**Snort와 ELK를 활용한IDS구축 및 공격이벤트 탐지 솔루션**

|  |  |
| --- | --- |
| 팀명 | 이글루 |
| 팀장 | 임종철 |
| 팀원 | 권세아, 한현정 |





**개 정 이 력**

|  |  |  |  |
| --- | --- | --- | --- |
| 개정 번호 | 개정 내용 요약 | 추가/수정 항목 | 개정 일자 |
| 0.1 | 최종 제정 승인 | 초안 작성 | 2020.09.01 |
| 0.2 | 임종철 작성 | 내용 추가 | 2020.09.03 |
| 0.4 | 권세아 작성 | 내용 추가 | 2020.09.03 |
| 0.8 | 한현정 작성 | 내용 추가 | 2020.09.04 |
| 1.0 | 최종본 | 최종 수정/검토 | 2020.09.04 |

**문 서 규 칙**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Snort와 elk를 활용한 IDS구축 및 공격 이벤트 탐지 | | | ④ |
| Category | 첨부파일 버전 | 문서 최종 수정일 |
| ① | ② | ③ |

* 작성 및 확인은 Microsoft Word 2007으로 작성 되어 졌으며, Acrobat Reader로 읽는다.
* Category(①)에는 Manual, Utility, Tip, Analysis Report 로 구분하며, 기재된 정보가

Manual과 Utility가 혼합된 경우에는 "Manual + Utility" 라고 표기되며, 머리글의

Category에 해당 구분 정보를 표기된다.

* 본 분서의 저작권자인 팀의 로고와 이름이 표기된다. (④)
* 첨부 파일 버전(②)은 첨부 파일이 존재하는 경우에 기재되며, 첨부 파일의 버전이

표기된다. (유틸리티의 경우 최종 버전은 날짜 표기 대신 버전으로 대체한다)

* 문서 최종 수정일(③)에는 문서의 최종 수정날짜가 표기된다.

**목 차**

[1. 프로젝트 개요 9](#_Toc50123199)

[1.1 프로젝트 목표 9](#_Toc50123200)

[1.2 팀원 및 역할 9](#_Toc50123201)

[1.3 프로젝트 일정 10](#_Toc50123202)

[1.4 프로젝트 환경 11](#_Toc50123203)

[1.5 부분별 기대효과 12](#_Toc50123204)

[1.5.1 현황 (AS-IS) 12](#_Toc50123205)

[1.5.2 개선 및 기대효과 (TO-BE) 12](#_Toc50123206)

[1.6 Open Source 13](#_Toc50123207)

[1.6.1 Snort 13](#_Toc50123208)

[1.6.2 ELK Stack 14](#_Toc50123209)

[1.7 Cloud 환경에서 Infra Structure 구축 17](#_Toc50123210)

[2. 프로젝트 내용 18](#_Toc50123211)

[2.1 File Inclusion 18](#_Toc50123212)

[2.1.1 정의 18](#_Toc50123213)

[2.1.2 공격 18](#_Toc50123214)

[2.1.3 룰 정책 추가 20](#_Toc50123215)

[2.1.4 탐지 확인 21](#_Toc50123216)

[2.2 File Upload 22](#_Toc50123217)

[2.2.1 정의 22](#_Toc50123222)

[2.2.2 공격 22](#_Toc50123223)

[2.2.3 룰 정책 추가 23](#_Toc50123224)

[2.2.4 탐지 확인 23](#_Toc50123225)

[2.3 SQL Injection 25](#_Toc50123226)

[2.3.1 정의 25](#_Toc50123227)

[2.3.2 공격 25](#_Toc50123228)

[2.3.3 룰 정책 추가 26](#_Toc50123229)

[2.3.4 탐지 확인 26](#_Toc50123230)

[2.4 XSS(Cross Site Script) 27](#_Toc50123231)

[2.4.1 정의 27](#_Toc50123232)

[2.4.2 공격 27](#_Toc50123233)

[2.4.3 룰 정책 추가 30](#_Toc50123234)

[2.4.4 탐지 확인 31](#_Toc50123235)

[2.5 OS Command Injection 32](#_Toc50123236)

[2.5.1 정의 32](#_Toc50123237)

[2.5.2 공격 32](#_Toc50123238)

[2.5.3 룰 정책 추가 33](#_Toc50123239)

[2.5.4 탐지 확인 33](#_Toc50123240)

[2.6 Brute Force 34](#_Toc50123241)

[2.6.1 정의 34](#_Toc50123242)

[2.6.2 공격 34](#_Toc50123243)

[2.6.3 룰 정책 추가 35](#_Toc50123244)

[2.6.4 탐지 확인 35](#_Toc50123245)

[2.7 UDP Flooding 36](#_Toc50123246)

[2.7.1 정의 36](#_Toc50123247)

[2.7.2 공격 36](#_Toc50123248)

[2.7.3 룰 정책 추가 36](#_Toc50123249)

[2.7.4 탐지 확인 37](#_Toc50123250)

[2.8 ICMP Flooding 38](#_Toc50123251)

[2.8.1 정의 38](#_Toc50123252)

[2.8.2 룰 정책 추가 38](#_Toc50123253)

[2.8.3 탐지 확인 39](#_Toc50123254)

[2.9 Syn flooding 40](#_Toc50123255)

[2.9.1 정의 40](#_Toc50123256)

[2.9.2 공격 40](#_Toc50123257)

[2.9.3 룰 정책 추가 41](#_Toc50123258)

[2.9.4 탐지 확인 41](#_Toc50123259)

[2.10 SYN-ACK Flooding 42](#_Toc50123260)

[2.10.1 정의 42](#_Toc50123261)

[2.10.2 공격 42](#_Toc50123262)

[2.10.3 룰 정책 추가 43](#_Toc50123263)

[2.10.4 탐지 확인 43](#_Toc50123264)

[2.11 Land Attack 44](#_Toc50123265)

[2.11.1 정의 44](#_Toc50123266)

[2.11.2 공격 44](#_Toc50123267)

[2.11.3 룰 정책 추가 45](#_Toc50123268)

[2.11.4 탐지 확인 45](#_Toc50123269)

[2.12 NMAP XMAS Scan 46](#_Toc50123270)

[2.12.1 정의 46](#_Toc50123271)

[2.12.2 공격 46](#_Toc50123272)

[2.12.3 룰 정책 추가 47](#_Toc50123273)

[2.12.4 탐지 확인 47](#_Toc50123274)

[3. 결론 48](#_Toc50123275)

[4. 추후과제 및 후기 49](#_Toc50123276)

**표 목 차**

[[표 1‑1] 팀원 및 역할 7](#_Toc441060694)

[[표 1‑2] 프로젝트 일정 8](#_Toc441060695)

[[표 1‑3] 프로젝트 환경 9](#_Toc441060696)

[[표 2‑1] 공격을 알려 주는 기능 List 20](#_Toc441060697)

**그 림 목 차**

[[그림 1‑1] Snort 로고 13](#_Toc50123277)

[[그림 1‑2] ELK Stack 구조도 14](#_Toc50123278)

[[그림 1‑3] ELK Stack의 대쉬보드 15](#_Toc50123279)

[[그림 1‑4] 실제 구현한 Kibana 화면 15](#_Toc50123280)

[[그림 1‑5] Elastic Search 에 저장된 데이터 16](#_Toc50123281)

[[그림 1‑6] GCP 클라우드 환경 17](#_Toc50123282)

[[그림 2‑1] 취약한 변수값 확인 18](#_Toc50123283)

[[그림 2‑2] 취약한 변수값을 활용하여 파일 확인 19](#_Toc50123284)

[[그림 2‑3] File Inclusion 공격을 통해 서버의 /etc/passwd 확인 모습 19](#_Toc50123285)

[[그림 2‑4] File Inclusion 공격을 통해 서버의 info.php 파일 확인 모습 20](#_Toc50123286)

[[그림 2‑5] File Inclusion 공격 로그 확인 20](#_Toc50123287)

[[그림 2‑6] File Inclusion 탐지 룰 정책 21](#_Toc50123288)

[[그림 2‑7] File Inclusion 공격 로그 확인 21](#_Toc50123289)

[[그림 2‑8] 웹쉘 업로드 22](#_Toc50123290)

[[그림 2‑9] 웹셀 실행 모습 23](#_Toc50123291)

[[그림 2‑10] File Upload 공격 로그 확인 23](#_Toc50123292)

[[그림 2‑11] File Upload 룰 정책 추가 23](#_Toc50123293)

[[그림 2‑12] File Upload 공격 로그 확인 24](#_Toc50123294)

[[그림 2‑13] sql Injection 실행 25](#_Toc50123295)

[[그림 2‑14] SQL 룰 추가 26](#_Toc50123296)

[[그림 2‑15] SQL Injection 공격 로그 확인 26](#_Toc50123297)

[[그림 2‑16] XSS Reflected 공격 27](#_Toc50123298)

[[그림 2‑17] XSS Stored 공격 28](#_Toc50123299)

[[그림 2‑18] DOM 기반 사용 확인 28](#_Toc50123300)

[[그림 2‑19] DOM 기반임을 확인할 수 있는 페이지 29](#_Toc50123301)

[[그림 2‑20] DOM 기반 XSS 공격 29](#_Toc50123302)

[[그림 2‑21] XSS Reflected 공격 로그 확인 30](#_Toc50123303)

[[그림 2‑22] SQL Injection룰 세팅 30](#_Toc50123304)

[[그림 2‑23] XSS(Cross Site Script) 공격 로그 확인 31](#_Toc50123305)

[[그림 2‑24] OS Command Injection 공격 32](#_Toc50123306)

[[그림 2‑25] OS Command Injectino 공격 실행 결과 32](#_Toc50123307)

[[그림 2‑26] OS Command Injection 공격 탐지 룰 33](#_Toc50123308)

[[그림 2‑27] OS Command Injection 공격 탐지 모습 33](#_Toc50123309)

[[그림 2‑28] Brute Force를 위한 로그인 34](#_Toc50123310)

[[그림 2‑29] Burf Suite로 패킷을 캡쳐하여 Brute Force 공격을 실행하는 모습 35](#_Toc50123311)

[[그림 2‑30] Brute Force 공격 탐지 룰 35](#_Toc50123312)

[[그림 2‑31] Brute Force 공격 탐지 모습 35](#_Toc50123313)

[[그림 2‑32] 칼리 리눅스에서 UDP Flooding 공격 실행 모습 36](#_Toc50123314)

[[그림 2‑33] UDP 패킷 캡쳐 모습 36](#_Toc50123315)

[[그림 2‑34] UDP Flooding 공격 탐지 룰 37](#_Toc50123316)

[[그림 2‑35] UDP Flooding 공격 탐지 모습 37](#_Toc50123317)

[[그림 2‑36] 칼리 리눅스에서 ICMP Flooding 공격 실행 모습 38](#_Toc50123318)

[[그림 2‑37] ICMP 패킷 캡쳐 모습 38](#_Toc50123319)

[[그림 2‑38] ICMP Flooding 공격 탐지 룰 39](#_Toc50123320)

[[그림 2‑39] ICMP Flooding 공격 탐지 모습 39](#_Toc50123321)

