CHAPTER 14

웹 방화벽 (WAF)

이현호 hyeonholee@korea.com

웹 응용프로그램 방화벽(WAF)

WAF(Web Application Firewall)

웹 응용프로그램에 대한 HTTP/HTTPS 트래픽을 모니터링하여 악성 요청을 차단하는 방화벽

작동 원리

- **트래픽 모니터링**
  - WAF는 웹 애플리케이션으로 들어오는 모든 HTTP/HTTPS 요청을 실시간으로 모니터링
- **패턴 매칭**
  - WAF는 OWASP Top 10 취약점과 같은 공통 보안 취약점에 대응하기 위해 사전에 정의된 보안 규칙 세트를 사용하여 악성 패턴 식별
- **행동 분석**
  - WAF는 정상적인 사용자 행동과 비정상적인 행동을 구분하기 위해 행동 분석 기법을 사용
- **요청 차단**
  - 악성 요청이 감지되면 WAF는 해당 요청을 차단하거나 제한

웹 응용프로그램 방화벽(WAF)

주요 기능

- **필터링**
  - **악성 요청 차단 :** SQL Injection, XSS 등의 공격을 차단한다.
  - **허용된 트래픽만 통과시키기 :** 허용된 트래픽만 웹 서버로 전달하여 안전한 트래픽을 보장한다.
- **로깅**
  - **모든 요청 및 응답 기록 :** 모든 HTTP/HTTPS 요청과 응답을 기록하여 보안 사고 발생 시 추적 가능성을 높인다.
  - **분석 및 추적 가능성 제공 :** 로그 데이터를 분석하여 공격 패턴을 파악하고, 보안 정책을 강화할 수 있다.
- **경고**
  - **의심스러운 활동 감지 시 관리자에게 경고 알림 :** 실시간으로 의심스러운 활동을 감지하고, 관리자에게 경고 알림을 보낸다.

웹 응용프로그램 방화벽(WAF)

WAF가 제공하는 보호 수준

- **실시간 공격 차단**
  - WAF는 실시간으로 HTTP/HTTPS 트래픽을 분석하여 악성 요청을 차단하여 SQL Injection, XSS, CSRF 등의 공격 방지 가능
- **알려진 취약점 방어**
  - WAF는 알려진 취약점에 대한 공격을 차단하여 OWASP Top 10 취약점 등에 대한 대응 규칙 제공

웹 응용프로그램 방화벽(WAF)

