# EAM 인증을 활용한 통합관제시스템 구축

2018. 11. 7


담당 교수 : 양환석 교수님

8층에서 살아남기

(김성윤, 박영진, 박광우, 김인권, 임종철, 윤새민)

# 목 차

# 조원 편성

| 성 명 | 담 당 |
|---|---|
| 김성윤 | 팀장 / EAM 인증체제 구현 |
| 박영진 | 웹 페이지 구축 / EAM 인증체제 구현 |
| 박광우 | 웹 페이지 구축 / DB 설계 |
| 김인권 | 쉘 프로그래밍 / IDS Rules 결정 |
| 임종철 | 방화벽 / DB 설계 |
| 윤새민 | 쉘 및 UI 프로그래밍 |

# 주제 선정

❍ 현재 널리 사용되는 ID/PW 인증방식은 각 서버마다 별도의 ID/PW 사용으로 기억에 의존하거나 메모 방식으로 관리
⇨ 보안관리에 한계가 있고 사용 불편

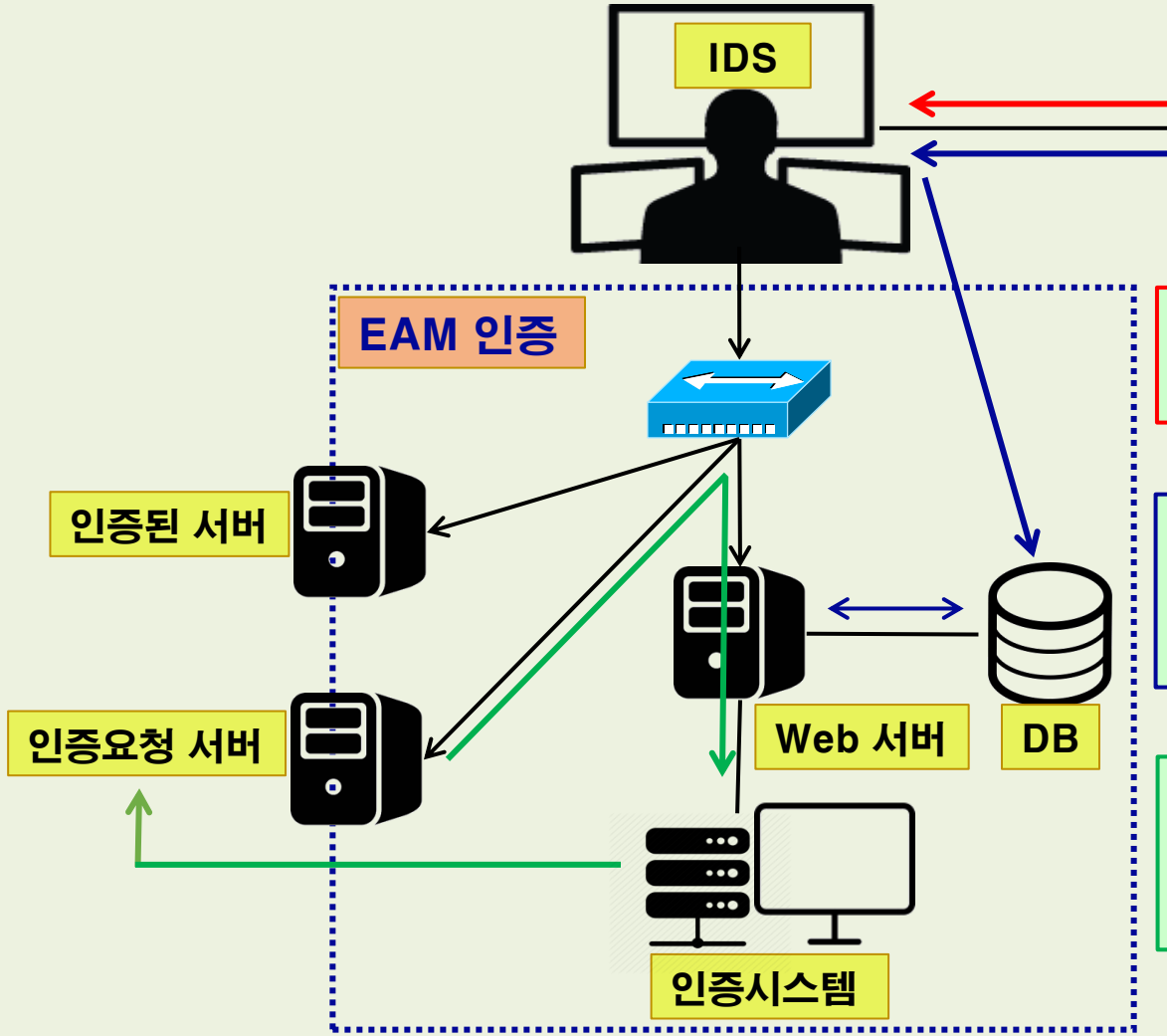
❍ 또한 클라이언트별 권한관리가 어려워 대형 서버 등에 적용이 제한

⇨ EAM 인증/권한관리 방식을 적용, SSO 인증, 차등적인 권한관리 및 유해 트래픽을 탐지/차단하는 방안을 연구

EAM(Extranet Access Management) ⇨ 단일 ID와 PW로 인증과 접근 권한을 동시에 관리하는 기술(SSO + 권한관리)

※ 단일 인증으로 여러 서비스를 이용할 수 있고, 권한관리를 통해 각 계정별 차등적 권한 적용이 가능

# 구상도




1. 유해 트래픽 감지 시 자동으로 방화벽에 등록

2. 유해 트래픽과 트래픽량 정보를 파싱하여 DB에 저장

3. 최초 서버 합류 시 인증
   - 서버 승인 요청
   - 관리자 인증/승인

실시간 트래픽 량과 유해한 트래픽, 과거 유해 IP와 실시간 차단 목록 확인 가능

# 추진 경과

(단위 : 월)

| 구분 \ 기간 (2018년) | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|
| 환경구축 | | | | | | | | | |
| IDS & DB | | | | | | | | | |
| 쉘 프로그래밍 | | | | | | | | | |
| 웹 프로그래밍 | | | | | | | | | |
| EAM 인증 | | | | | | | | | |
| 성능 점검/보완 | | | | | | | | | |

# 개발 환경 및 시스템 개발(1/8)

## 개발 환경

| | |
|---|---|
| 운영체제 | Ubuntu 16.04 LTS |
| WEB | Html5, PHP7, CSS, J query |
| 침입탐지(IDS) | Suricata |
| 공격 차단 | IPtables, Shell script |
| EAM 인증 | Node.js, Shell script |

# 개발 환경 및 시스템 개발(2/8)

## IDS 구축(Suricata)

❍ IDS 탐지를 위한 Rule 작성

Suricata

Rule 목록

```
kim@ubuntu:/etc/suricata/rules$ ls
app-layer-events.rules          emerging-scada.rules
botcc.portgrouped.rules         emerging-scan.rules
```

Rule 작성

```
  - decoder-events.rules # available in suricata sources under rule
  - stream-events.rules  # available in suricata sources under rules dir
  - http-events.rules    # available in suricata sources under rules dir
  - smtp-events.rules    # available in suricata sources under rules dir
