

# SIEM + 취약점 분석툴 연동을 이용한 실시간 보안관제

이슬기, 윤성준, 진영훈

2019.07.02

지도 교수 : 윤종문, 이병천

# 이슬기

콘솔 프로그램 작성

취약점 분석 툴 연동

PPT작성

# 윤성준

콘솔 프로그램 작성

SIEM분석 및 결과 값 연동

# 진영훈

SIEM분석 및 결과 값 연동

취약점 분석 툴 연동

PPT작성

## 요 약

---

오픈 소스 취약점 분석툴과 Open SIEM을 연동하여 이용하는 방법론을 탐구하였으며 **취약점에 대한 실시간 보안관제가 가능함**을 보여주었다.

- 취약점 분석 툴 : Nmap, Angry IP scanner, Suricata
- Open SIEM : Splunk

# 목 차

---

1. 서론
2. 요소기술
3. 프로젝트 설계 및 구현
4. 결론 및 기대 효과

# 1. 서론

1. SIEM이란

2. SIEM의 중요성

3. SIEM을 이용한 통합보안관제의 필요성

# SIEM이란?

Security Information & Event Management

- 빅데이터 기반으로, 기업 내에서 발생하는 모든 보안 이벤트를 통합 관리



## SIEM의 중요성

- 각 단위 제품별 다양한 관리방법을 사용하기 때문에 제품별 전문지식이 필요.
- 보안은 도입보다 관리와 효과적인 운영이 더 중요.
- 중앙 집중적 보안 관리는 효율적인 관리와 운영에 큰 기여를 할 수 있다.



**○ Hard to manage it !**

✓ 결론 : 통합보안관제의 기능을 수행하는 SIEM은 필수적이다.

## SIEM을 이용한 통합보안관제의 필요성

- SIEM의 보안영역에서 **취약점**에 대한 관리는 수행되고 있지 않다.
- 취약점 분석 툴과 통합보안관제가 연동된 모델
  - ✓ 로그기반의 **후탐지의 한계 극복**
  - ✓ **취약점**에 대한 부분까지 **탐지 가능**