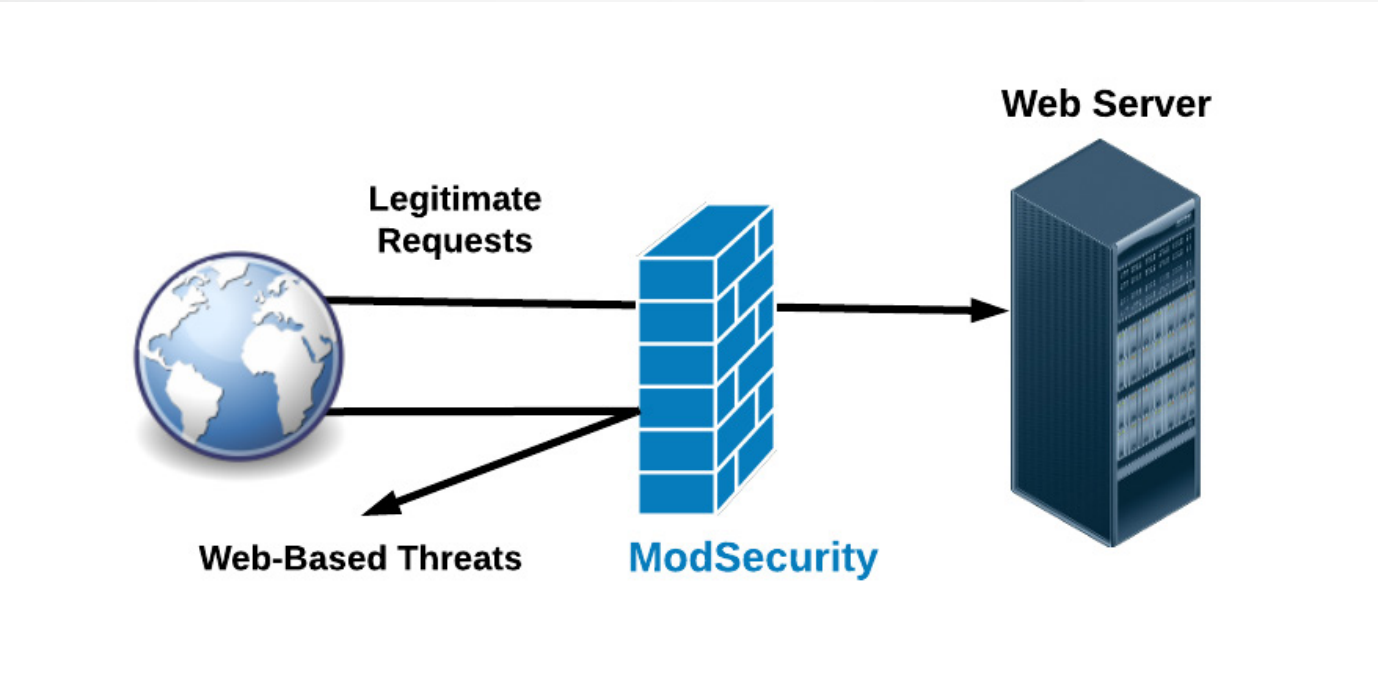
WAF의 한계

- **새로운 공격 패턴에 대한 대응 한계**
  - WAF는 사전에 정의된 규칙을 기반으로 작동하여 새로운 공격 패턴에 대한 대응이 제한되므로, 지속적인 업데이트와 규칙 추가 필요
- **설정 오류로 인한 오탐 및 과탐 가능성**
  - WAF의 설정이 잘못되면 정상적인 요청이 차단되거나 악성 요청이 탐지되지 않을 수 있으므로, 설정과 튜닝이 중요
- **다른 보안 솔루션과의 통합 필요성**
  - WAF는 단독으로 모든 보안 문제를 해결할 수 없으므로 방화벽, IDS/IPS 등 다른 정보보호시스템과의 통합 필요

ModSecurity

웹 서비스의 공격을 효과적으로 차단할 수 있는 공개 웹 응용프로그램 방화벽(WAF) 모듈

- ModSecurity 는 웹 공격에 대한 침입탐지 및 침입방지 기능을 추가해주는 아파치 웹 서버 모듈로 동작
- 클라이언트와 아파치 웹 서버 사이에 ModSecurity가 존재하는 형태
- 클라이언트로부터 악의적인 접속요청이 발견되면 차단, 로깅 등 사전에 정의된 행위 수행



ModSecurity

주요 특징

- **오픈 소스**
  - ModSecurity는 무료로 사용할 수 있는 오픈 소스 소프트웨어
- **확장 가능성**
  - 다양한 규칙 세트를 추가하여 보안 정책을 확장 가능
- **실시간 트래픽 분석**
  - HTTP 트래픽을 실시간으로 분석하여 악성 트래픽 탐지 가능
- **광범위한 공격 방어**
  - OWASP Top 10을 포함한 다양한 웹 공격을 방어 가능

ModSecurity 설정파일

파일 구조

- **modsecurity.conf :** 메인 설정파일
- **rules.d/ :** 규칙 파일을 저장하는 디렉토리
- **crs-setup.conf :** OWASP Core Rule Set(CRS) 설정 파일

주요 설정항목

- **SecRuleEngine :** WAF 활성화 여부 (On, Off, DetectionOnly)
- **SecRequestBodyAccess :** 요청 본문 분석 활성화 (On, Off)
- **SecResponseBodyAccess :** 응답 본문 분석 활성화 (On, Off)
- **SecAuditLog :** 감사 로그 파일 경로
- **SecDefaultAction :** 기본 규칙 행동 설정

ModSecurity 적용

기본 설정 예시

```
SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess On
SecAuditLog /var/log/modsec_audit.log
SecDefaultAction "phase:2,deny,log,status:403"
```