[[그림 2‑40] 칼리리눅스에서 SYN Flooding 공격 모습 40](#_Toc50123322)

[[그림 2‑41] SYN Flooding 공격 패킷 캡쳐 40](#_Toc50123323)

[[그림 2‑42] SYN Flooding 공격 탐지 룰 41](#_Toc50123324)

[[그림 2‑43] SYN Flooding 공격 탐지 모습 41](#_Toc50123325)

[[그림 2‑44] 칼리 리눅스에서 SYN-ACK Flooding 공격 42](#_Toc50123326)

[[그림 2‑45] SYN-ACK Flooding 공격 패킷 42](#_Toc50123327)

[[그림 2‑46] SYN-ACK Flooding 공격 패킷 2 43](#_Toc50123328)

[[그림 2‑47] SYN-ACK Flooding 공격 탐지 룰 43](#_Toc50123329)

[[그림 2‑48] SYN-ACK Flooding 공격 탐지 모습 43](#_Toc50123330)

[[그림 2‑49] 칼리에서 Land Attack 공격 44](#_Toc50123331)

[[그림 2‑50] Land Attack 패킷 캡쳐 44](#_Toc50123332)

[[그림 2‑51] Land Attack 룰 추가 45](#_Toc50123333)

[[그림 2‑52] Land Attack 탐지 모습 45](#_Toc50123334)

[[그림 2‑53] kail에서 NMAP XMAS 공격 모습 46](#_Toc50123335)

[[그림 2‑54] 와이어샤크에서 XMAS 패킷 캡쳐 46](#_Toc50123336)

[[그림 2‑55] nmap xmas 정책 추가 47](#_Toc50123337)

[[그림 2‑56] NMAP XMAS 탐지 47](#_Toc50123338)

# 프로젝트 개요

* 1. 프로젝트 목표

오픈 소스를 활용하여 웹과 네트워크 취약점에 대응하는 방어 솔루션을 구축한다. 또한 공격 이벤트를 탐지하고, 시각화하여 웹과 네트워크의 보안 이벤트에 빠르게 대응할 수 있도록 한다. IDS구축에 사용되는 오픈 소스는 Snort와 ELK Stack (Elasticsearch, Logstash, Kibana + Beats)이다. 공격 이벤트 탐지 시 오탐과 미탐을 줄이기 위한 규칙(rule) 설정에 주안점을 두었다. 이를 통하여 보다 신뢰도 높은 이벤트 탐지 솔루션을 제안하는 것을 목표로 한다.

* 1. 팀원 및 역할

아래 [표 1-1]과 같다. (이름 가나다 순)

|  |  |
| --- | --- |
| 이름 | 역할 |
| 권세아 | - 공격 방법 연구  - 방어규칙 연구 및 작성  - Snort와 ELK Stack을 활용한 IDS 구축 |
| 임종철 | - GCP 상의 네트워크 및 서버 구축  - 방어규칙 연구 및 작성 |
| 한현정 | - 방어규칙 연구 및 작성  - 보고서 및 발표자료 작성 |

[표 1‑1] 팀원 및 역할

* 1. 프로젝트 일정

아래 [표 1-2]와 같다.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1일차 | | 2일차 | | 3일차 | | 4일차 | | 5일차 | |
| 자료조사 |  |  |  |  |  |  |  |  |  |  |
| 주제선정 |  |  |  |  |  |  |  |  |  |  |
| 환경구축 |  |  |  |  |  |  |  |  |  |  |
| 공격방법연구 |  |  |  |  |  |  |  |  |  |  |
| 방어규칙연구 |  |  |  |  |  |  |  |  |  |  |
| 대시보드 |  |  |  |  |  |  |  |  |  |  |
| 문서작성 |  |  |  |  |  |  |  |  |  |  |
| 종료 및 발표 |  |  |  |  |  |  |  |  |  |  |

[표 1‑2] 프로젝트 일정

* 1. 프로젝트 환경

아래 [표 1-3]의 프로그램들을 이용하여 프로젝트를 진행하였다.

|  |  |  |
| --- | --- | --- |
| **운영 체제**  **(Web Server, Snort, ELK)** |  | Ubuntu 16.04 LTS |
| **인프라 구축**  **(Web Server, Snort, ELK)** |  | Google Cloud |
| **웹 서버 운영**  **(Web server)** |  | Apache2 |
|  | Php |
|  | MariaDB |
| **웹서버의 취약한**  **어플리케이션** |  | DVWA |
| **악의적 사용자** |  | Kali Linux 2.3.0.5 |

[표 1‑3] 프로젝트 환경

* 1. 부분별 기대효과
     1. 현황 (AS-IS)
* 비정상적인 공격이 있을 경우 실시간 탐지 및 차단이 어렵다.
* 룰을 지속적으로 업데이트하지 않아, 오탐과 미탐의 가능성이 높다.
* 서비스에 남은 로그를 시각화할 수 없어, 분석하는 데에 한계가 있다.  
  + 1. 개선 및 기대효과 (TO-BE)

- Rule 정책으로 인한 패턴 적용으로 비정상적인 공격 인식 가능하다.

- 실시간으로 공격 패턴을 탐지할 수 있고, 심각한 공격의 경우 차단이 가능하다.

- 로그를 대쉬보드에서 한 눈에 시각화해서 보고, 확인할 수 있다.

* 1. Open Source
     1. Snort

1. 개요

* 실시간으로 트래픽 분석과 패킷을 수집하는 네트워크 침입 탐지 시스템

1. 기능

* 침입탐지 Rule에 일치되는 네트워크 트래픽을 감시하고 기록 및 경고
* 프로토콜 분석, 패킷 내용조사, 패턴 매칭
* 버퍼오버플로우나 스텔스 포트스캔 등 다양한 공격을 탐지

1. 구조

* Sniffer : 패킷 스니핑(packet sniffing)을 이용한 IDS 시스템
* 패킷 캡쳐를 위해 libpcap 사용
* 룰 기반(rule based)으로 탐지 엔진을 갖추고 있음
* 일반 로그, tcp dump 포맷 로그, 실시간 경고(alert), 파일일, 윈도우 팝업 등 여러 포맷을 통해 결과값 제공



[그림 1‑1] Snort 로고

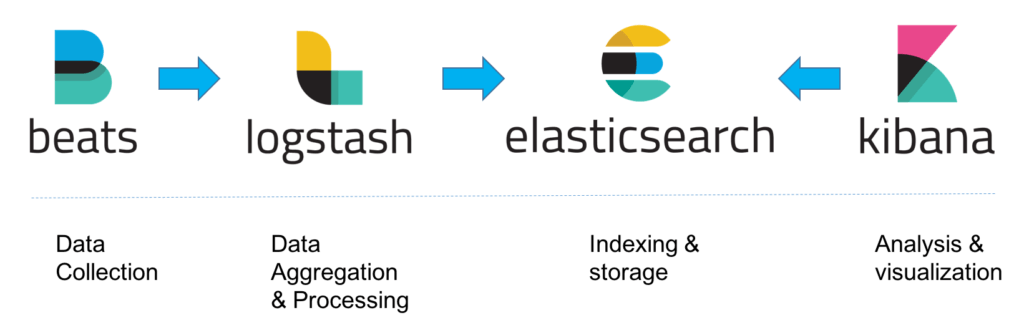
* + 1. ELK Stack

1. 개요

* ELK는 Elastic Search, Logstash, Kibana의 앞 글자를 딴 단어로, 접근성과 용이성이 좋은 Log 및 데이터 분석 도구
* ELK 솔루션에 Beats가 추가되면서 ELK Stack이라고 불림

1. 기능

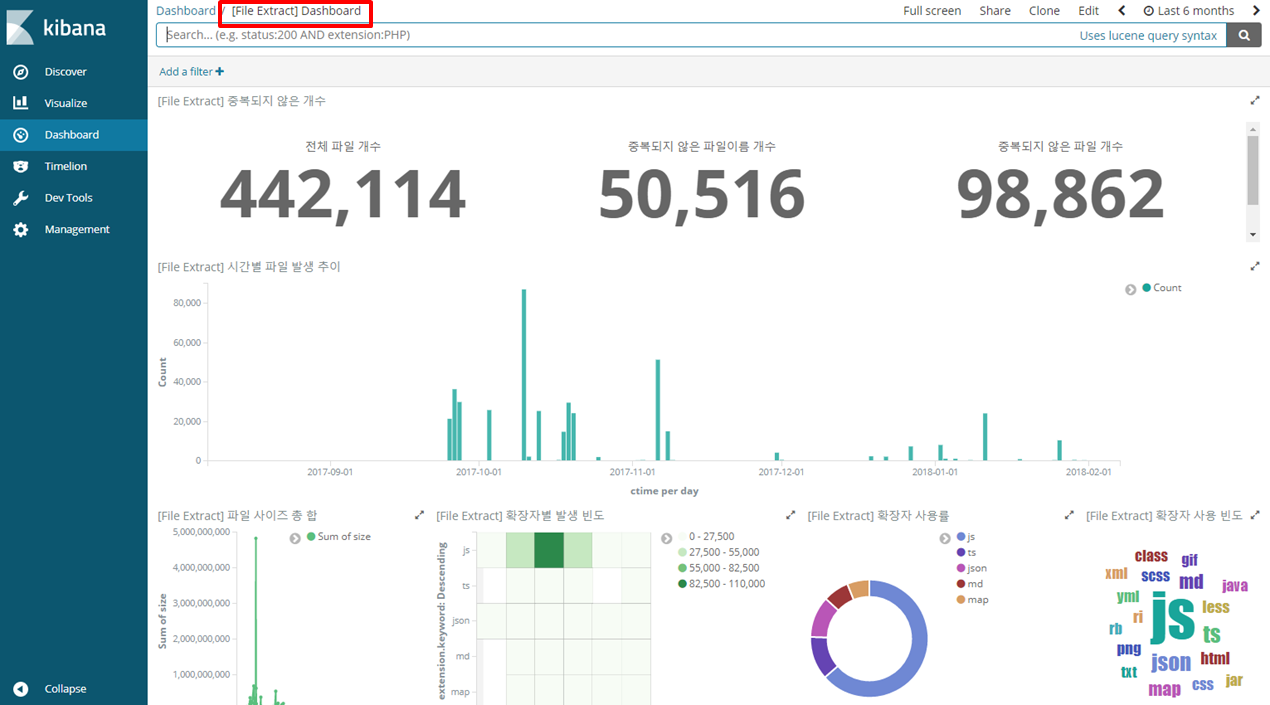
* ElasticSearch는 분석 및 저장한다. Logstash를 통해 수신된 데이터를 저장소에 저장하는 역할을 담당한다. 정형, 비정형, 위치정보, 메트릭 등 원하는 방법으로 다양한 유형의 검색을 수행하고 결합한다.
* Logstash는 로그 및 데이터를 수집한다. 다양한 소스에서 동시에 데이터를 수집하고 변환하여 보관소로 보낸다. 수집할 로그를 선정해서, 지정된 서버(ElasticSearch)에 인덱싱하여 전송한다.
* Kibana는 이러한 데이터들을 시각화하는 도구이며, 대쉬보드를 제공한다.
* 시각화를 담당하는 HTML + Javascript 엔진이다.
* Beats는 다양한 유형의 데이터를 수집하여, ElasticSearch 혹은 Logstash에 전송하는 소스 데이터를 발송한다.