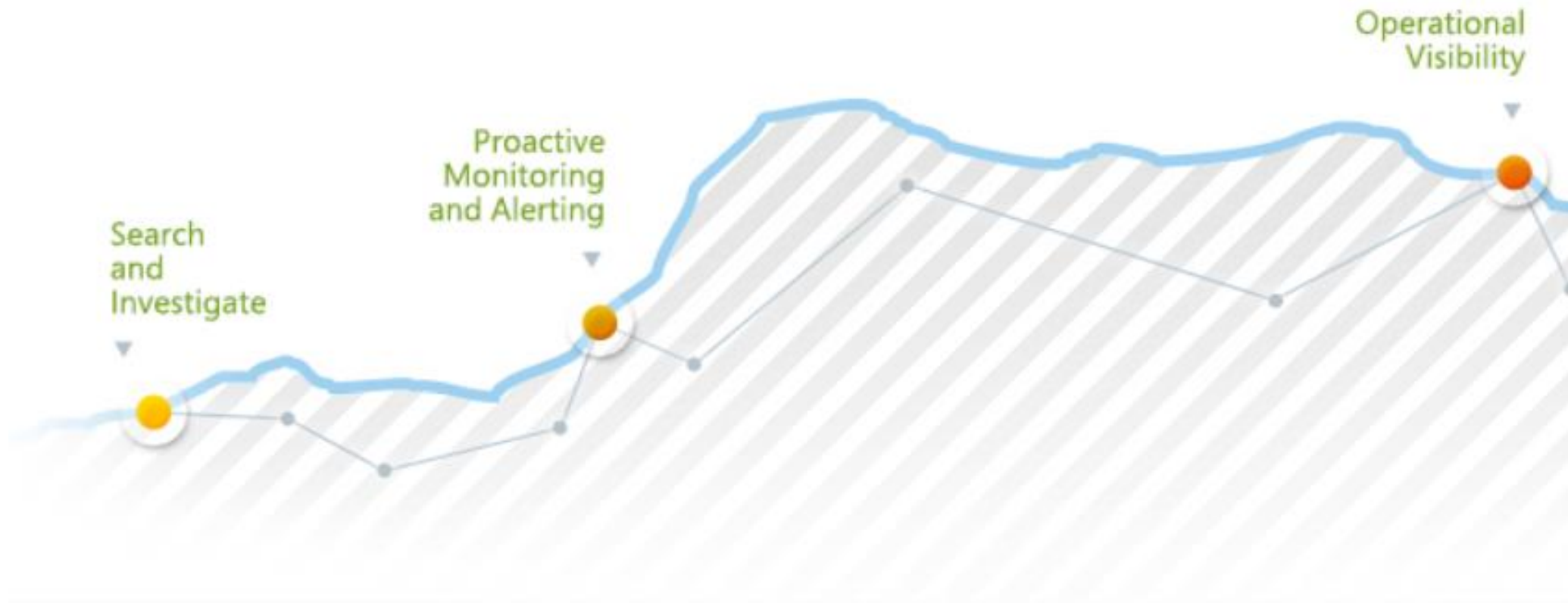


## 2. 요소 기술

1. Splunk
2. Angry IP scanner
3. Nmap
4. Suricata

## ○ Splunk

- ✓ IT분야에서 발생하는 다양한 데이터를 수집하고 모니터링하며 분류, 분석할 수 있는 엔진을 제공
- ✓ 다양한 분석을 통해 사용자가 원하는 대쉬보드를 자유롭게 생성



# 요소기술 - Splunk

## Why Splunk?



신속한 가치 실현



단일 플랫폼,  
다양한 활동



계층과 영역을 초월하는  
통합된 시각화



단순한 '조회'가 아닌  
'해답' 제공



데이터의 종류, 형태,  
규모와 무관

# 요소기술 - Splunk

## 특징 및 장점



### 가장 빠른 업무 개발

수집부터 보고까지의  
전 과정을 제공하는  
유일한 End-to-End 솔루션



### 무제한의 확장성

일별로 수십 MB ~ 수십 TB 까지  
다양한 규모의 데이터 처리 가능



### 서비스의 이해도

제한 없는 데이터의 상관관계  
분석 제공



### 통합 플랫폼 역할

표준 방식의 API를 제공하여  
기존의 프로그램에서도 사용

```
./splunk add monitor [result_file] -index main -sourcetype [사용자정의타입]
```



## ○ Angry IP Scanner

- ✓ 네트워크 내의 **Active장비 식별**
- ✓ Ping명령은 네트워크 확대 및 이상여부 판단에 대해 **한계적**

IP	Ping	Hostname	Ports [3+]	Web detect
195.80.116.226	[n/a]	[n/s]	[n/s]	[n/s]
195.80.116.227	9 ms	[n/a]	80,443	Resin/4.0.37
195.80.116.228	10 ms	[n/a]	80,443	[n/a]
195.80.116.229	9 ms	[n/a]	80,443	Apache
195.80.116.230	13 ms	mx3.rm.k.ee	[n/a]	[n/a]
195.80.116.231	10 ms	mx4.rm.k.ee	[n/a]	[n/a]
195.80.116.232	[n/a]	[n/s]	[n/s]	[n/s]
195.80.116.233	[n/a]	[n/s]	[n/s]	[n/s]
195.80.116.234	[n/a]	[n/s]	[n/s]	[n/s]
195.80.116.235	9 ms	[n/a]	80,443	[n/a]
195.80.116.236	[n/a]	[n/s]	[n/s]	[n/s]
195.80.116.237	[n/a]	[n/c]	[n/c]	[n/c]

```
./ipscan -s -o [Result_file_name] -f:rage [start_point] [end_point]
```



## ○ NMAP

✓ Open Status 의 **포트 식별**

### &Using Option

- sS : Syn Flag를 이용한 SYN-SCAN
- O : Operation System 식별
- oX: Xml format 으로 저장
- p-: 모든 포트에 대해 스캔
- T5 : Fast scan

**nmap -sS -O -p- -T5 -oX [result\_file\_name]**

```
Starting Nmap 6.00 ( http://nmap.org ) at 2012-05-17 12
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.00031s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh     OpenSSH_5.2p1 Debian 3ubuntu7
| ssh-hostkey: 1024-bit rsa1 ssh1:1c:0a:d6:67:54:9c:
|_ 2048 79:f8:9c:20:82:85:ec:
30/tcp    open  http-ti (Ubuntu)
|_ http-ti
9929/tcp  open
Device type: general purpose
Running: Linux 2.6.X|3.
OS CPE: cpe:/o:linux:kernel:2.6 cpe:/o:linux:kernel:3
OS details: Linux 2.6.32 - 2.6.39, Linux 2.6.38 - 3.0
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:kernel
```