SQL 인젝션 차단 규칙 예시

```
SecRule ARGS "union select" "id:'100002',phase:2,deny,log,status:403,msg:'SQL Injection
Detected'"
```

XSS 차단 규칙 예시

```
SecRule ARGS "<script>" "id:'100003',phase:2,deny,log,status:403,msg:'XSS Detected'"
```

➡ 이후, 웹 응용프로그램 서버(WAS)를 재기동하여 설정 및 정책(규칙) 적용 (sudo systemctl restart httpd 등)

HTTPS 환경에 대한 적용

Apache

```
<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /path/to/cert.pem
    SSLCertificateKeyFile /path/to/key.pem
    Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
```

Nginx

```
server {
    listen 443 ssl;
    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
}
```

➡ 이후, 방화벽 설정을 통해 HTTP 및 HTTPS 통신 허용 필요

보안 정책 및 규칙의 이해

보안 정책 및 규칙

시스템이나 조직에서 보안을 유지하기 위해 설정한 규칙과 절차로, ModSecurity에서는 이러한 보안 정책을 규칙 파일로 정의하여 웹 응용프로그램을 보호

보안 정책 및 규칙의 역할과 중요성

예방 – 탐지 – 대응

- **역할**
  - **예방 :** 잠재적인 보안 위협을 사전에 차단합니다.
  - **탐지 :** 시스템 내에서 발생하는 보안 위협을 감지합니다.
  - **대응 :** 보안 사고 발생 시 신속하게 대응합니다.
- **중요성**
  - 보안 정책은 잠재적인 위협으로부터 시스템을 보호하는데 필수적이며, 일관된 보안 관행을 보장
  - ModSecurity의 기본 보안 정책과 규칙을 통해 웹 응용프로그램에 대한 다양한 공격을 효과적으로 차단

보안 정책 및 규칙의 이해

보안 정책 및 규칙의 구성요소

- **접근 제어**
  - 사용자와 시스템 자원 간의 상호작용을 제어
    - **예 :** 특정 IP 주소나 사용자 에이전트의 접근을 차단
- **입력 검증**
  - 시스템에 들어오는 모든 입력을 검증
    - **예 :** SQL Injection, XSS 등의 공격을 차단
- **오류 처리**
  - 오류가 발생할 때의 처리 방법을 정의
    - **예 :** 사용자에게 노출되지 않는 안전한 오류 메시지를 제공

기본 보안 정책

설정파일 옵션

SecRuleEngine

- ModSecurity 엔진을 활성화 또는 비활성화
- **값**
  On : ModSecurity 엔진을 활성화하여 규칙을 적용합니다.
  Off : ModSecurity 엔진을 비활성화하여 규칙을 적용하지 않습니다.
  DetectionOnly : ModSecurity 엔진을 활성화하되, 규칙 위반 시 차단하지 않고 로그만 기록합니다.

```
SecRuleEngine On
```

SecRequestBodyAccess

- 요청 본문(Request Body)을 분석할지 여부를 설정
- **값**
  On : 요청 본문을 분석합니다.
  Off : 요청 본문을 분석하지 않습니다.

```
SecRequestBodyAccess On
```

기본 보안 정책

설정파일 옵션

SecResponseBodyAccess

- 응답 본문(Response Body)을 분석할지 여부를 설정
- **값**
  On : 응답 본문을 분석합니다.
  Off : 응답 본문을 분석하지 않습니다.

```
SecResponseBodyAccess On
```

SecAuditEngine

- 감사 로그(Audit Log)를 기록할지 여부를 설정
- **값**
  On : 모든 트랜잭션을 감사 로그에 기록합니다.
  Off : 감사 로그를 기록하지 않습니다.
  RelevantOnly : 규칙을 위반하거나 관심 있는 트랜잭션만 감사 로그에 기록합니다.

```
SecAuditEngine RelevantOnly
```