[그림 1‑2] ELK Stack 구조도

1. 시각화 및 보고

* 운영 또는 보안 요구 사항에 따라 조정된 사용자 지정 대시보드 및 보고서에서 동향 및 특징을 초보자도 보기 간편하게 시각화

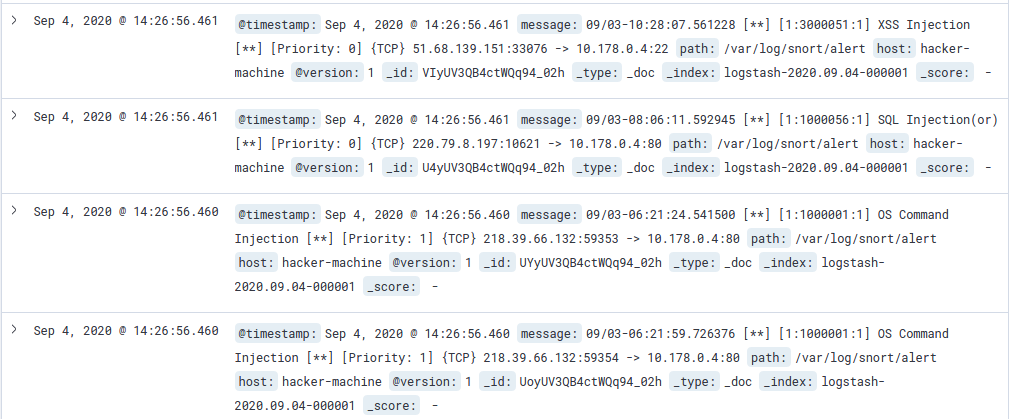


[그림 1‑3] ELK Stack의 대쉬보드

1. 실제 구현 화면



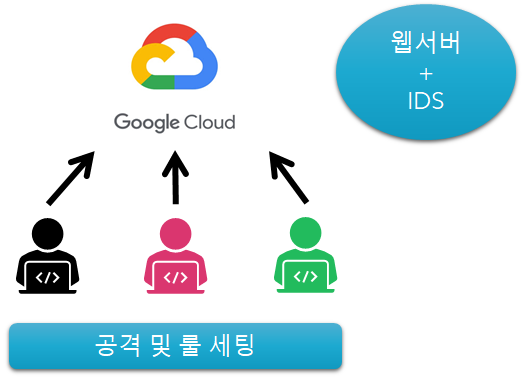
[그림 1‑4] 실제 구현한 Kibana 화면



[그림 1‑5] Elastic Search 에 저장된 데이터

* 1. Cloud 환경에서 Infra Structure 구축

최근 클라우드 환경 트랜드에 발맞추어, 개발 환경을 구글 클라우드 플랫폼을 활용하였다. 피해자 PC를 GCP EC2로 구축하였다. 상세 스펙은 아래와 같다. 해당 PC에 Snort와 ELK Stack을 함께 구축하여 피해 로그를 확인할 수 있도록 했다.



[그림 1‑6] GCP 클라우드 환경

# 프로젝트 내용

* 1. File Inclusion
     1. 정의

PHP로 이루어진 웹페이지에서 파라미터 값을 정확하게 검사하지 않는 경우에 발생하는 공격 및 취약점이다 php의 inclusion 함수를 악용한다.

RFI(Remote File Inclusion)와 LFI(Local File Inclusion)로 나뒤는데, RFI는 공격자 서버에 존재하는 파일을 가져와 이용하는 공격 기법이고, LFI는 공격 대상 서버의 내부에 존재하는 파일을 이용하는 공격 기법이다

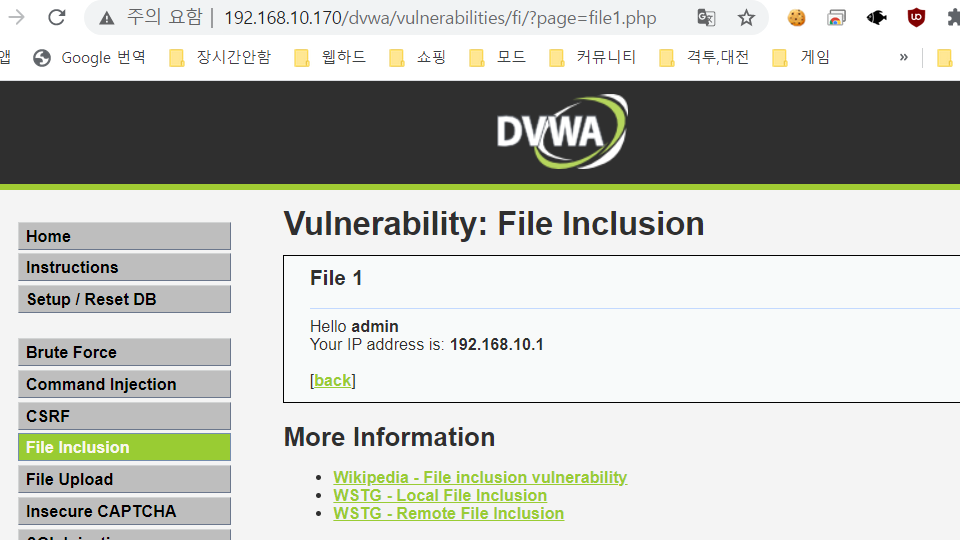
* + 1. 공격

**[Step #1]** 변수값을 이용하는 페이지를 확인



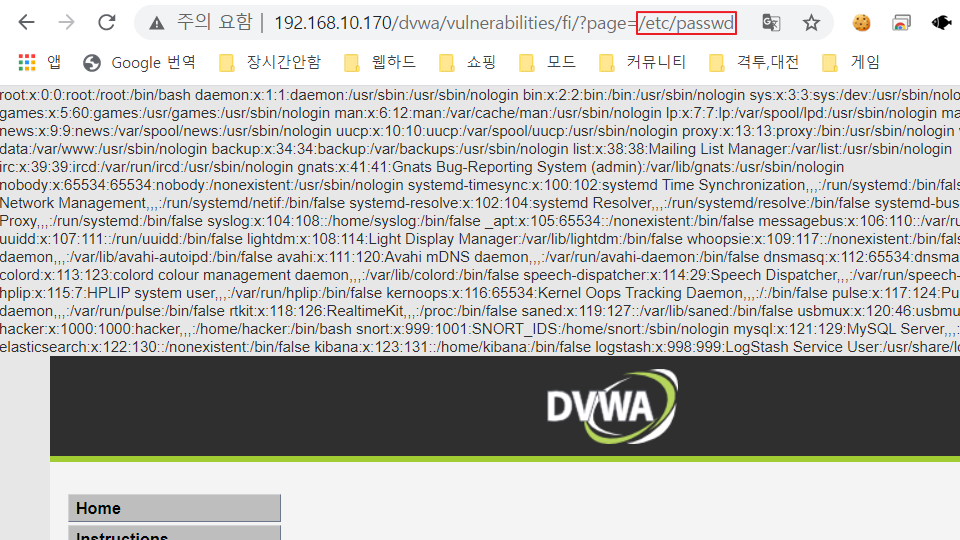
[그림 2‑1] 취약한 변수값 확인

**[Step #2]** 변수값을 사용해 사이트 내 파일을 보여줌



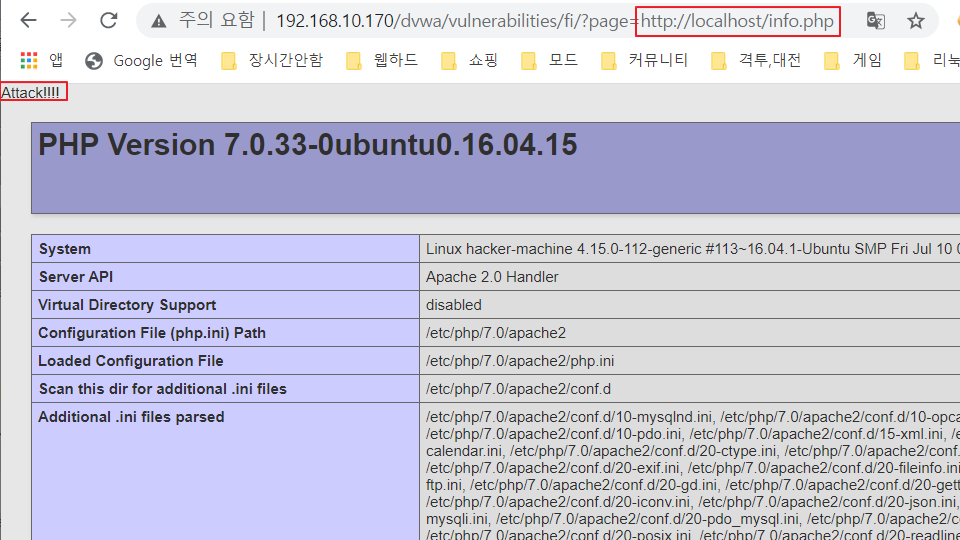
[그림 2‑2] 취약한 변수값을 활용하여 파일 확인

**[Step #3]** LFI 공격에 성공해, 서버내 '/etc/passwd' 파일 확인



[그림 2‑3] File Inclusion 공격을 통해 서버의 /etc/passwd 확인 모습

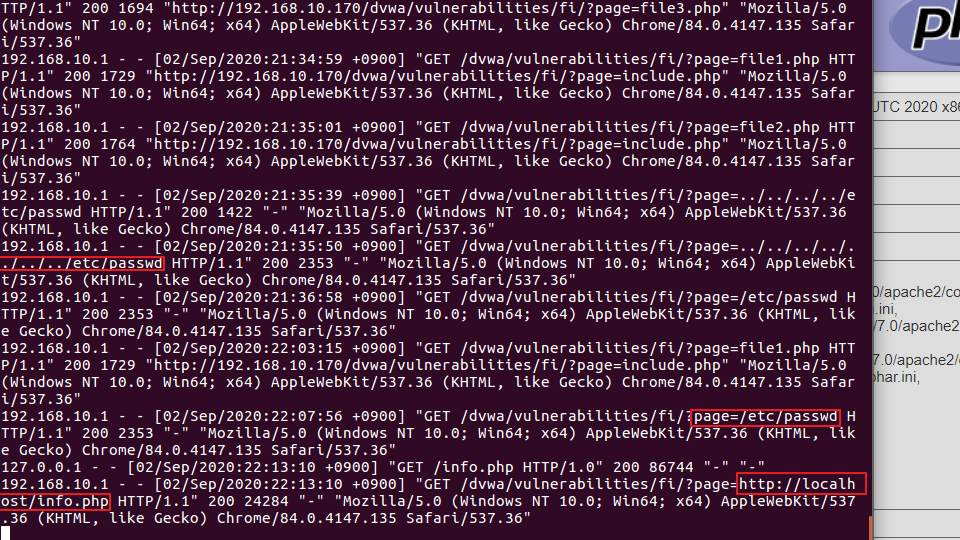
**[Step #4]** RFI 공격에 성공해, 공격자 서버에 존재하는 파일을 가져와 보여주고 있음 (http://localhost : 공격자의 서버라고 가정)