## ○ Suricata

- ✓ 대표적인 공개 IDS
- ✓ 룰 기반 비정상 행위 탐지
- ✓ 효율적인 알고리즘으로  
비정상 행위 빠르게 식별

```
root@kali: /etc/suricata
File Edit View Search Terminal Help
--user <user> : run suricata as this user after init
--group <group>[its... : run suricata as this group after init
--erf-in <path> : process an ERF file
--unix-socket[=<file>] : use unix socket to control suricata work
--set name=value : set a configuration value
Method: Deploy Configuration Add Remove
To run the engine with default configuration on interface eth0 with signature file "signatures.rules", r
un the command as:
suricata -c suricata.yaml -s signatures.rules -i eth0
root@kali: /etc/suricata# suricata -c suricata.yaml -s test.rules -i eth0
23/6/2019 -- 14:14:05 - <Notice> - This is Suricata version 4.1.2 RELEASE
23/6/2019 -- 14:14:05 - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the pattern tes
t.rules
23/6/2019 -- 14:14:06 - <Notice> - all 4 packet processing threads, 4 management threads initialized, en
gine started.
23/6/2019 -- 14:14:06 - <Error> - [ERRCODE: SC_ERR_APP_READ(191)] - Interface 'eth0' is down
23/6/2019 -- 14:14:06 - <Warning> - [ERRCODE: SC_ERR_APP_CREATE(190)] - Couldn't init AF_PACKET socket,
retrying soon
23/6/2019 -- 14:14:06 - <Error> - [ERRCODE: SC_ERR_APP_READ(191)] - Interface 'eth0' is down
23/6/2019 -- 14:14:06 - <Warning> - [ERRCODE: SC_ERR_APP_CREATE(190)] - Couldn't init AF_PACKET socket,
retrying soon
23/6/2019 -- 14:14:06 - <Error> - [ERRCODE: SC_ERR_APP_READ(191)] - Interface 'eth0' is down
23/6/2019 -- 14:14:06 - <Warning> - [ERRCODE: SC_ERR_APP_CREATE(190)] - Couldn't init AF_PACKET socket,
retrying soon
23/6/2019 -- 14:14:06 - <Error> - [ERRCODE: SC_ERR_APP_READ(191)] - Interface 'eth0' is down
23/6/2019 -- 14:14:06 - <Warning> - [ERRCODE: SC_ERR_APP_CREATE(190)] - Couldn't init AF_PACKET socket,
retrying soon
```

# 요소기술 - Suricata

## ○ Suricata

&test.rules 中

alert tcp any any -> any any

(msg:"tcp syn flooding"; flags:S;

threshold:type threshold, track

by src, count 50, seconds 1;

sid:10044;)

```
root@kali: /var/log/suricata
File Edit View Search Terminal Help
root@kali:/var/log/suricata# ls
eve.json fast.log stats.log suricata.log
root@kali:/var/log/suricata# tail fast.log
06/27/2019-04:53:05.404710  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2284 -> 172.20.10.7:0
06/27/2019-04:53:05.411038  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2334 -> 172.20.10.7:0
06/27/2019-04:53:05.415096  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2384 -> 172.20.10.7:0
06/27/2019-04:53:05.418549  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2434 -> 172.20.10.7:0
06/27/2019-04:53:05.425704  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2484 -> 172.20.10.7:0
06/27/2019-04:53:05.430191  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2534 -> 172.20.10.7:0
06/27/2019-04:53:05.432929  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2584 -> 172.20.10.7:0
06/27/2019-04:53:05.446216  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2634 -> 172.20.10.7:0
06/27/2019-04:53:05.451075  [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] {TCP} 172.20.10.5:2682 -> 172.20.10.7:0
06/27/2019-04:53:29.321753  [**] [1:10008:0] dns test [**] [Classification: (null)] [Priority: 3] {UDP} 172.20.10.7:35603 -> 172.20.10.1:53
root@kali:/var/log/suricata#
```