기본 보안 정책

설정파일 옵션

SecAuditLog

- 감사 로그 파일의 경로를 지정
- **값**
  파일 경로 : 감사 로그가 저장될 파일의 경로를 지정합니다.

```
SecAuditLog /var/log/httpd/modsec_audit.log
```

SecAuditLogParts

- 감사 로그에 기록할 트랜잭션의 부분을 정의
- **값**

A : 요청 라인	**B :** 요청 헤더	**C :** 요청 본문	**D :** 응답 헤더
E : 응답 본문	**F :** 중간 응답 본문	**G :** 최종 응답 본문	**H :** 요청 트레일러
I : 응답 트레일러	**J :** 응답 상태 코드	**K :** 메시지 시작 시간	**Z :** 메시지 끝

```
SecAuditLogParts ABIJDEFHZ
```

기본 보안 정책

설정파일 옵션

SecDebugLog

- 디버그 로그 파일의 경로를 지정
- **값**
  파일 경로 : 디버그 로그가 저장될 파일의 경로를 지정합니다.

```
SecDebugLog /var/log/httpd/modsec_debug.log
```

SecDebugLogLevel

- 디버그 로그의 상세 수준을 설정
- **값**
  0 – 9 : 숫자가 높을수록 상세한 로그를 기록합니다. 일반적으로 0은 디버깅을 비활성화하고, 9는 매우 상세한 로그를 기록

```
SecDebugLogLevel 3
```

기본 보안 정책

설정파일 옵션

SecDefaultAction

- 기본 규칙 동작을 정의
- **값**
  규칙 동작 : 규칙이 트리거될 때 수행할 기본 동작을 정의합니다.

```
SecDefaultAction "phase:2,deny,log,status:403"
```

SecAction

- 조건 없이 항상 실행되는 규칙을 정의
- **값**
  동작 : 항상 수행될 동작을 지정합니다.

```
SecAction "id:'100001',phase:1,log,auditlog,msg:'ModSecurity enabled'"
```

기본 보안 정책

설정파일 옵션

SecRule

- 특정 조건이 충족될 때 수행할 동작을 정의하는 규칙을 작성
- **값**
  변수, 연산자, 동작 : 조건을 정의하는 변수와 연산자, 그리고 조건이 충족될 때 수행할 동작을 지정합니다.

```
SecRule ARGS "<script>" "id:'100003',phase:2,deny,log,status:403,msg:'XSS Detected'"
```

  * **예 :** SQL 인젝션 차단 규칙

```
SecRule ARGS "union select" "id:'100002',phase:2,deny,log,status:403,msg:'SQL
Injection Detected'"
```

  * **예 :** XSS 차단 규칙

```
SecRule ARGS "<script>" "id:'100003',phase:2,deny,log,status:403,msg:'XSS Detected'"
```

기본 보안 규칙

기본 규칙 작성 원칙

- **SecRule**
  - ModSecurity 규칙의 기본 구성 요소
    - **예 :** SecRule ARGS "<script>" "id:'100003',phase:2,deny,log,status:403,msg:'XSS Detected'"
- **변수**
  - 변수에 따라 검사 대상과 범위 설정
  - 변수명 뒤에 콜론(:)을 붙여 하위 변수(요소) 지정 가능
    - **예 :** ARGS:id (매개변수 id에 대해 검사), REQUEST_HEADERS:User-Agent (User-Agent 헤더에 대해 검사) 등
- **연산자**
  - 변수를 기준으로 문자열이나 값에 대한 조건을 평가

기본 보안 규칙

SecRule의 동작

- **Phase**
  - 요청 처리 단계
    - **예 :** [1] 요청 헤더 분석, [2] 요청 본문 분석, [3] 응답 헤더 분석, [4] 응답 본문 분석
- **Transformation**
  - 입력 값을 변환하여 분석
  - t:변환함수
    - **예 :** lowercase (소문자로 변환), uppercase (대문자로 변환), trim (앞뒤 공백 제거), urlDecode, urlEncode, htmlEntityDecode, jsDecode (이스케이프 시퀀스 디코딩), normalizePath (경로 정규화), compressWhitespace (연속된 공백 압축), removeWhitespace (공백 제거), base64Decode, replaceNulls (Null 바이트 제거) 등
- **Actions**
  - 규칙이 트리거될 때 수행할 작업
    - **예 :** deny, log, status:403