  - dns-events.rules     # available in suricata sources under rules dir
  - tls-events.rules     # available in suricata sources under rules dir
  - modbus-events.rules  # available in suricata sources under rules dir
  - app-layer-events.rules  # available in suricata sources under rules dir
  - dnp3-events.rules       # available in suricata sources under rules dir
  - ntp-events.rules       # available in suricata sources under rules dir

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config
```

**Suricata 구축 및 유해 Traffic를 거르기 위한 Rule 적용**

# 개발 환경 및 시스템 개발(3/8)

## IDS 구축(유해트래픽 탐지)

❍ 로그파일에 유해트래픽 기록

IDS 로그파일

```
root@ubuntu:/home/kim# cat fast.log
05/13/2018-22:25:21.698802  [**] [1:2240001:2] SURICATA DNS Unsolicited response [**]
[Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.233.2:53
 -> 192.168.233.148:33715
05/13/2018-22:25:21.698802  [**] [1:2240001:2] SURICATA DNS Unsolicited response [**]
[Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 192.168.233.2:53
 -> 192.168.233.148:33715
05/13/2018-22:39:09.564876  [**] [1:2240001:2] SURICATA DNS Unsolicited response [**]
[Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 168.126.63.1:53
-> 192.168.233.148:49169
05/13/2018-23:12:27.585699  [**] [1:2240001:2] SURICATA DNS Unsolicited response [**]
[Classification: Generic Protocol Command Decode] [Priority: 3] {UDP} 168.126.63.1:53
-> 192.168.233.148:40729
05/13/2018-23:47:39.748349  [**] [1:10001:1] kimsungyonn rule [**] [Classification: (n
ull)] [Priority: 3] {TCP} 192.168.233.148:35662 -> 112.175.50.171:80
05/13/2018-23:47:39.798710  [**] [1:10001:1] kimsungyonn rule [**] [Classification: (n
ull)] [Priority: 3] {TCP} 192.168.233.148:35662 -> 112.175.50.171:80
05/13/2018-23:47:39.824377  [**] [1:10001:1] kimsungyonn rule [**] [Classification: (n
ull)] [Priority: 3] {TCP} 192.168.233.148:35662 -> 112.175.50.171:80
```