[그림 2‑4] File Inclusion 공격을 통해 서버의 info.php 파일 확인 모습

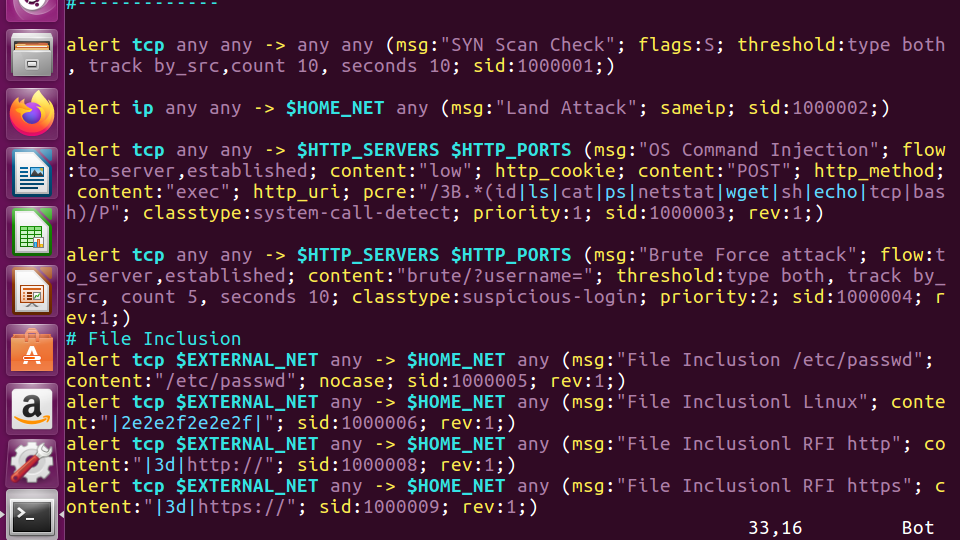
* + 1. 룰 정책 추가

**[Step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑5] File Inclusion 공격 로그 확인

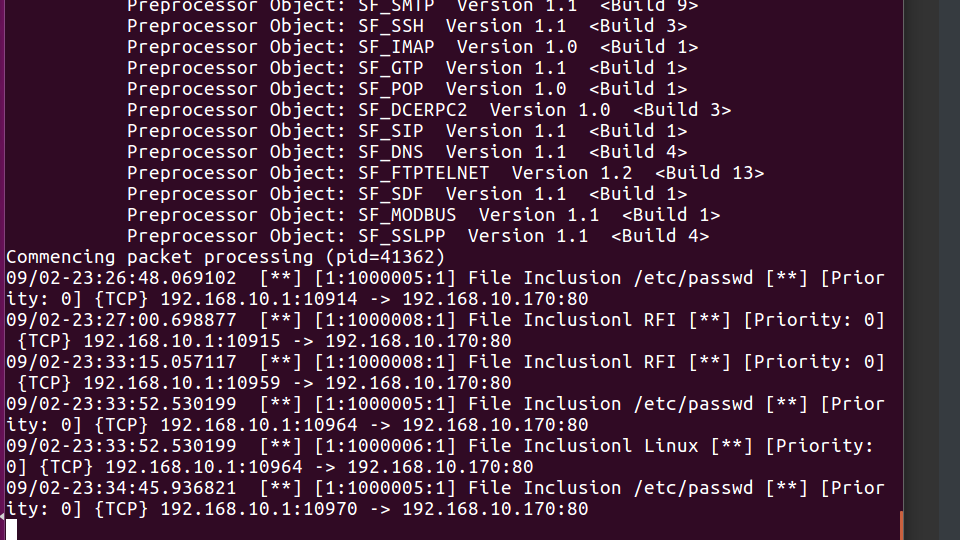
**[Step #2]** 확인한 패턴과 공격에 대한 정보를 분석 후, 알맞은 룰 정책 추가



[그림 2‑6] File Inclusion 탐지 룰 정책

* + 1. 탐지 확인

**[step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑7] File Inclusion 공격 로그 확인

* 1. File Upload

2. 2. 1. 정의

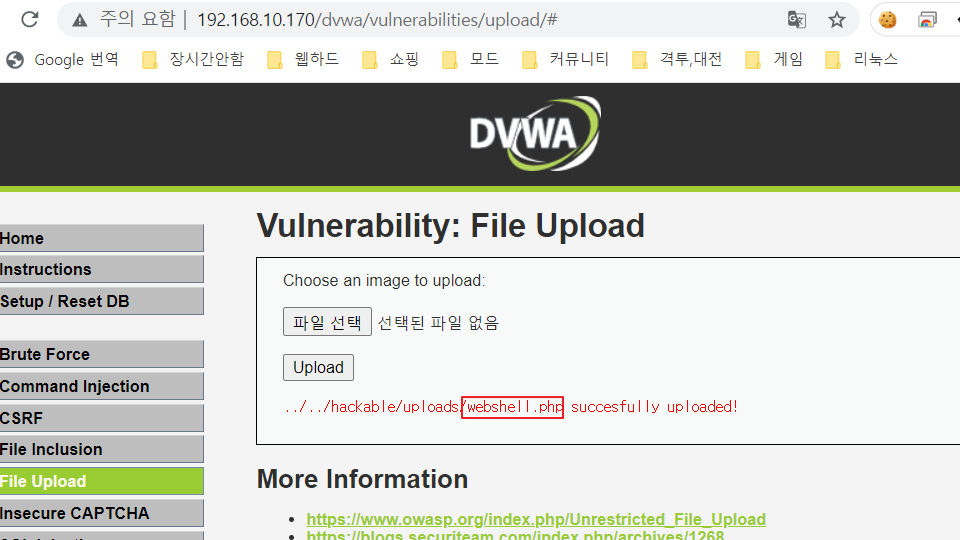
파일 업로드 기능이 존재하는 웹 어플리케이션에서 확장자 필터링이 제대로 이루어지지 않았을 경우 공격자가 악성 스크립트 파일(웹쉘)을 업로드하여 해당 웹쉘을 통해 원격에서 시스템을 제어할 수 있는 취약점이다.

공격자는 취약점을 이용하여 시스템 명령 수행 등 시스템을 제어할 수 있으며 웹 페이지 변조 등의 공격도 수행할 수 있다

웹쉘은 악의적인 목적으로 웹서버 내 임의의 명령을 실행할 수 있는 서버 스크립트 파일이며, 다양한 언어(ASP, PHP, JSP 등)로 만들어진다.

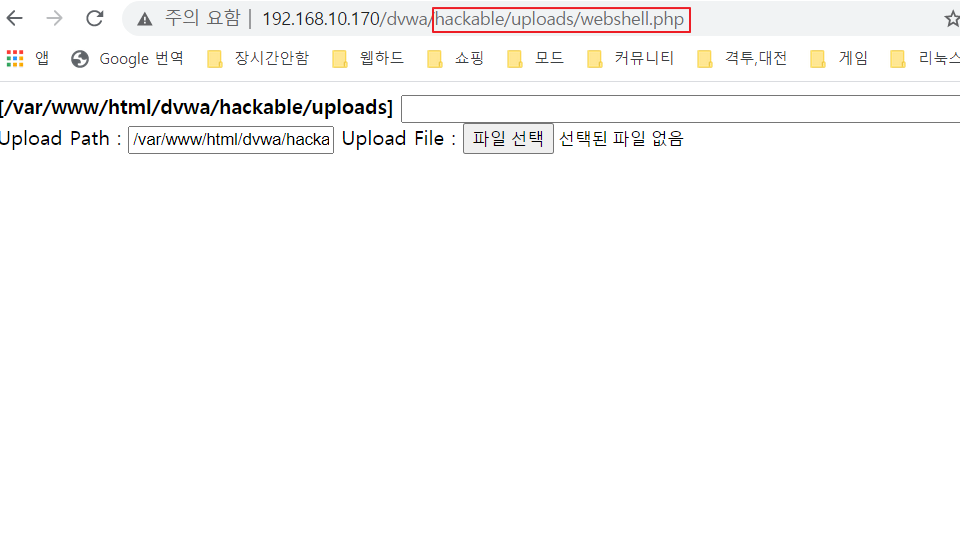
* + 1. 공격

**[Step #1]** 웹셀이 업로드 가능한 것을 확인



[그림 2‑8] 웹쉘 업로드

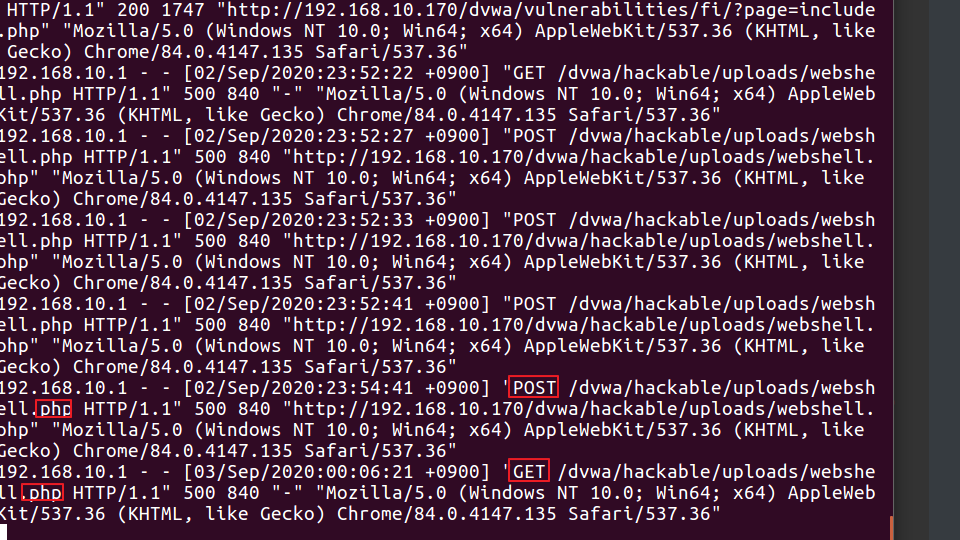
**[Step #2]** 업로드시 보여주는 경로를 사용해, 업로드한 웹셀 실행