SecRule 변수

요청 변수 (Request Variables)

- **REQUEST_URI**
  - 클라이언트가 요청한 URI 경로
    - **예 :** /index.php
- **REQUEST_METHOD**
  - HTTP 요청 메소드
    - **예 :** GET, POST, PUT, DELETE 등
- **REQUEST_FILENAME**
  - 웹 서버가 요청한 파일 시스템 경로
    - **예 :** /var/www/html/index.php

SecRule 변수

요청 변수 (Request Variables)

- **REQUEST_HEADERS**
  - 요청 헤더의 전체 집합
    - **예 :** User-Agent, Host, Accept 등
- **ARGS**
  - 요청 매개변수의 전체 집합
    - **예 :** ?id=123&name=test
- **ARGS_GET**
  - GET 요청 매개변수
    - **예 :** id=123

SecRule 변수

요청 변수 (Request Variables)

- **ARGS_POST**
  - POST 요청 매개변수
    - **예 :** 폼 데이터 (id=123&name=test)
- **REQUEST_BODY**
  - POST 요청의 본문 데이터
    - **예 :** 폼 데이터, JSON, XML 등
- **REQUEST_COOKIES**
  - 요청된 쿠키의 전체 집합
    - **예 :** sessionid=abcd1234

SecRule 변수

응답 변수 (Response Variables)

- **RESPONSE_STATUS**
  - 서버에서 클라이언트로 전송된 HTTP 응답 상태 코드
    - **예 :** 200, 404, 500 등
- **RESPONSE_HEADERS**
  - 응답 헤더의 전체 집합
    - **예 :** Content-Type, Set-Cookie 등
- **RESPONSE_BODY**
  - 서버에서 클라이언트로 전송된 응답 본문
    - **예 :** HTML, JSON, XML 데이터 등

SecRule 변수

환경 변수 (Environment Variables)

- **REMOTE_ADDR**
  - 클라이언트의 IP 주소
    - **예 :** 192.168.1.1 등
- **REMOTE_PORT**
  - 클라이언트의 포트 번호
    - **예 :** 54321 등
- **SERVER_ADDR**
  - 서버의 IP 주소
    - **예 :** 192.168.1.100 등
- **SERVER_PORT**
  - 서버의 포트 번호
    - **예 :** 80, 443 등

SecRule 변수

기타 변수 (Other Variables)

- **FILES**
  - 업로드 된 파일 데이터로, 멀티 파트 양식 데이터의 파일 본문
- **FILE_NAMES**
  - 업로드 된 파일명
    - **예 :** file1.txt, image.png 등
- **FILES_SIZES**
  - 업로드 된 파일 크기(bytes)
    - **예 :** 1024 등
- **FILES_TMP_CONTENT**
  - 업로드 된 파일의 임시 내용으로, 파일 본문의 일부

OWASP CRS

OWASP CRS

ModSecurity에서 사용할 수 있는 기본 보안 규칙 세트로, 다양한 웹 응용프로그램 공격에 대해 사전에 정의된 규칙 제공

OWASP CRS 다운로드

```
sudo yum install mod_security_crs
```

기본 설정 파일 편집 (/etc/httpd/conf.d/mod_security.conf)

```
IncludeOptional /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf
IncludeOptional /etc/httpd/modsecurity.d/activated_rules/*.conf
```

* IncludeOptional 지시어를 사용하여 CRS 설정 파일과 규칙 파일을 포함 (OWASP CRS 설치 시 자동으로 포함되므로 개별 설정 불요)

Apache 재시작

```
sudo systemctl restart httpd
```

고급 보안 정책

고급 보안 정책의 역할

- **고도화된 공격 탐지 및 차단**
  - 최신 보안 위협에 대응하기 위해 지속적인 정책 업데이트가 필요
  - 고급 정책은 일반적인 공격 외에도 고도화된 공격을 탐지하고 차단하는데 목표
- **정교한 보안 제어**
  - 고급 정책은 특정 트래픽 패턴을 세밀하게 분석하여 오탐과 과탐을 줄이는 효과
    - **예 :** 정상적인 사용 패턴과 공격 패턴을 구분하는 규칙을 작성합니다.