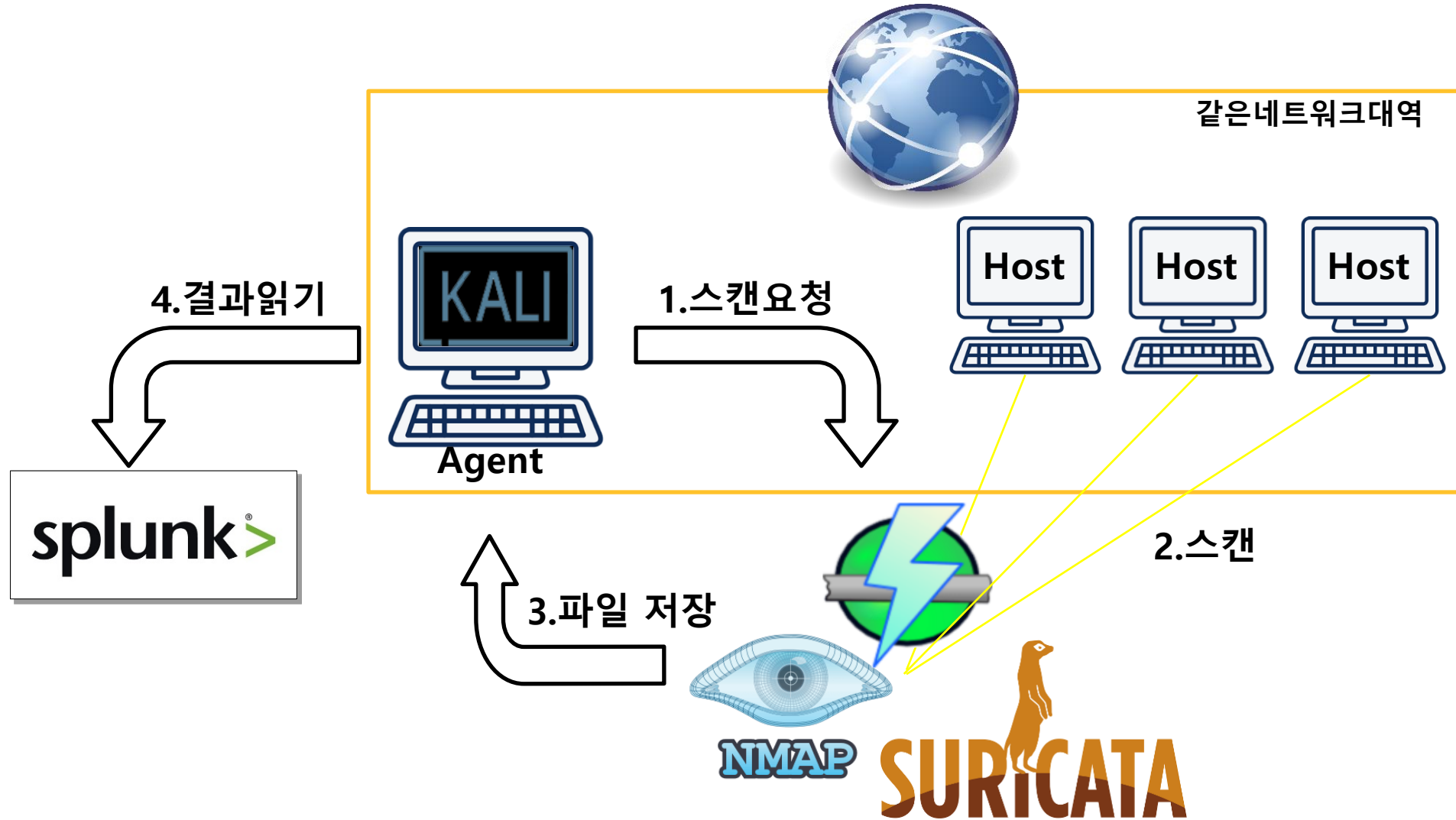
suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/test.rules -i eth0