[그림 2‑9] 웹셀 실행 모습

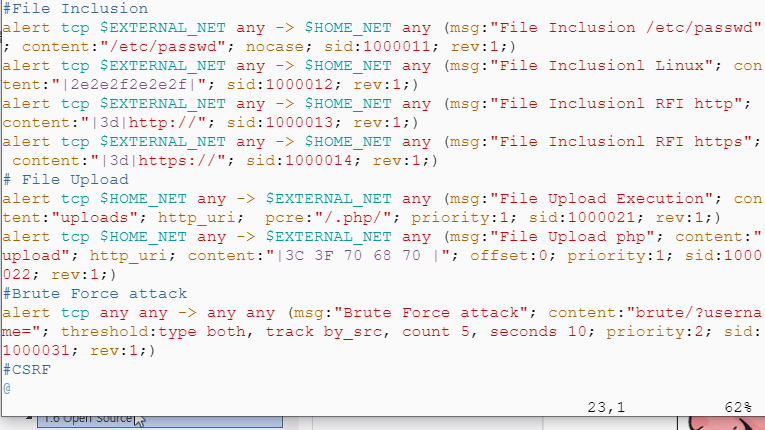
* + 1. 룰 정책 추가

**[Step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑10] File Upload 공격 로그 확인

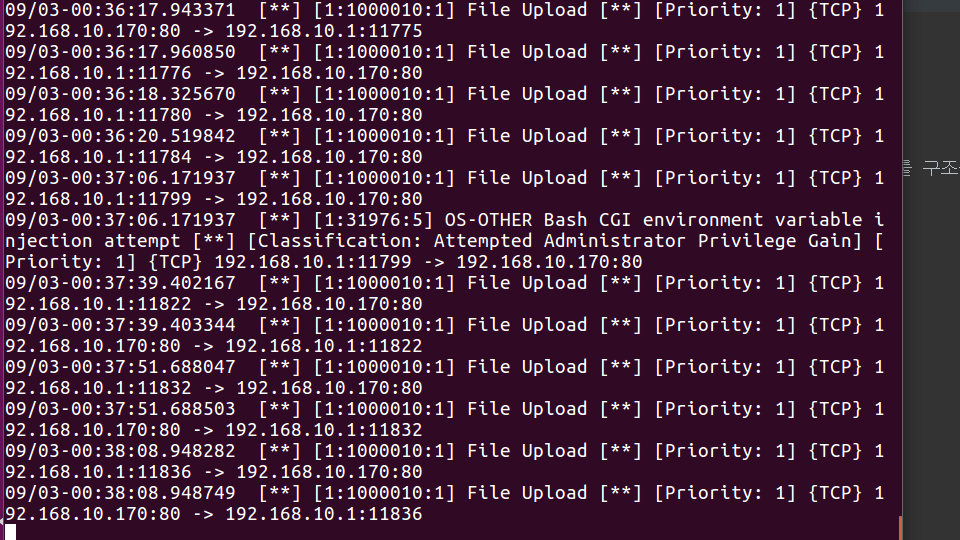
**[Step #2]** 확인한 패턴과 공격에 대한 정보를 분석 후, 알맞은 룰 정책 추가



[그림 2‑11] File Upload 룰 정책 추가

* + 1. 탐지 확인

**[step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑12] File Upload 공격 로그 확인

* 1. SQL Injection
     1. 정의

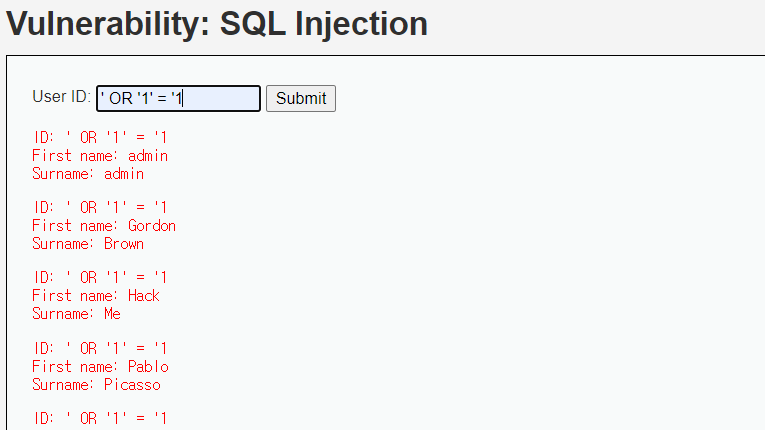
SQL Injection 이란 악의적인 사용자가 보안상의 취약점을 이용하여, 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 공격법이다.

SQL Injection의 공격 종류는 아래와 같다.

|  |
| --- |
| - Error based SQL Injection : 논리적 에러를 이용한 SQL Injection  - Union based SQL Injection : Union 명령어를 이용한 SQL Injection  - Blind SQL Injection : 참, 거짓을 확인하여 이용한 Boolean based SQL  응답에 걸리는 시간을 이용한 Time based SQL  - Stored Procedure SQL Injection : 저장된 프로시저 에서의 SQL Injection  - Mass SQL Injection : 다량의 SQL Injection 공격 |

* + 1. 공격

**[Step #1]** SQL Injection이 가능한 것을 확인



[그림 2‑13] sql Injection 실행

* + 1. 룰 정책 추가

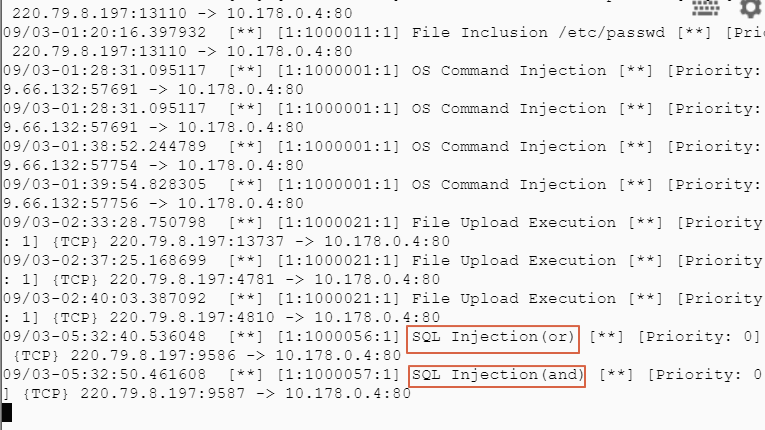
**[Step #1]** Snort에서 제공하는 SQL 룰셋을 적용한 후, 이를 보완하여 적용하였다.



[그림 2‑14] SQL 룰 추가

* + 1. 탐지 확인

**[step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑15] SQL Injection 공격 로그 확인

* 1. XSS(Cross Site Script)
     1. 정의

- XSS(Cross Site Scripting)은 스크립트를 인젝션하여 쿠키 정보 획득 또는 의도하지 않는 동작을 발생시키는 공격법이다. 사이트를 교차해서 스크립트를 발생시키며, 클라이언트측을 대상으로 한 공격이라는 특징이 있다.

- 공격의 종류에는 Stored 방식과 Reflected 방식, DOM기반 방식이 있다.   
- Stored 방식은 웹 애플리케이션 취약점이 있는 웹 서버에 악성 스크립트를 영구적으로 저장해 놓는 방법으로, 웹 사이트의 게시판, 사용자 프로필 및 코멘트 필드 등에 악성 스크립트를 삽입해 놓으면, 사용자가 사이트를 방문하여 저장되어 있는 페이지에 정보를 요청할 때, 서버는 악성 스크립트를 사용자에게 전달하여 사용자 브라우저에서 스크립트가 실행되면서 공격한다  
- Reflected 방식은 웹 애플리케이션의 지정된 변수를 이용할 때 발생하는 취약점을 이용하는 것으로, 검색 결과, 에러 메시지 등 서버가 외부에서 입력받은 값을 받아 브라우저에게 응답할 때 전송하는 과정에서 입력되는 변수의 위험한 문자를 사용자에게 그대로 돌려주면서 발생한다.  
- 마지막으로, DOM 기반 방식은 피해자의 브라우저가 HTML 페이지를 구문 분석할 때마다 공격 스크립트가 DOM 생성의 일부로 실행되면서 공격되며, 페이지 자체는 변하지 않으나, 페이지에 포함되어 있는 브라우저측 코드가 DOM 환경에서 악성코드로 실행된다

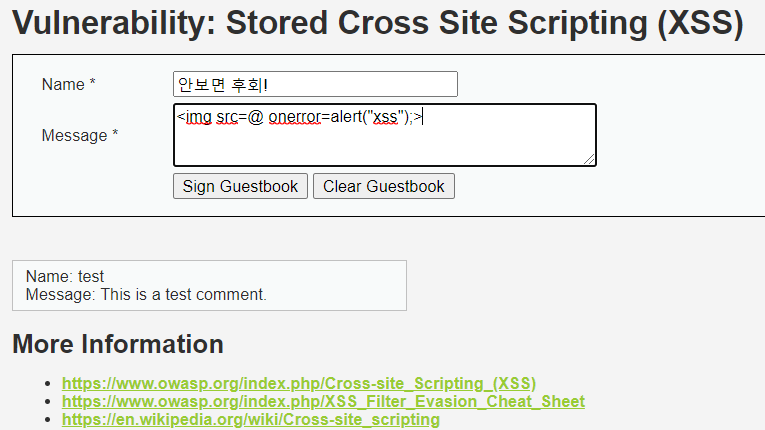
* + 1. 공격

**[Step #1]** XSS Reflected 공격이 가능한 것을 확인



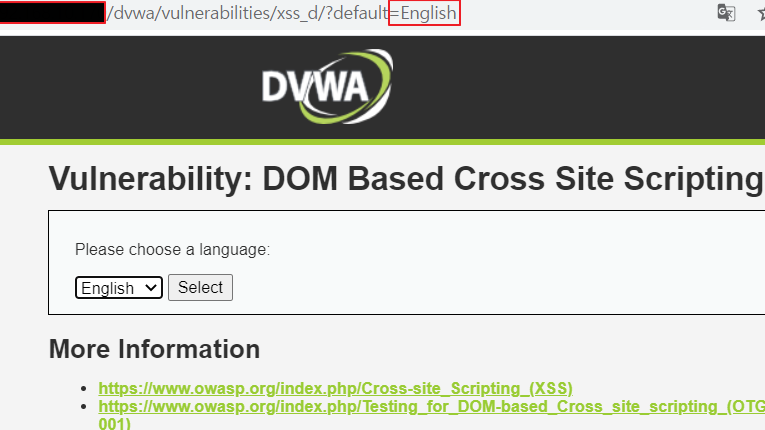
[그림 2‑16] XSS Reflected 공격

**[Step #2]** XSS Stored 공격이 가능한 것을 확인



[그림 2‑17] XSS Stored 공격

**[Step #3]** DOM(Document Object Model) 기반을 사용하는 것을 확인



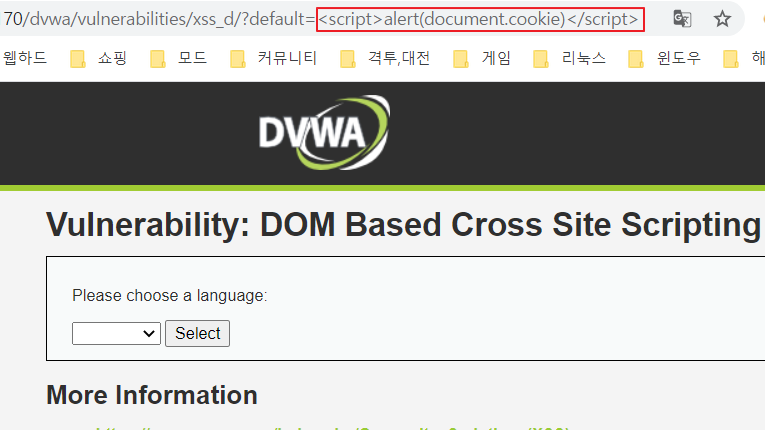
[그림 2‑18] DOM 기반 사용 확인

**[Step #4]** DOM(Document Object Model) 기반을 사용하는 것을 확인2



[그림 2‑19] DOM 기반임을 확인할 수 있는 페이지

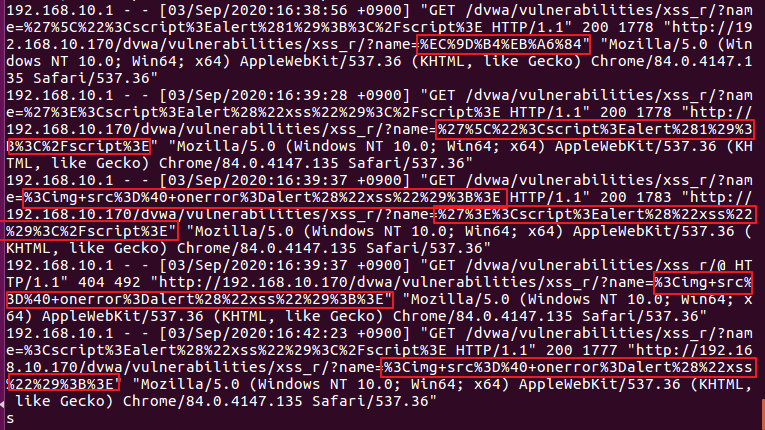
**[Step #5]** DOM(Document Object Model) 기반의 XSS 공격