**지정한 Rule에 의해 탐지된 유해트래픽 로그파일에 기록**

# 개발 환경 및 시스템 개발(4/8)

## IDS 구축(유해트래픽 저장)

❍ 실시간 로그 변화 체크

```
root@ubuntu:/home/kim# ./newdb.sh
file checking
4423
file c
4423
```

Log파일 변화 체크

```
mysql> select * from attacklog;
+----+---------------------+---------+-------------------+-----------------------+-----------------------------------------------------------------------+
| id | time                | Protocol | src               | dst                   | rule                                                                  |
+----+---------------------+---------+-------------------+-----------------------+-----------------------------------------------------------------------+
|  1 | 05/13/2018-22:25:21.698802 | UDP      | 192.168.233.2:53  | 192.168.233.148:33715 | SURICATA DNS Unsolicited response |
|  2 | 05/13/2018-22:39:09.564876 | UDP      | 168.126.63.1:53   | 192.168.233.148:49169 | SURICATA DNS Unsolicited response |
|  3 | 05/13/2018-23:12:27.585699 | UDP      | 168.126.63.1:53   | 192.168.233.148:40729 | SURICATA DNS Unsolicited response |
|  4 | 05/13/2018-23:47:39.748349 | TCP      | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
|  5 | 05/13/2018-23:47:39.798710 | TCP      | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
```

변화가 생긴 로그 DB에 정보 저장

변화가 생긴 로그파일의 기록을 데이터베이스에 저장

# 개발 환경 및 시스템 개발(5/8)

## IPtables 구축(유해 IP 방화벽 등록)

❍ 저장된 로그 정보를 이용하여 방화벽에 유해 IP를 등록

IP등록 쉘코드

```
#!/bin/bash

echo "dynamic(yes) static(no) defult(key)"
read tem

case "$t
   yes |
echo "Fi
read qwe
sort -u
while re
    echo
iptables
done < a
iptables
iptables
   n* |
echo "IP
read ipt

echo "po
read por

echo "pr
read pro

echo "$i
iptables
iptables
iptables
   *) ip
```

방화벽 현황

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:22
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:21
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpts:1024:65535
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443
ACCEPT     tcp  --  192.168.1.1          0.0.0.0/0           tcp dpt:99
ACCEPT     tcp  --  192.168.0.1          192.168.0.8         tcp dpt:22
DROP       tcp  --  192.168.233.148      115.175.50.171      tcp dpt:80

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp spt:21
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp spts:1024:65535
```

**Suricata Rule에 걸린 유해 IP을 방화벽(IPtables)에 등록**

# 개발 환경 및 시스템 개발(6/8)

## 인증 소스코드(클라이언트)

에이전트

```
var io = require('socket.io-client');
const os = require('os');
var commanda = process.argv[2];
var commandb = process.argv[3];
var commandc = process.argv[4];
var commandd = process.argv[5];
var array = {
  platform: os.platform(),
  hostname: os.hostname(),
  uptime: os.uptime(),
  cpus: os.cpus(),
  network: os.networkInterfaces(),
  totalmem: os.totalmem(),
  loadavg: os.loadavg(),
  username: commanda,
  password: commandb,
  key: commandc,
  lev: commandd  };
```

클라이언트의 정보 값 비교를 위한 입력 배열

클라이언트의 PC 정보를 수신

**요청 시 PC정보를 같이 받아와서 클라이언트 PC파악에 유리함**

# 개발 환경 및 시스템 개발(7/8)

## 인증 소스코드(인증서버)

인증 서버