고급 보안 정책

고급 보안 정책의 구성요소

- **사용자 정의 규칙**
  - 특정 요구에 맞게 사용자 정의 규칙을 작성
    - **예 :** 특정 IP 주소 차단, 특정 URL 패턴 필터링
- **로그 분석**
  - 로그 데이터를 분석하여 공격 패턴을 파악하고 규칙을 업데이트
    - **예 :** 로그에서 반복적인 패턴을 식별하여 새로운 규칙 작성
- **실시간 모니터링 및 알림**
  - 실시간 모니터링을 설정하고 다른 도구와의 연동을 통해 알림을 설정하여 새로운 위협에 빠르게 대응
    - **예 :** 실시간 알림 설정, 자동 대응 시스템 구축
  - 단, ModSecurity 자체에는 알림 기능이 없으므로 ELK Stack 등 다른 도구와의 연동 필요

정규표현식

정규표현식(Regular Expression, Regex/Regexp)

어떤 문자열 내에서 '특정한 형태나 규칙을 가진 문자열'을 찾기 위해 그 형태나 규칙을 나타내는 패턴을 정의하는 식

정규표현식의 등장 배경

Windows 등 GUI 운영체제가 대중화되기 전에는 방대한 양의 문자열을 편집하는 과정에서 문자열을 찾거나 선택하거나 교체하기 위해 명령어(식)를 입력

- **POSIX 정규식 :** POSIX 표준에 편입된 최초의 명령어(식) 모음
- **vim 정규식 :** POSIX 정규식 중 표준식(Basic Regular Expression, BRE)을 기본 골격으로 하여 vi/vim 편집기에 사용된 정규표현식
- **PCRE(Perl Compatible Regular Expression) :** Perl(프로그래밍 언어) 스크립트 언어에서 발전된 정규표현식

정규표현식의 규칙 적용

정교한 패턴 매칭을 위해 정규 표현식을 사용

정규표현식의 필요성

```
In this example, we will explore various patterns within a
text. The text includes several repeated phrases like
"data data data analysis" and "data data data data
analysis", numbers in sequences such as 123, 456, and 789,
and patterns like 101-202-3030 and 404-505-6060. You can
find phone numbers formatted as (123) 456-7890, emails
like example@example.com, and dates in the format of 2023-
02-20. The purpose of "pattern recognition" in pattern
recognition is to illustrate how regex can be effectively
used for data extraction and text analysis. For more
information, visit our website at https://www.regex-
example.com.
```

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "**data data data** analysis" and "**data data data data** analysis", numbers in sequences such as 123, 456, and 789, and patterns like 101-202-3030 and 404-505-6060. You can find phone numbers formatted as (123) 456-7890, emails like example@example.com, and dates in the format of 2023-02-20. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for **data** extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as **123**, **456**, and **789**, and patterns like **101-202-3030** and **404-505-6060**. You can find phone numbers formatted as (**123**) **456-7890**, emails like example@example.com, and dates in the format of **2023-02-20**. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as 123, 456, and 789, and patterns like 101-202-3030 and 404-505-6060. You can find phone numbers formatted as **(123) 456-7890**, emails like example@example.com, and dates in the format of 2023-02-20. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as 123, 456, and 789, and patterns like **101-202-3030** and **404-505-6060**. You can find phone numbers formatted as **(123) 456-7890**, emails like example@example.com, and dates in the format of 2023-02-20. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as 123, 456, and 789, and patterns like 101-202-3030 and 404-505-6060. You can find phone numbers formatted as (123) 456-7890, emails like **example@example.com**, and dates in the format of 2023-02-20. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as 123, 456, and 789, and patterns like 101-202-3030 and 404-505-6060. You can find phone numbers formatted as (123) 456-7890, emails like example@example.com, and dates in the format of **2023-02-20**. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at https://www.regex-example.com.