[그림 2‑20] DOM 기반 XSS 공격

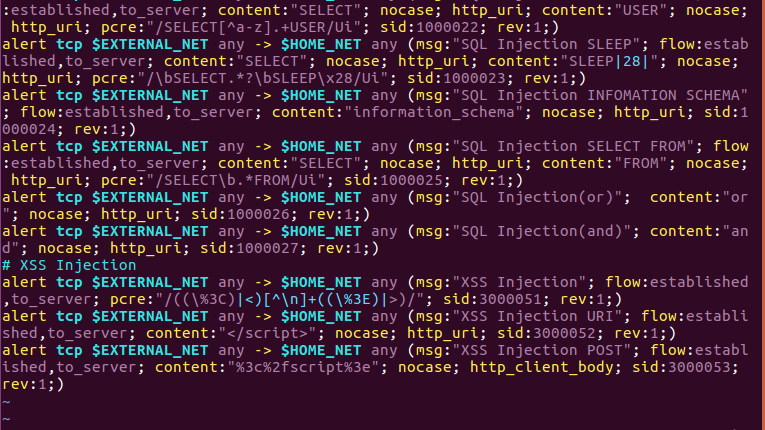
* + 1. 룰 정책 추가

**[Step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



[그림 2‑21] XSS Reflected 공격 로그 확인

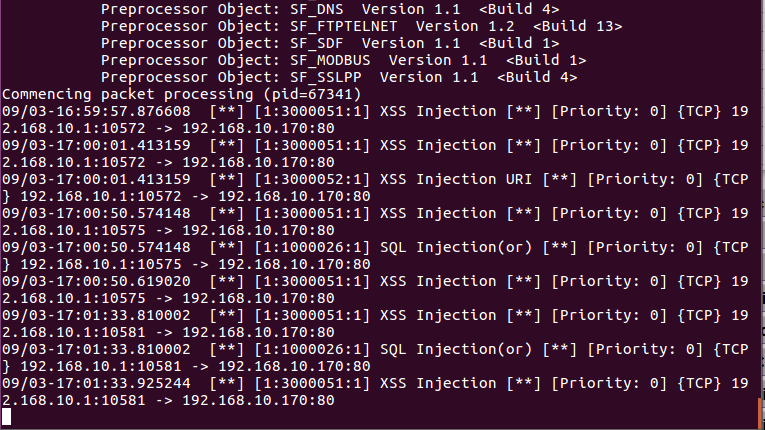
**[Step #2]** 추가적인 알맞은 XSS 룰셋을 추가함



[그림 2‑22] SQL Injection룰 세팅

* + 1. 탐지 확인

**[step #1]** 웹서버 로그를 분석해 공격의 패턴을 확인함



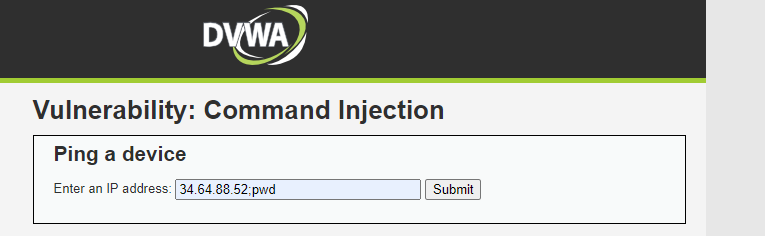
[그림 2‑23] XSS(Cross Site Script) 공격 로그 확인

* 1. OS Command Injection
     1. 정의

Command Injection은 웹 애플리케이션에서 시스템 명령을 사용할 때, 세미콜론, &, && 등을 사용하여 서버 운영체제에 접근하는 공격이다. OWASP Top10 중 1위에 속해있는 Injection 공격으로 서버자체의 콘솔 명령어를 실행시킬 수 있기 때문에 공격이 성공한다면 매우 큰 피해가 발생 할 수 있다.

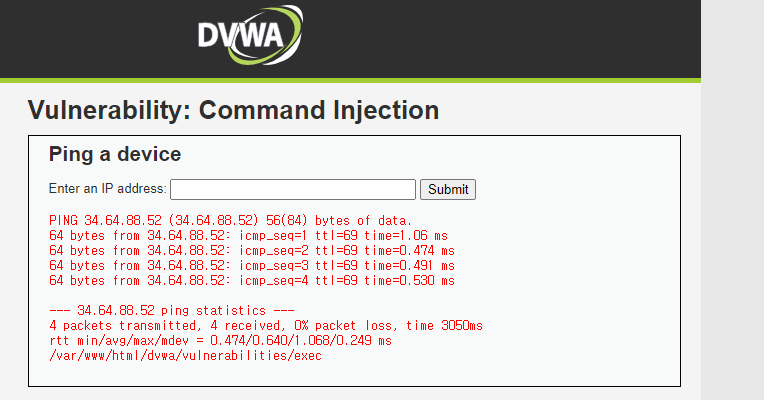
* + 1. 공격

**[step #1]** IP뒤에 ; , & , && 등을 붙여서 운영체제 명령어를 입력하여 실행시킨다.



[그림 2‑24] OS Command Injection 공격

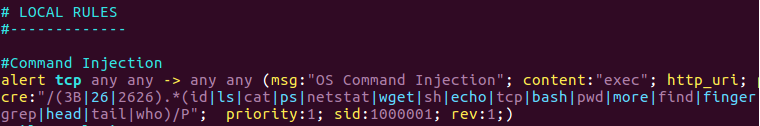
**[step #2]** 명령어를 실행한 후 ping과 pwd의 실행결과를 볼 수 있다.



[그림 2‑25] OS Command Injectino 공격 실행 결과

* + 1. 룰 정책 추가

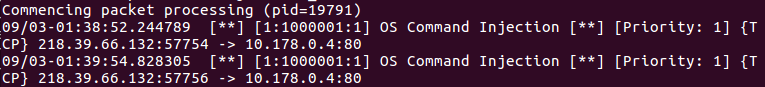
**[step #1]** OS Command Injection에 대한 공격을 분석한 후, 탐지할 수 있는 룰을 세팅한다.



[그림 2‑26] OS Command Injection 공격 탐지 룰

* + 1. 탐지 확인

**[step #1]** OS Command Injection 공격이 다시 들어왔을 때 snort를 통해 탐지 된 것을 볼 수 있다.



[그림 2‑27] OS Command Injection 공격 탐지 모습

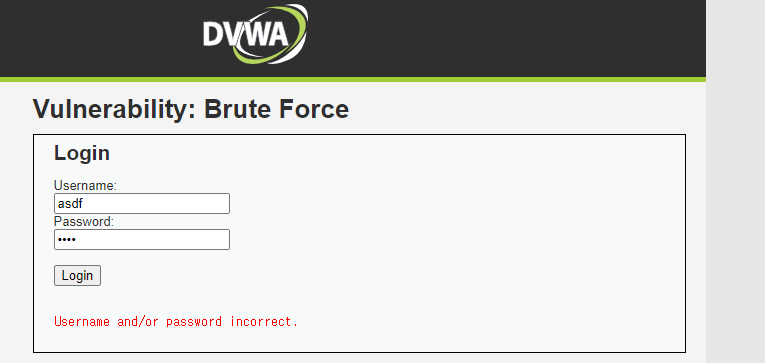
* 1. Brute Force
     1. 정의

데이터베이스에 저장된 사용자의 아이디와 비밀번호를 알기 위해 문자, 숫자 및 기호의 가능한 모든 조합을 체계적으로 시도하여 비밀번호를 찾으려고 시도하는 공격이다.

* + 1. 공격

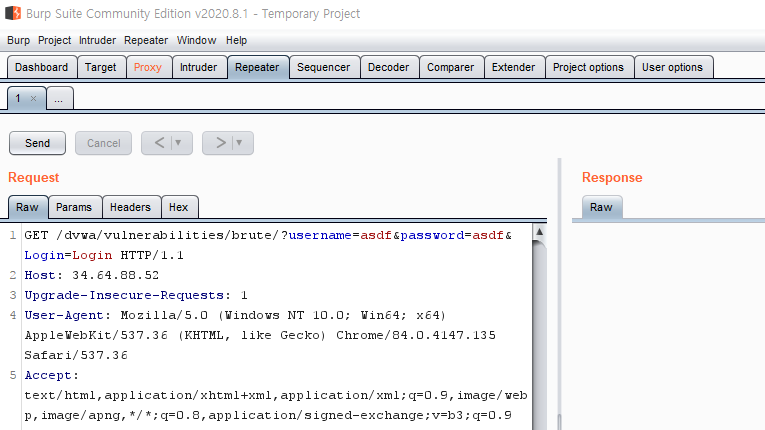
정상적인 사용자가 비밀번호를 모른다하더라도 10초 내에 아이디와 비밀번호를 5번 입력하는것은 불가능한 일이기 때문에 Brute Force공격이라는 것을 알 수 있다.

**[step #1]** DVWA 로그인 페이지에서 아이디와 비밀번호를 입력하고 Burp suite를 켜서 패킷을 잡는다.



[그림 2‑28] Brute Force를 위한 로그인

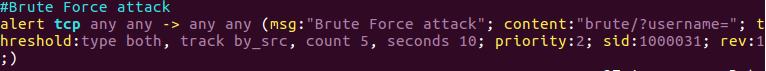
**[step #2]** 패킷을 Repeater로 보내고 username에서 아이디를 변경해가면서 10초동안 5번 이상의 send 버튼을 누른다.



[그림 2‑29] Burf Suite로 패킷을 캡쳐하여 Brute Force 공격을 실행하는 모습

* + 1. 룰 정책 추가

**[step #1]** 아래와 같이 Brute Force 공격을 탐지할 수 잇는 룰을 추가한다.



[그림 2‑30] Brute Force 공격 탐지 룰

* + 1. 탐지 확인

**[step #1]** Brute Force 공격 탐지 모습



[그림 2‑31] Brute Force 공격 탐지 모습

* 1. UDP Flooding
     1. 정의

Dos(Denial of Service) 공격의 일종으로 대량의 UDP 패킷을 이용하여 대상 호스트의 네트워크 자원을 소모시키는 공격이다. SYN Flooding 공격과는 달리 네트워크 bandwidth를 소모시키는 것이 목적이다.