```
socket.on('example message 2', function (data) {
connection.connect(function (err) { //디비연결

    connection.query("select * from users where username = ?", [info[0].username], function (err, resu
        if (err) throw err;
        if (md5(info[0].password) === result[0].password) { # 패스워드 비교
            if (info[0].key == result[0].authkey) { // auth compare #키값
                console.log("auth success");
                connection.query("select * from servers where username = ?"
                    if (err) throw err;
                    if (result.length === 0) {
                        connection.query("insert into servers (username, ip, mac, hostname, platform, permi
                            [info[0].username, info[0].ip, info[0].mac  info[0].hostname, info[0].platform,
                                if (err) throw err;
const text = info[0].ip+" "+info[0].lev;
fs.writeFileSync("target.txt", '\ufeff' + text, {encoding: 'utf8'});
    exit('insert db success');
```

클라이언트의 정보를 비교후 일치하면 승인 대기열에 추가

추후 방화벽 등록을 위해서 텍스트파일로 정보를 저장

형태로 저장

**클라이언트 정보 비교 확인 후, 승인 대기열에 추가하고 추가 정보를 파일로 기록**

# 개발 환경 및 시스템 개발(8/8)

## 방화벽 등록(권한별 자동 등록)

방화벽 등록

```
if [ 4 = $autha ]
then
echo "4 insert" > `date +%H-%M-%S`.txt
```

권한 레벨 4 : 디폴트