정규표현식의 필요성

In this example, we will explore various patterns within a text. The text includes several repeated phrases like "data data data analysis" and "data data data data analysis", numbers in sequences such as 123, 456, and 789, and patterns like 101-202-3030 and 404-505-6060. You can find phone numbers formatted as (123) 456-7890, emails like example@example.com, and dates in the format of 2023-02-20. The purpose of "pattern recognition" in pattern recognition is to illustrate how regex can be effectively used for data extraction and text analysis. For more information, visit our website at **https://www.regex-example.com**.

정규표현식 사용 규칙

SQL 인젝션 패턴

```
SecRule ARGS "select.+from" "id:'200002',phase:2,t:lowercase,deny,log,status:403,msg:
'SQL Injection Detected'"
```

XSS 패턴

```
SecRule ARGS ".*<script>.*" "id:'200003',phase:2,deny,log,status:403,msg:'XSS Detected'"
```

동적 스크립트 생성 패턴

```
SecRule ARGS "document.write\(['\"]<scr['\"]\);document.write\(['\"]ipt['\"]\);" "id:
'200006',phase:2,deny,log,status:403,msg:'Dynamic Script Injection Detected'"
```

Base64 부호화 페이로드 패턴

```
SecRule ARGS "@rx (?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?" "id:
'200007',phase:2,deny,log,status:403,msg:'Base64 Encoded Payload Detected'"
```

SecRule 변수

고급 변수 (Advanced Variables)

- **MATCHED_VARS**
  - 현재 규칙에 의해 일치된 변수들의 집합으로, 여러 변수에서 일치 항목이 있을 때 사용
- **MATCHED_VAR**
  - 현재 규칙에 의해 일치된 첫 번째 변수
- **MATCHED_VARS_NAMES**
  - 현재 규칙에 의해 일치된 변수들의 이름 목록
- **TX**
  - 트랜잭션 범위 내에서 사용자 정의 변수로, 규칙 간 데이터 공유 시 사용
    - **예 :** TX:username, TX:attack_score, TX:block_flag 등

SecRule 변수

상태 변수 (Status Variables)

- **INBOUND_DATA_ERROR**
  - 입력 데이터의 오류 여부
- **OUTBOUND_DATA_ERROR**
  - 출력 데이터의 오류 여부
- **MULTIPART_STRICT_ERROR**
  - 멀티파트 양식의 엄격한 구문 분석 오류 여부
- **RESPONSE_BODY_PROCESSOR_ERROR**
  - 응답 본문 처리 오류 여부

SecRule 연산자

기본 연산자

- **@rx**

```
ARGS "@rx ^[a-zA-Z0-9]+$" "msg:'Invalid characters'"
```

  - 정규표현식을 사용하여 문자열이 패턴과 일치하는지 여부 검사

- **@beginsWith**

```
REQUEST_URI "@beginsWith /admin" "msg:'Admin access detected'"
```

  - 문자열이 특정 문자열로 시작하는지 확인

- **@endsWith**

```
REQUEST_URI "@endsWith .php" "msg:'PHP file requested'"
```

  - 문자열이 특정 문자열로 끝나는지 확인

- **@contains**

```
REQUEST_BODY "@contains password" "msg:'Sensitive data detected'"
```

  - 특정 문자열을 포함하는지 확인

- **@pm**

```
REQUEST_BODY "@pm select insert update delete" "msg:'SQL keywords'"
```