### 3. 프로젝트 설계 및 구현

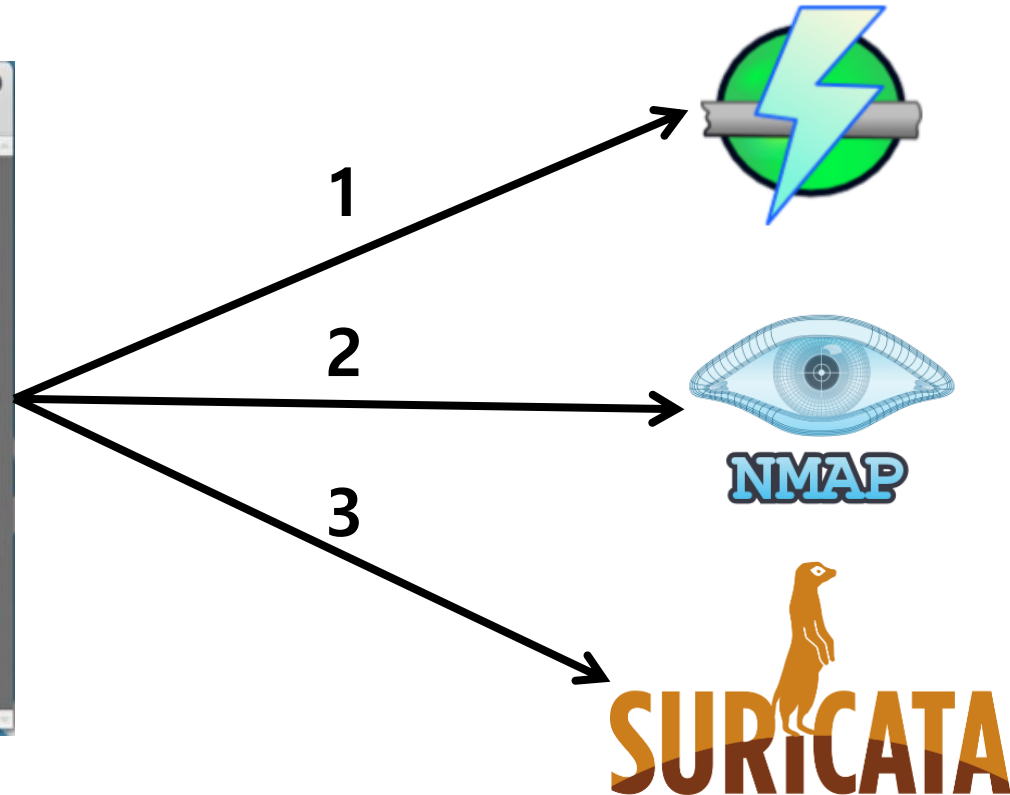
1. 전체 구상도
2. 데모 시나리오

# 전체 구상도



# SIEM\_Agent program

```
Terminal  
File Edit View Search Terminal Help  
----- Splunk & Network Vulnerabilities -----  
[1]: Active Host Scanning  
[2]: Regular Port Scanning  
[3]: IDS Starting  
[0 & Default]: EXIT  
Choose Number :|
```



Agent Source. Git : [https://github.com/JBU-seol/SIEM\\_Agent](https://github.com/JBU-seol/SIEM_Agent)

# 데모 시나리오 1. Angry IP Scanner 를 이용한 호스트 탐지

## 앱별 Wi-Fi 사용기록

최근에 Wi-Fi를 켜거나 끈 기록이 있는 앱을 확인합니다.

## Hotspot 2.0

자동으로 핫스팟 2.0 Wi-Fi 네트워크에 연결합니다.



