc/security/faillock.conf; then
            deny_threshold=$(grep "deny=" /etc/security/faillock.conf | awk -F '=' '{print $2}')
            echo "로그인 실패 임계값 : $deny_threshold"
        else
            echo "로그인 실패 임계값 설정이 적용되어 있지 않습니다."
             echo "설정을 변경하려면 /etc/security/faillock.conf 파일을 정책에 맞게 수정하십시오."
         fi
    else
        echo "/etc/security/faillock.conf 파일이 존재하지 않습니다."
    fi
}

check_account_lock_threshold
```

U 03 코드

```
echo
echo " ▌U 04 ▌패스워드 파일 보호"
echo

check_password_encryption() {
  echo "/etc/passwd와 /etc/shadow에서 암호화 설정 확인 결과 :"

  while IFS=: read -r username password _; do
    if [[ "$username" == "root" || "$username" == "en" ]]; then
      shadow_entry=$(grep "^$username:" /etc/shadow)

      if [[ "$password" == "x" ]]; then
          if [[ -z "$shadow_entry" || "$(echo $shadow_entry | cut -d: -f2)" == "" ]]; then
          echo "경고: 사용자 '$username'는 /etc/passwd에 암호화가 되어 있지만 /etc/shadow에 암호화된 비밀번호가 없습니다."
        else
          echo "사용자 '$username'는 /etc/shadow에 암호화가 되어 있습니다."
        fi
      else
        echo "경고: 사용자 '$username'는 /etc/passwd에 암호화가 되어 있지 않습니다."
        echo "패스워드 정책을 설정하여 적용하십시오."
      fi
    fi
  done < /etc/passwd
}

check_password_encryption
```

u 04 코드

- BoF 공격에 대해 간략히 서술하고 코드를 작성하여 테스트를 진행하여 확인

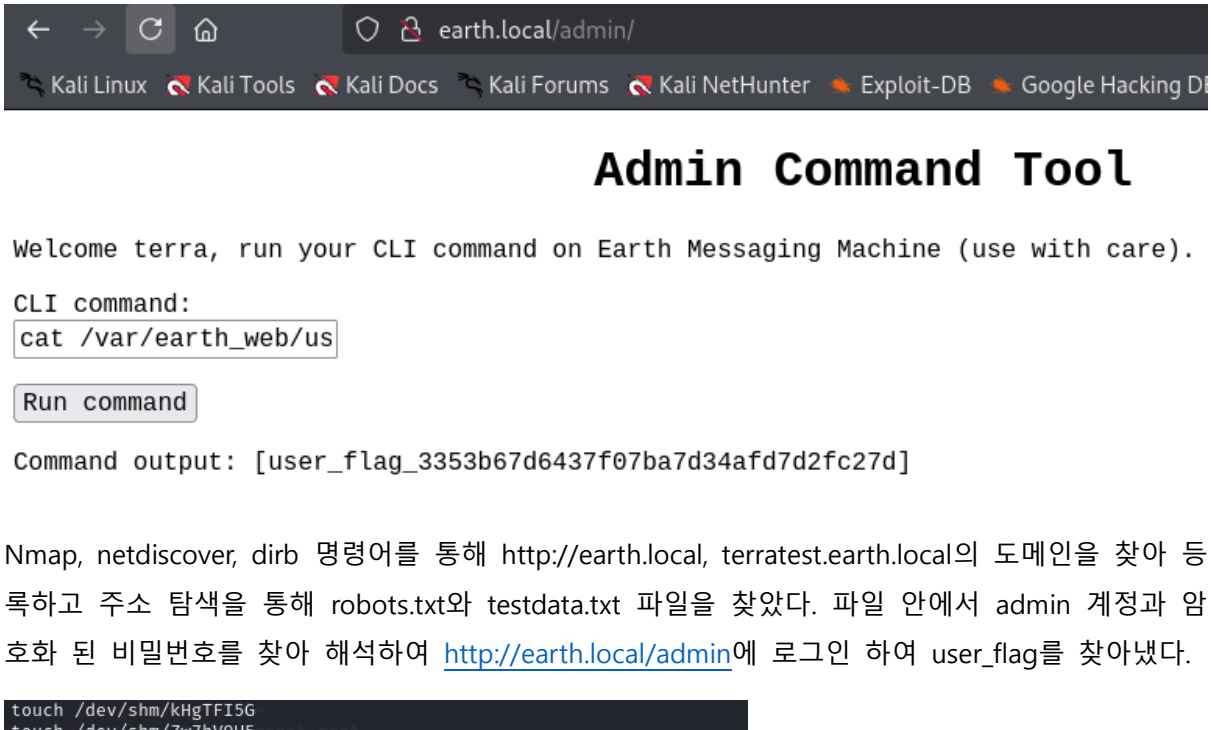
BoF 란 Buffer overflow 로 버퍼의 크기를 초과하여 데이터를 쓸 때 다른 메모리영역에 덮어씌우기가 되는데, 덮어씌워지기가 되면서 악성코드가 삽입, 예기치 않은 동작을 하게 되는 등 오류가 일어나는 현상을 말한다.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int check_auth(char *password){
  int auth = 0;
  char temp [10];
  strncpy(temp, password, strlen(password));

  if(!strcmp(temp, "ADMINPASSWORD"))
     auth = 1;
     return auth;
}

int main(int argc, char *argv[]){
  if(argc!=2){
    printf("Usage:./bof ADMIN_PASSWORD \n");
    exit(-1);
  }
  if(check_auth(argv[1])){
    printf("ACCESS SUCCESSFUL! \n");
  }else{
    printf("ACCESS DENIED! \n");
  }
}
```

```
[root@localhost ~]# ./bof 1234
ACCESS DENIED!
[root@localhost ~]# ./bof ADMINPASSWORD
ACCESS SUCCESSFUL!
[root@localhost ~]# ./bof 123412341234123412341234
ACCESS SUCCESSFUL!
Segmentation fault (core dumped)
```