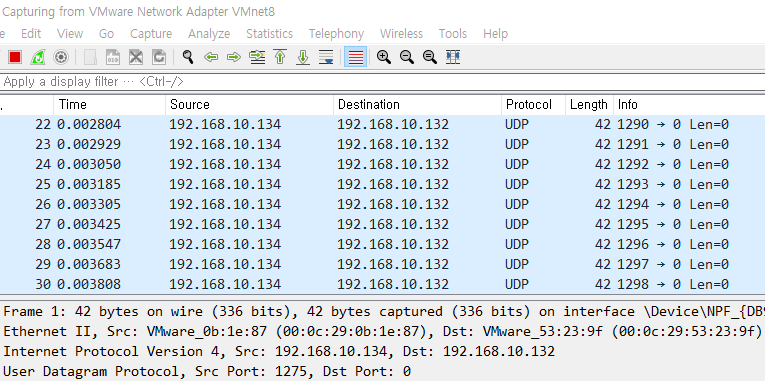
* + 1. 공격

**[step #1]** 칼리 리눅스에서 공격 대상 IP에게 UDP 패킷을 최대 속도로 보낸다.



[그림 2‑32] 칼리 리눅스에서 UDP Flooding 공격 실행 모습

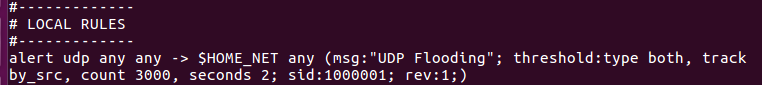
**[step #2]** 와이어샤크를 통해 확인해보면 192.168.10.134에서 192.168.10.132로 수많은 UDP 패킷이 보내진 것을 확인 할 수 있다.



[그림 2‑33] UDP 패킷 캡쳐 모습

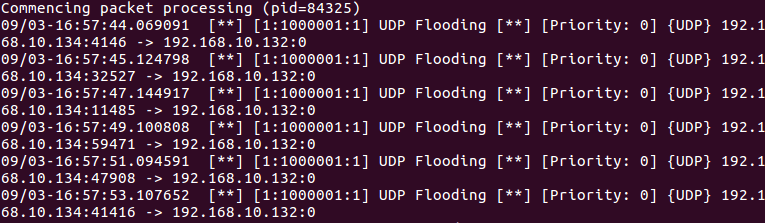
* + 1. 룰 정책 추가

**[step #1]** UDP Flooding에 대한 공격을 분석한 후, 탐지할 수 있는 룰을 세팅한다. 2초동안 3000번 카운트 되었을 때 로그를 발생시킨다.



[그림 2‑34] UDP Flooding 공격 탐지 룰

* + 1. 탐지 확인

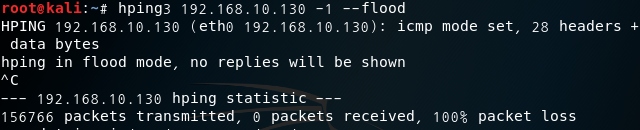
**[step #1]** UDP Flooding 공격이 다시 들어왔을 때 snort를 통해 탐지 된 것을 볼 수 있다.  


[그림 2‑35] UDP Flooding 공격 탐지 모습

* 1. ICMP Flooding
     1. 정의

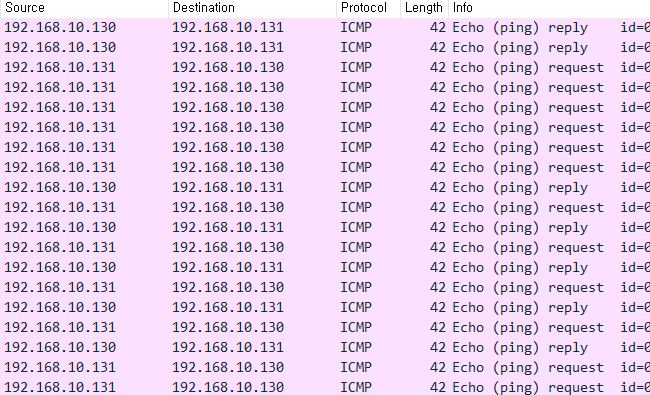
Dos(Denial of Service) 공격의 일종으로 대량의 ICMP 패킷을 이용하여 대상 호스트의 네트워크 자원을 소모시키는 공격이다.

**[step #1]** 칼리 리눅스에서 공격 대상 IP에게 ICMP 패킷을 최대 속도로 보낸다.



[그림 2‑36] 칼리 리눅스에서 ICMP Flooding 공격 실행 모습

**[step #2]** 와이어샤크를 통해 확인해보면 192.168.10.131에서 192.168.10.130로 수많은 ICMP 패킷이 보내진 것을 확인 할 수 있다.



[그림 2‑37] ICMP 패킷 캡쳐 모습

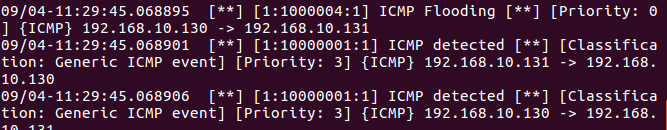
* + 1. 룰 정책 추가

**[step #1]** ICMP Flooding 공격을 분석한 후, 탐지할 수 있는 룰을 세팅한다. 2초동안 3000번 카운트 되었을 때 로그를 발생시킨다.



[그림 2‑38] ICMP Flooding 공격 탐지 룰

* + 1. 탐지 확인

**[step #1]** ICMP Flooding 공격이 다시 들어왔을 때 snort를 통해 탐지 된 것을 볼 수 있다.  


[그림 2‑39] ICMP Flooding 공격 탐지 모습

* 1. Syn flooding
     1. 정의

대량의 SYN 패킷을 타깃 서버로 보내, 서버의 대기큐를 가득 채워 정상 클라이언트의 연결 요청을 방해하는 공격이다. 조작된 출발지 IP로부터 서버는 SYN패킷을 전송받고, TCP-3way handshake를 위해 SYN-ACK플래그를 가짜 출발지로 전송한다. 가짜 주소이므로 ACK응답을 받지 못하고 연결할 때까지 기다리게 된다

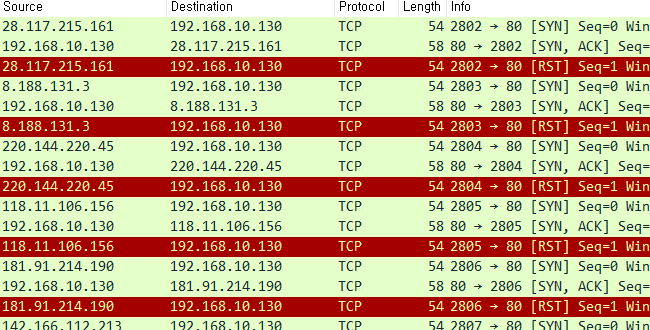
* + 1. 공격

**[step #1]** 칼리 리눅스에서 공격 대상 IP에게 SYN 패킷을 최대 속도로 보낸다. 이때, --rand-source 옵션을 사용해서 출발지 IP를 존재하지 않는 IP로 위조해서 보낸다.



[그림 2‑40] 칼리리눅스에서 SYN Flooding 공격 모습

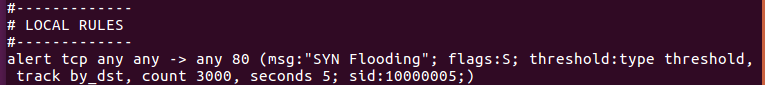
**[step #2]** 와이어샤크를 통해 확인해보면 랜덤 IP에서 192.168.10.132로 수많은 SYN 패킷이 보내진 것을 확인 할 수 있다.



[그림 2‑41] SYN Flooding 공격 패킷 캡쳐

* + 1. 룰 정책 추가

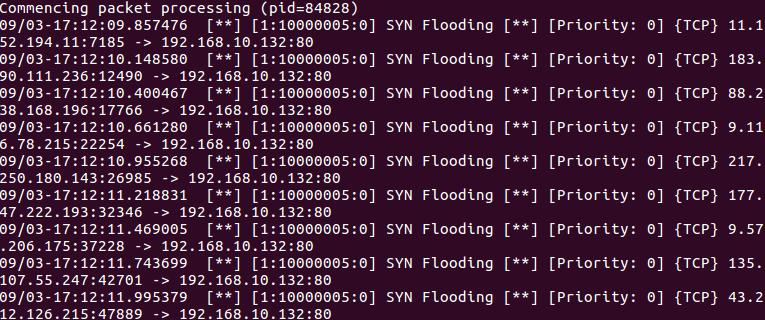
**[step #1]** SYN Flooding에 대한 공격을 분석한 후, 탐지할 수 있는 룰을 세팅한다. 5초동안 3000번 카운트 되었을 때 로그를 발생시킨다.



[그림 2‑42] SYN Flooding 공격 탐지 룰

* + 1. 탐지 확인

**[step #1]** SYN Flooding 공격이 다시 들어왔을 때 snort를 통해 탐지 된 것을 볼 수 있다.



[그림 2‑43] SYN Flooding 공격 탐지 모습

* 1. SYN-ACK Flooding
     1. 정의

TCP 3-way handshake에 처음부터 SYN-ACK 패킷을 전송하여 불필요한 RST 플래그 패킷을 유발해 CPU 자원이 과부하 되게 만드는 공격이다.