```
if [ 2 = $autha ]
then
echo "2 insert" > `date +%H-%M-%S`.txt
iptables -t nat -A POSTROUTING -s IP -o ens33 -j MASQUERADE
sshpass -p 'pw' scp -o StrictHostKeyChecking=no /home/kim/Desktop/
sshpass -p 'pw' scp -o StrictHostKeyChecking=no /home/kim/Desktop/agent/target.txt mail@192.168
fi
```

권한 레벨 2 :Internet
내부망 DNS 서버
내부망 Mail 서버

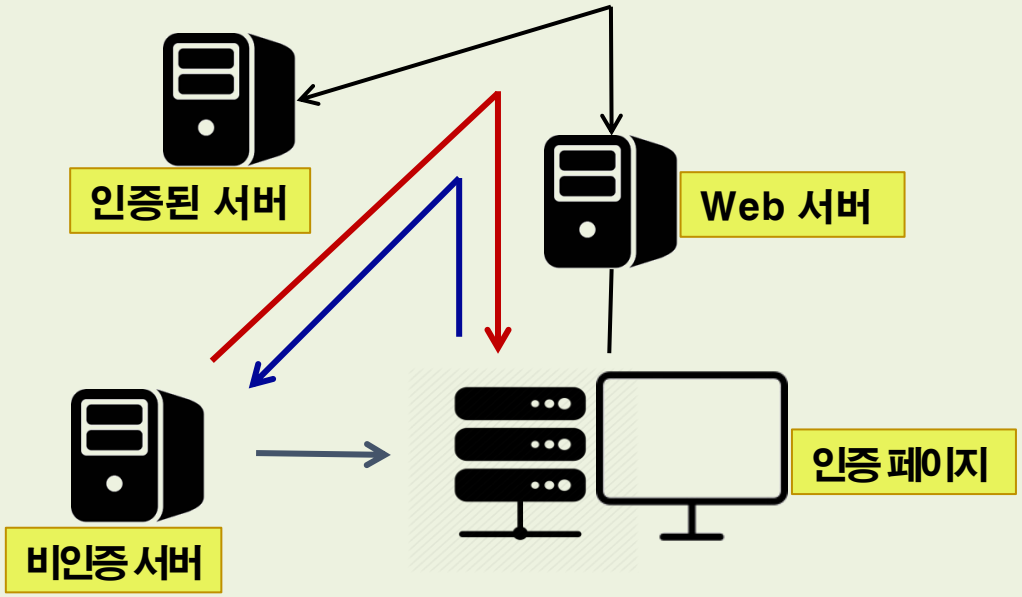
```
if [ 1 = $autha ]
then
echo "1 insert" > `date +%H-%M-%S`.txt
iptables -t nat -A POSTROUTING -s IP -o ens33 -j MASQUERADE
sshpass -p 'pw' scp -o StrictHostKeyChecking=no /home/kim/Desktop/agent/target.txt web@192.168.
sshpass -p 'pw' scp -o StrictHostKeyChecking=no /home/kim/Desktop/agent/target.txt dns@192.168.
sshpass -p 'pw' scp -o StrictHostKeyChecking=no /home/kim/Desktop/agent/target.txt mail@192.168
fi
```

권한 레벨 1 : 내부망에
관한 모든 접근권한 부여

**인증페이지에서 관리자가 권한 체크 후 승인을 하면, 권한 별로 자동 방화벽 등록**

# 개발 결과 및 시스템운영(1/6)

## EAM 운영절차(승인요청)




1. 비인가 서버의 내부망 합류 요청