BoF 코드와 테스트 결과 (오버플로우 발생으로 엑세스가 성공함)

- CTF VM 를 해킹하여 user_flag 및 root_flag 를 획득



Nmap, netdiscover, dirb 명령어를 통해 http://earth.local, terratest.earth.local의 도메인을 찾아 등록하고 주소 탐색을 통해 robots.txt와 testdata.txt 파일을 찾았다. 파일 안에서 admin 계정과 암호화 된 비밀번호를 찾아 해석하여 http://earth.local/admin에 로그인 하여 user_flag를 찾아냈다.

이후 root flag를 이용해 earth와 kali를 연결하여 reset_root를 실행하기 위한 트리거를 만들었다. 결국 root계정의 비밀번호를 Earth로 초기화하고 계정에 접속해 root flag를 찾았다.

- IDS 시스템의 업데이트와 리눅스의 커널 업데이트에 대해 서술하시오

IDS(침입탐지시스템)는 네트워크, 시스템에 발생하는 침입, 공격을 실시간으로 감지하는 시스템으로 주요 요소로는 시그니처 업데이트, 기능업데이트 등이 있다.

시그니처 업데이트는 이미 알려진 공격의 패턴을 저장, 참고하여 네트워크 트래픽을 분석하여 침입을 식별한다. 이 업데이트는 새로운 유형의 공격을 탐지할 수 있게 해 준다. 기능 업데이트는 IDS 시스템이 공격 대상이 될 경우가 있기 때문에 IDS시스템의 보안 취약점을 수정하고 패치하는 것이다.

리눅스 커널은 하드웨어와 상위 소프트웨어 간의 상호작용을 관리한다. 커널 업데이트는 시스템의 성능, 보안, 기능 향상을 돕고 보안패치, 기능 개선 등이 있다.

커널에 보안 취약점이 발견되면 해커들이 악용할 우려가 있기 때문에 즉시 수정하여 업데이트 하하는 것 중요하다. 커널 버전을 최신으로 업데이트하여 시스템 성능 향상, 새로운 하드웨어 지원을 받으면 좋다.