## 네트워크 인증서 설치

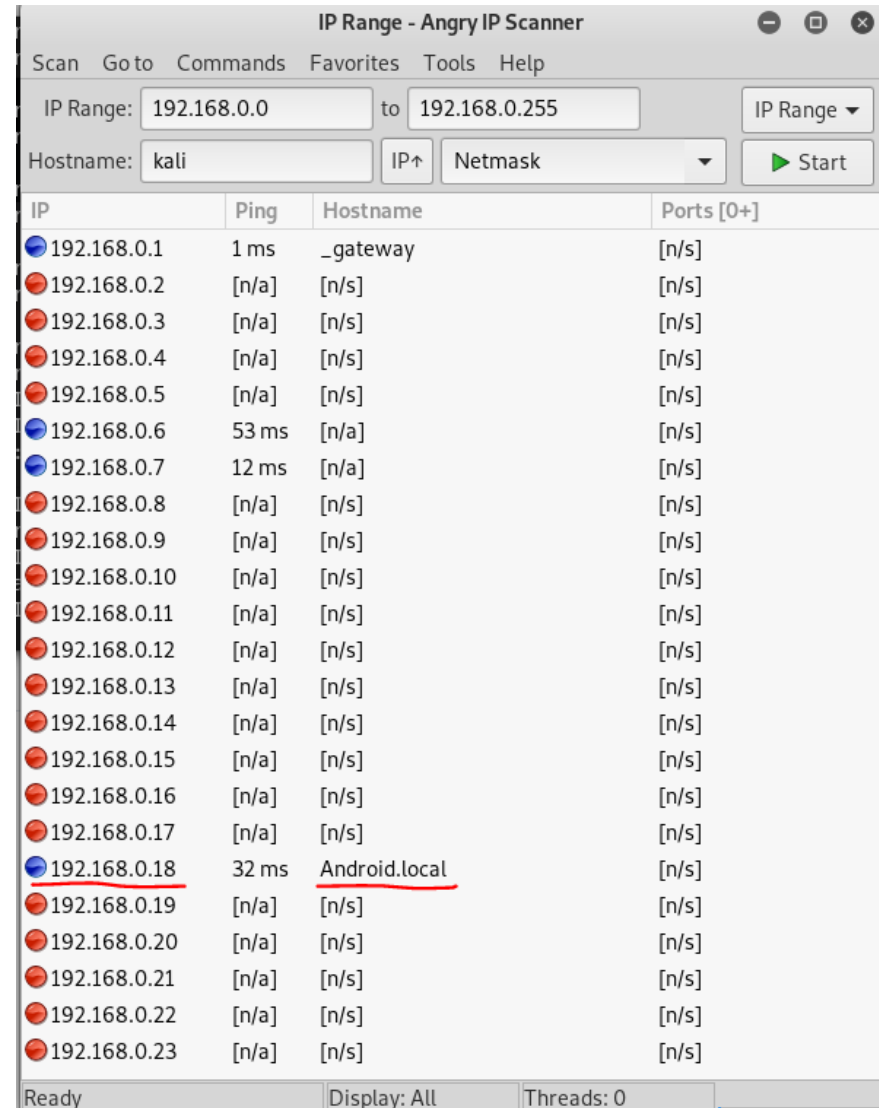
휴대전화에 있는 보안 인증서를 설치합니다.

MAC 주소 : 

IP 주소 : f3

192.168.0.18

Hostname을 통해 **장비명**까지  
정확하게 식별 할 수 있다.



IP	Ping	Hostname	Ports [0+]
192.168.0.1	1 ms	_gateway	[n/s]
192.168.0.2	[n/a]	[n/s]	[n/s]
192.168.0.3	[n/a]	[n/s]	[n/s]
192.168.0.4	[n/a]	[n/s]	[n/s]
192.168.0.5	[n/a]	[n/s]	[n/s]
192.168.0.6	53 ms	[n/a]	[n/s]
192.168.0.7	12 ms	[n/a]	[n/s]
192.168.0.8	[n/a]	[n/s]	[n/s]
192.168.0.9	[n/a]	[n/s]	[n/s]
192.168.0.10	[n/a]	[n/s]	[n/s]
192.168.0.11	[n/a]	[n/s]	[n/s]
192.168.0.12	[n/a]	[n/s]	[n/s]
192.168.0.13	[n/a]	[n/s]	[n/s]
192.168.0.14	[n/a]	[n/s]	[n/s]
192.168.0.15	[n/a]	[n/s]	[n/s]
192.168.0.16	[n/a]	[n/s]	[n/s]
192.168.0.17	[n/a]	[n/s]	[n/s]
192.168.0.18	32 ms	Android.local	[n/s]
192.168.0.19	[n/a]	[n/s]	[n/s]
192.168.0.20	[n/a]	[n/s]	[n/s]
192.168.0.21	[n/a]	[n/s]	[n/s]
192.168.0.22	[n/a]	[n/s]	[n/s]
192.168.0.23	[n/a]	[n/s]	[n/s]