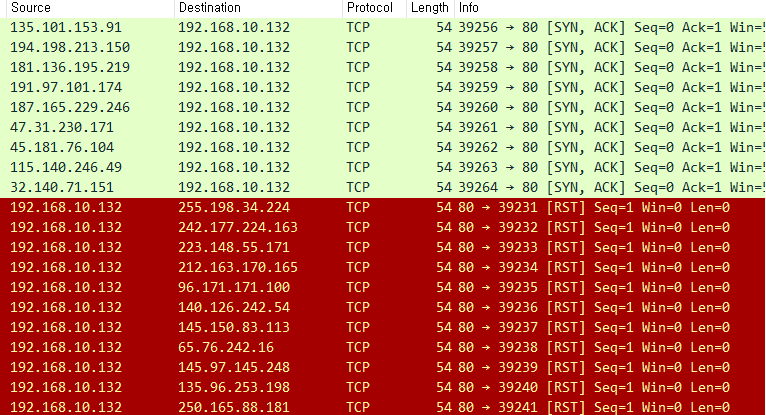
* + 1. 공격

**[step #1]** 칼리 리눅스에서 공격 대상 IP에게 SYN과 ACK 패킷을 최대 속도로 보낸다. 이때, --rand-source 옵션을 사용해서 출발지 IP를 존재하지 않는 IP로 위조해서 보낸다.



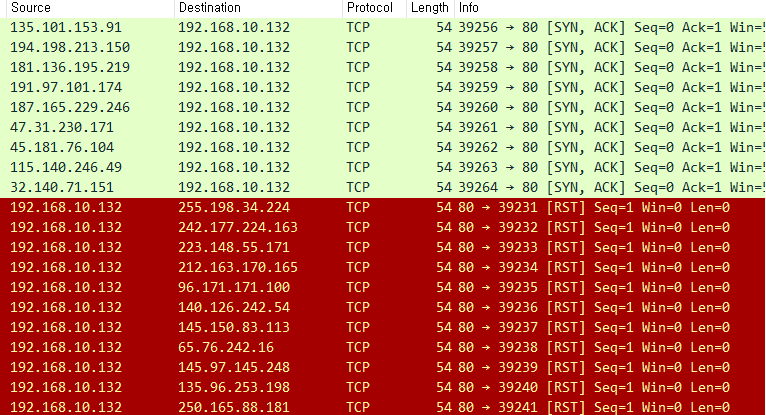
[그림 2‑44] 칼리 리눅스에서 SYN-ACK Flooding 공격

**[step #2]** 와이어샤크를 통해 확인해보면 랜덤한 IP에서 192.168.10.132로 수많은 [SYN, ACK] 패킷이 보내진 것을 확인 할 수 있다.



[그림 2‑45] SYN-ACK Flooding 공격 패킷

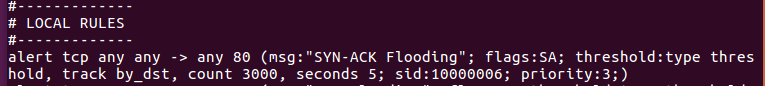
**[step #3]** 패킷에 대한 응답으로 192.168.10.132에서 수많은 RST 패킷이 보내진 것을 확인 할 수 있다.



[그림 2‑46] SYN-ACK Flooding 공격 패킷 2

* + 1. 룰 정책 추가

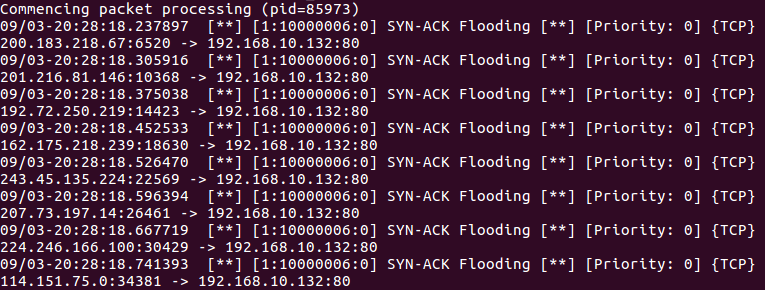
**[step #1]** SYN-ACK Flooding에 대한 공격을 분석한 후, 탐지할 수 있는 룰을 세팅한다. 5초동안 3000번 발생했을 때 로그를 발생시킨다.



[그림 2‑47] SYN-ACK Flooding 공격 탐지 룰

* + 1. 탐지 확인

**[step #1]** SYN-ACK Flooding 공격 탐지 확인



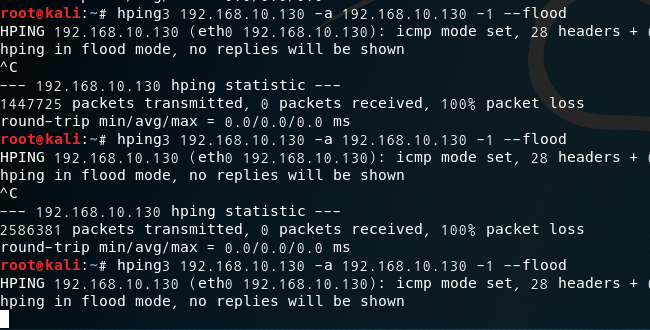
[그림 2‑48] SYN-ACK Flooding 공격 탐지 모습

* 1. Land Attack
     1. 정의

Land Attack 은 Local Area Network Denial Attack 으로 IP Spoofing을 이용한 SYN공격이다. 공격자는 공격대상 시스템의 IP주소로 출발지 주소를 변경한 TCP SYN패킷을 보낸다. 패킷을 받은 시스템은 TCP SYN-ACK를 자신에게 보내게 된다. 이를 반복하다보면 서버는 SYN패킷을 받은 후 연결을 위해 계속해서 RAM을 확보해 두게 되고, RAM이 꽉 차게 되면 서비스를 할 수 없다.

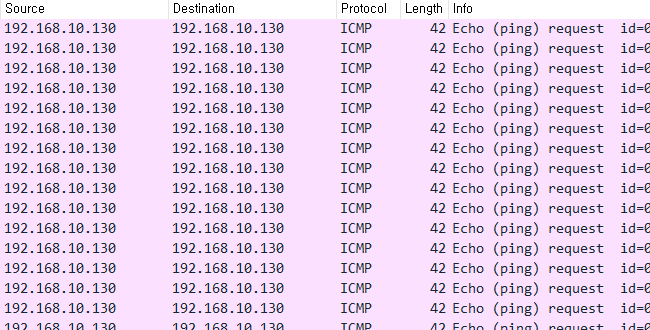
* + 1. 공격

**[Step #1]** 칼리에서 출발지 주소를 공격대상 IP 주소로 변경하여 SYN 패킷을 전송한다.



[그림 2‑49] 칼리에서 Land Attack 공격

**[step #2**] 와이어샤크를 통해 패킷캡쳐



[그림 2‑50] Land Attack 패킷 캡쳐

* + 1. 룰 정책 추가

[step #1] 공격을 탐지할 수 있는 룰을 세팅



[그림 2‑51] Land Attack 룰 추가

* + 1. 탐지 확인

[step #1] 로그를 분석해 공격의 패턴을 확인함



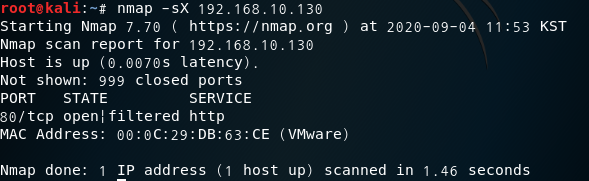
[그림 2‑52] Land Attack 탐지 모습

* 1. NMAP XMAS Scan
     1. 정의

공격 대상 시스템에 세션을 성립하지 않고 스캔하는 방식으로, 로그가 남지 않으며 열린 포트를 확인할 수 있다. TCP 플래그 값을 모두 설정하거나, 일부 값을 설정하여 패킷을 보내 스캔한다. SYN+FIN과 같은 패킷을 보낸다. 포트가 열려있으면 아무 응답이 없고, 닫혀있으면 rst+ack패킷이 돌아온다는 점을 이용한다.

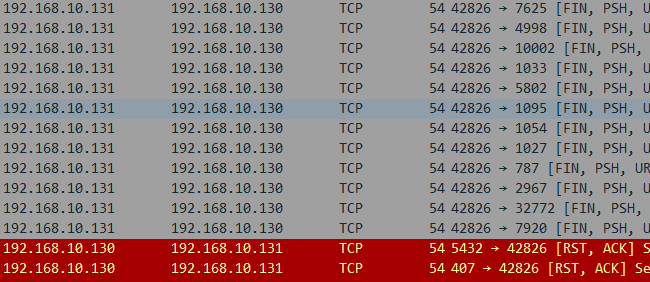
* + 1. 공격

**[step #1]** kail 에서 공격 nmap을 활용하여 공격한다.



[그림 2‑53] kail에서 NMAP XMAS 공격 모습

**[step #2]** 피해PC의 포트에 FIN, PSH, URG 플래그를 설정하여 보낸다.



[그림 2‑54] 와이어샤크에서 XMAS 패킷 캡쳐

* + 1. 룰 정책 추가

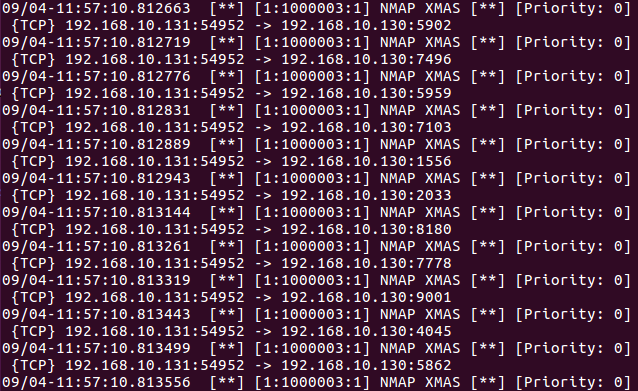
[step #1] 공격을 탐지할 수 있는 룰을 세팅



[그림 2‑55] nmap xmas 정책 추가

* + 1. 탐지 확인

[step #1] 로그를 분석해 공격의 패턴을 확인함



[그림 2‑56] NMAP XMAS 탐지

# 결론

Snort와 ELK 스택을 활용하여 웹과 네트워크 취약점에 대응하는 방어 솔루션을 구축하였다. GCP 클라우드를 이용하여 하나의 서버에서 명이 동시 공격을 할 수 있게 만들었으며, 룰 셋의 공유를 가능하게 만들었다. 다만 네트워크 해킹은 구글의 네트워크 보안 정책으로 인하여, 공격 실습이 어려웠다. 따라서 네트워크 공격 및 방어룰은 자신의 로컬에서 시행했다.

총 12개의 공격 탐지 규칙을 설정하고, 직접 탐지 실습을 진행했다. 각각 로그를 ELK 스택을 통해 시각화해보았으며, 와이어샤크를 활용해 패킷을 캡쳐해 분석했다. 이를 통해 웹과 네트워크의 해킹 과정을 상세203하게 확인하고, 방어 규칙을 더욱 상세하게 구성할 수 있었다. 특히, ELK 스택에서 실시간 로그 정보를 시각화하여 볼 수 있도록 한 것에 큰 의의가 있다.

# 추후과제 및 후기

GCP 클라우드를 이용하여 웹서버와 IDS를 구성하여 ELK로 화면을 구성하는 것을 목표로 하였다. 하지만 GCP에서는 snort의 룰셋이 잘 적용이 안는 경향을 보였으며, 그로 인하여 룰셋을 적용하고 테스트를 하는데 오랜 걸린 시간이 걸리게 되었다. 또한 여러 사용자가 동시에 룰셋 편집을 하는데 문제가 있다는 점을 알게되었다. 추후에는 GCP에서 룰셋이 잘 안되는 원인을 파악하고 더 상세하게 룰셋을 설정하기로 한다.

ELK 스택 구성에 원인 모를 에러들이 발생하여 구성에 오랜 시간이 걸리게 되었다. 추후에는 ELK 스택 구성 및 설정을 문서화 및 최적화하여 ELK 스택 구성에 많은 시간을 소모하지 않게 할 예정이다.

GCP 클라우드와 ELK 스택 구성에 많은 시간을 할애하여 원래 목적인 snort 룰 설정에 미흡한 점이 보인다. 다만 GCP 클라우드와 ELK 스택을 구성하면서 IDS 구축에 대한 이해와 지식이 상승하였으며, 직접 룰 세팅을 구성함으로써 룰 세팅에 대한 지식과 공격과 보안에 대한 이해가 더욱 높아지는 기회가 되었다.