2. 서버에서 인증 에이전트 파일 제공

3. 클라이언트 별로 인증 코드로 관리자 서버에 승인요청

인증 에이전트 파일 제공

TAR

클라이언트 별 인증 코드

nodejs cli.js Username Password Key Level

username : "test1", password : "your password", key : "YLQQZBckjN", level : "(1~4)"

# 개발 결과 및 시스템운영(2/6)

## EAM 운영절차(관리자 승인)

인증 페이지

승인 대기열

| ID | 소유자 | 아이피 | 맥 주소 | 호스트이름 | 운영체제 | 인증시간 | 권한레벨 | 승인 | 거부 |
|---|---|---|---|---|---|---|---|---|---|
| 9 | b | 192.168.253.129 | 00:0c:29:74:b5:cf | ubuntu | linux | 2018-10-14 12:04:20 | 4 | 승인 확인 | 거부 |
| 11 | j | 192.168.253.129 | 00:0c:29:74:b5:cf | ubuntu | linux | 2018-10-14 12:48:58 | 4 | 승인 확인 | 거부 |

⇩ 권한레벨 확인 후 승인

승인된 서버

승인 받은 클라이언트 목록

| ID | 소유자 | 아이피 | 맥 주소 | 호스트이름 | 운영체제 | 인증시간 | 권한레벨 |
|---|---|---|---|---|---|---|---|
| 1 | qw | 192.168.253.129 | 00:0c:29:74:b5:cf | ubuntu | linux | 2018-10-14 06:55:49 | 4 |
| 2 | qwe | 192.168.253.129 | 00:0c:29:74:b5:cf | ubuntu | linux | 2018-10-14 07:35:58 | 4 |