## 데모 시나리오 2. NMAP 을 이용한 포트 변화 감지

i	시간	이벤트
>	19/06/27 5:01:04.000	<pre>... 10 lines omitted ... &lt;port protocol="tcp" portid="8065"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;/port&gt; &lt;port protocol="tcp" portid="8089"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;service name="unknown" method="table" conf="3"/&gt;&lt;/por t&gt; &lt;port protocol="tcp" portid="8191"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;service name="limnerpressure" method="table" conf="3" /&gt;&lt;/port&gt; ... 1 line omitted ... &lt;os&gt;&lt;portused state="open" proto="tcp" portid="8000"/&gt; &lt;portused state="closed" proto="tcp" portid="1"/&gt; 28개 행 모두 표시 host = kali   source = /opt/splunk/var/data/nmaplog   sourcetype = MyNmapLogs</pre>

i	시간	이벤트
>	19/06/27 5:02:07.000	<pre>... 10 lines omitted ... &lt;port protocol="tcp" portid="8000"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;service name="http-alt" method="table" conf="3"/&gt;&lt;/po rt&gt; &lt;port protocol="tcp" portid="8065"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;/port&gt; &lt;port protocol="tcp" portid="8089"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;service name="unknown" method="table" conf="3"/&gt;&lt;/por t&gt; &lt;port protocol="tcp" portid="8191"&gt;&lt;state state="open" reason="syn-ack" reason_ttl="64"/&gt;&lt;service name="limnerpressure" method="table" conf="3" /&gt;&lt;/port&gt; ... 1 line omitted ... &lt;os&gt;&lt;portused state="open" proto="tcp" portid="4040"/&gt; 22개 행 모두 표시 host = kali   source = /opt/splunk/var/data/nmaplog   sourcetype = MyNmapLogs</pre>

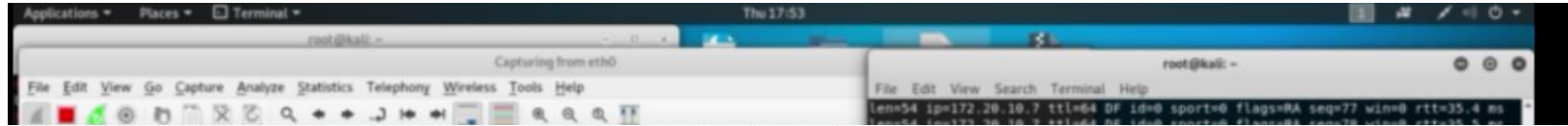
아파치 실행 전



아파치 실행 후

'#service apache2 start' 이후  
포트 4040이 open 됨을 splunk를 통해 실시간 감지

# 데모 시나리오 3. Suricata 를 이용한 **Syn-flooding attack** 탐지



i	시간	이벤트
>	19/06/27 4:53:05.451	06/27/2019-04:53:05.451075 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2682 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.446	06/27/2019-04:53:05.446216 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2634 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.432	06/27/2019-04:53:05.432929 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2584 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.430	06/27/2019-04:53:05.430191 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2534 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.425	06/27/2019-04:53:05.425704 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2484 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.418	06/27/2019-04:53:05.418549 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2434 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.415	06/27/2019-04:53:05.415096 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:1872 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.411	06/27/2019-04:53:05.411038 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2334 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27 4:53:05.404	06/27/2019-04:53:05.404710 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2284 -> 172.20.10.7:0 host = kali   source = /var/log/suricata/fast.log   sourcetype = snort_alert
>	19/06/27	06/27/2019-04:53:05.396229 [**] [1:10044:0] tcp syn flooding [**] [Classification: (null)] [Priority: 3] (TCP) 172.20.10.5:2234 -> 172.20.10.7:0

**Splunk에서 공격 탐지**

hping3 -c 500 -i u50 -S [target\_ip] → 1초에 50개의 SYN FLAG

## 4. 결론 및 향후 과제

## 결론

---

- ✓ 오픈소스 취약점 분석툴을 SIEM툴에 연계하여 실시간 취약점 보안관제를 운영할 수 있는 방안 제시
- ✓ 취약점을 진단하는 SIEM 체계는 다양하고 고도화된 위협에 대응할 수 있는 효과적인 정보보안 패러다임



## 향후 과제

- ✓ Agent 프로그램 기능 추가
  1. UI 추가
  2. Nessus 등 다른 취약점 분석 도구 연계
  3. 호스트 점검 결과를 서버로 전송/저장/분석하는 기능
- ✓ 분석 결과 통계화 방법론 모색
- ✓ SIEM을 통한 대응 방안 연구



Thank you

Q&A