**관리자 인증 페이지에서 요청을 확인 한 후에 인증 승인**

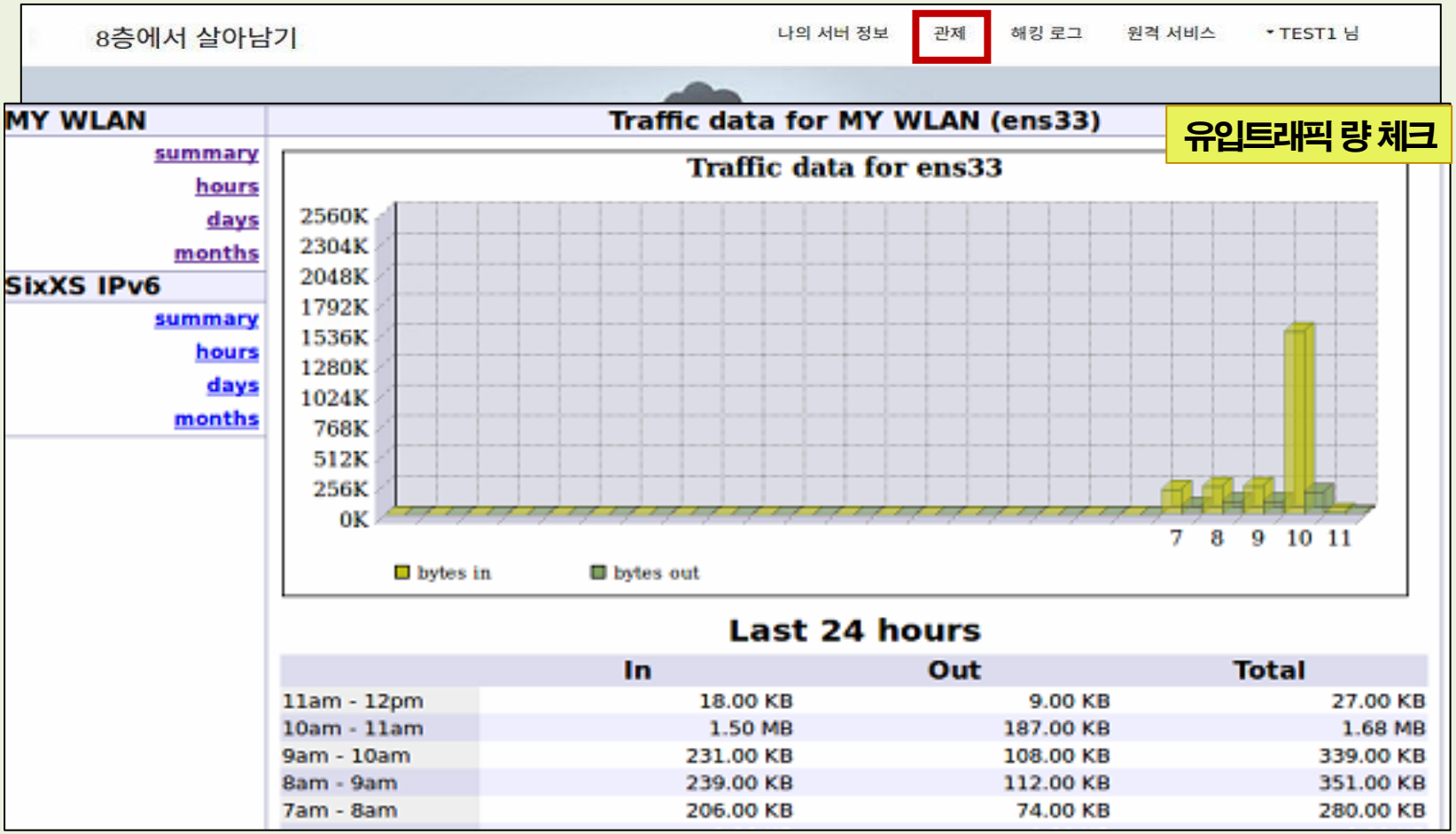
# 개발 결과 및 시스템운영(3/6)

## 메인 홈페이지


**클라이언트의 메인 홈페이지**

# 개발 결과 및 시스템운영(4/6)

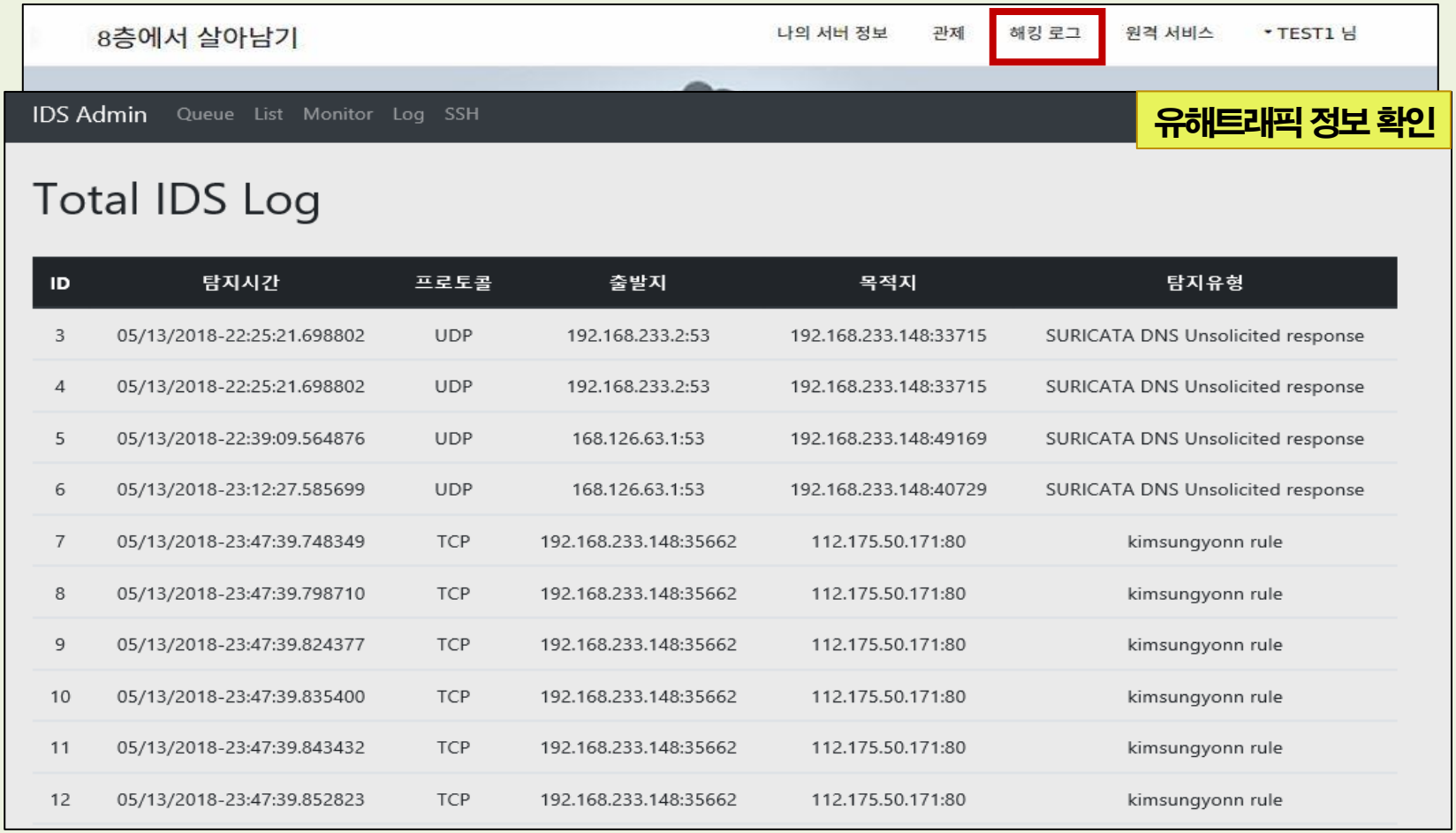
## 메인 홈페이지(관제)




**관제 메뉴에서 자신의 트래픽 유입량을 확인**

# 개발 결과 및 시스템운영(5/6)

## 메인 홈페이지(해킹로그)

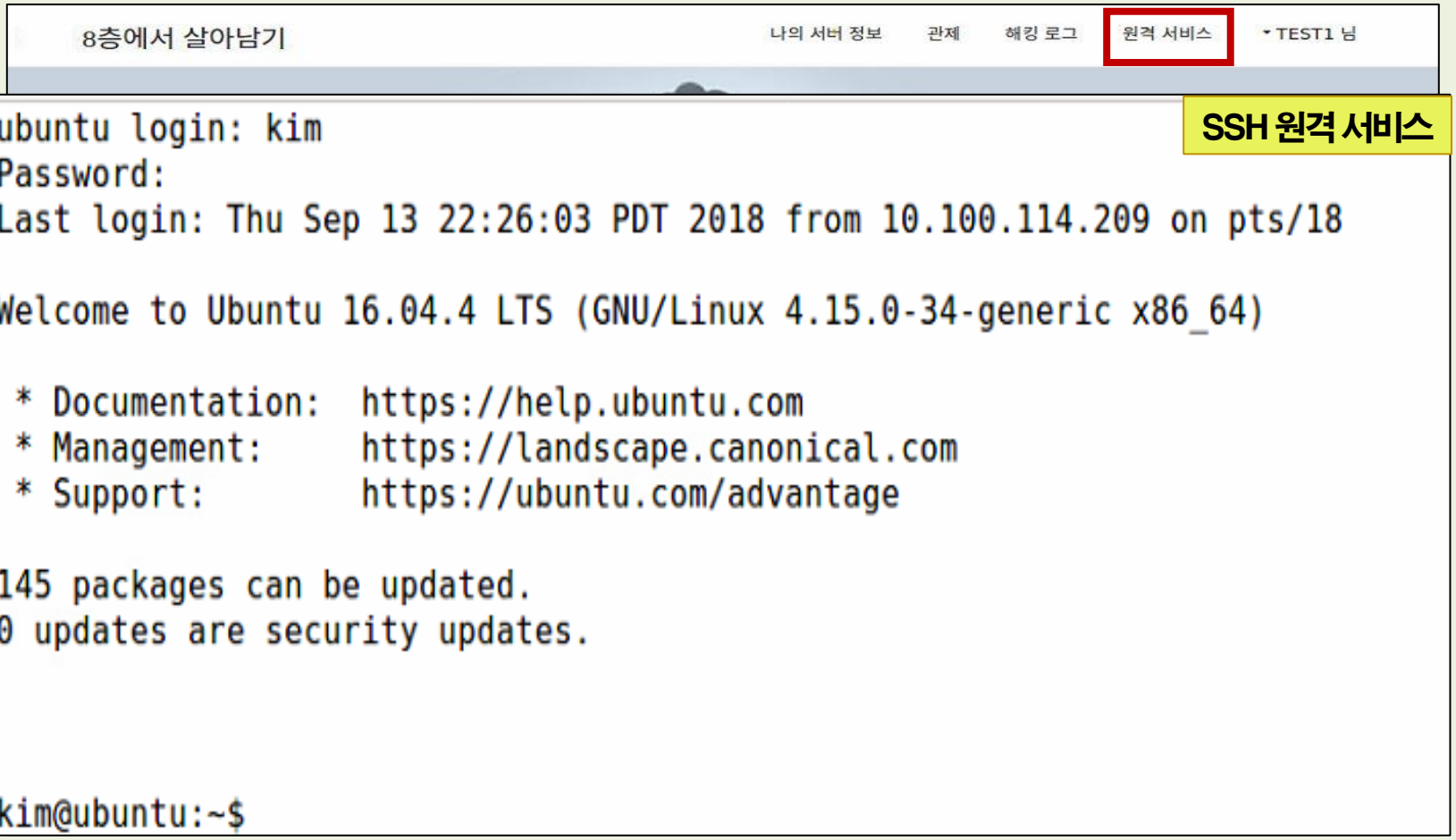



| ID | 탐지시간 | 프로토콜 | 출발지 | 목적지 | 탐지유형 |
|---|---|---|---|---|---|
| 3 | 05/13/2018-22:25:21.698802 | UDP | 192.168.233.2:53 | 192.168.233.148:33715 | SURICATA DNS Unsolicited response |
| 4 | 05/13/2018-22:25:21.698802 | UDP | 192.168.233.2:53 | 192.168.233.148:33715 | SURICATA DNS Unsolicited response |
| 5 | 05/13/2018-22:39:09.564876 | UDP | 168.126.63.1:53 | 192.168.233.148:49169 | SURICATA DNS Unsolicited response |
| 6 | 05/13/2018-23:12:27.585699 | UDP | 168.126.63.1:53 | 192.168.233.148:40729 | SURICATA DNS Unsolicited response |
| 7 | 05/13/2018-23:47:39.748349 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
| 8 | 05/13/2018-23:47:39.798710 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
| 9 | 05/13/2018-23:47:39.824377 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
| 10 | 05/13/2018-23:47:39.835400 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
| 11 | 05/13/2018-23:47:39.843432 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |
| 12 | 05/13/2018-23:47:39.852823 | TCP | 192.168.233.148:35662 | 112.175.50.171:80 | kimsungyonn rule |

**해킹 로그 메뉴에서 자신에게 유입된 유해트래픽 정보를 확인**

# 개발 결과 및 시스템운영(6/6)

## 메인 홈페이지(SSH)



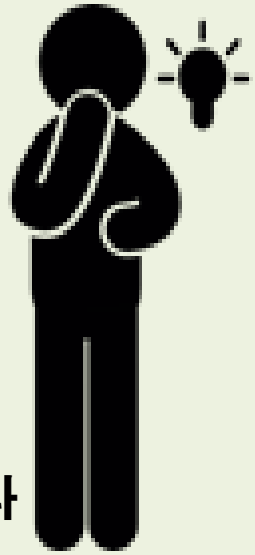

**원격 서비스 메뉴를 이용하여 쉽고 빠르게 자신의 서버에 접근가능**

# 결론 및 기대효과

## ❍ EAM 인증방식의 통합관제 시스템 개발 성과

- 인가된 각 서버마다 유해 트래픽을 탐지/분석하고 탐지된 유해 IP를 방화벽에 등록/관제하여 안전한 네트워크 환경을 제공

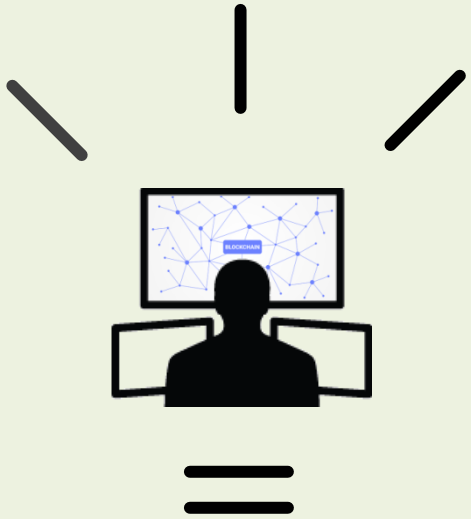
- SSO 방식의 인증체계를 구현하여 클라이언트의 작업 편리성을 높이고 차등적 권한관리가 가능함을 확인



## ❍ 기대 효과

- 시스템 개발과정에서 팀워크를 최대한 발휘함으로써 개인 역량보다 조직 역량을 직접 체험

- 특히 모든 조원들이 임무를 분담하여 인증체계와  웹서버 등을 구현, 실무 능력 향상시키는 계기




# Q&A

# Thank you