  - 공백이나 쉼표로 구분된 문자열 목록에서 일치 항목 검사

SecRule 연산자

논리 연산자

- **@eq**

```
RESPONSE_STATUS "@eq 404" "msg:'Page not found'"
```

  - 변수 값이 특정 값과 같은 지 비교

- **@gt**

```
RESPONSE_BODY_SIZE "@gt 1048576" "msg:'Large response body'"
```

  - 변수 값이 특정 값보다 큰 지 비교

- **@lt**

```
RESPONSE_TIME "@lt 1000" "msg:'Fast response time'"
```

  - 변수 값이 특정 값보다 작은 지 비교

- **@ge**

```
RESPONSE_TIME "@ge 3000" "msg:'Slow response time'"
```

  - 변수 값이 특정 값보다 크거나 같은 지 비교

- **@le**

```
RESPONSE_BODY_SIZE "@le 1024" "msg:'Small response body'"
```

  - 변수 값이 특정 값보다 작거나 같은 지 비교

SecRule 연산자

문자열 연산자

- **@streq**

```
REQUEST_METHOD "@streq POST" "msg:'POST request detected'"
```

  - 문자열의 일치 여부 확인

- **@strmatch**

```
REQUEST_URI "@strmatch /admin/*" "msg:'Admin path access'"
```

  - 문자열이 와일드카드 패턴과 일치하는지 비교

파일 관련 연산자

- **@inspectFile**

```
FILES_TMP_CONTENT "@inspectFile ban.txt" "msg:'Forbidden content'"
```

  - 파일 내용을 검사하여 특정 조건을 만족하는지 확인

SecRule 연산자

기타 연산자

- **@within**

```
ARGS:id "@within 100 200 300" "msg:'ID within range'"
```

  - 문자열의 일치 여부 확인

- **@validateByteRange**

```
ARGS "validateByteRange 0-255" "msg:'Invalid byte range'"
```

  - 문자열이 와일드카드 패턴과 일치하는지 비교

사용자 정의 규칙

특정 IP 주소 차단

```
SecRule REMOTE_ADDR "@ipMatch 192.168.1.100" "id:'300001',phase:1,deny,log,status:403,
msg:'IP Address Blocked'"
```

- **REMOTE_ADDR :** 클라이언트의 IP 주소
- **@ipMatch 연산자 :** IP 주소 비교

특정 URL 패턴 필터링

```
SecRule REQUEST_URI "@contains /admin" "id:'300002',phase:1,deny,log,status:403,msg:
'Admin Access Blocked'"
```

- **REQUEST_URI :** 요청된 URI
- **@contains 연산자 :** 지정된 문자열이 포함되어 있는지 검사

사용자 정의 규칙

로그 파일 분석

/var/log/httpd/modsec_audit.log

반복적인 공격 패턴 식별

- **SQL Injection 공격 패턴 :** union select
- **XSS 공격 패턴 :** <script>

식별된 공격 패턴을 토대로 규칙 작성

```
SecRule ARGS "union select" "id:'300003',phase:2,t:lowercase,deny,log,status:403,msg:
'Repeated SQL Injection Attack Detected'"
```

```
SecRule ARGS "<script>" "id:'300004',phase:2,deny,log,status:403,msg:'Repeated XSS
Attack Detected'"
```

실시간 모니터링 및 알림 설정

실시간 모니터링 및 알림 규칙

```
SecRule REQUEST_URI "@rx .*" "id:'300005',phase:1,log,auditlog,pass,msg:'Request
Received'"
```

- @rx .*는 모든 요청 URI를 대상으로 하므로, 모든 요청을 감사 로그에 기록하는 규칙을 의미

실시간 알림 설정

- ModSecurity는 자체적으로 실시간 알림을 보내는 기능이 없으므로,
  로그 데이터를 기반으로 ELK Stack 등 외부 도구와 연동하여 알림 설정

ELK Stack 설정

Logstash 설정

```
input {
  file {
    path => "/var/log/httpd/modsec_audit.log"
    start_position => "beginning"
  }
}
filter {
  grok {
    match => { "message" => "%{COMMONAPACHELOG}" }
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

Kibana 설정

- 대시보드를 통해 실시간 로그 모니터링 및 알림 설정

OWASP CRS

OWASP CRS

ModSecurity에서 사용할 수 있는 기본 보안 규칙 세트로, 다양한 웹 응용프로그램 공격에 대해 사전에 정의된 규칙 제공

OWASP CRS 다운로드

```
sudo yum install mod_security_crs
```

기본 설정 파일 편집 (/etc/httpd/conf.d/mod_security.conf)

```
IncludeOptional /usr/local/modsecurity-crs/crs-setup.conf
IncludeOptional /usr/local/modsecurity-crs/rules/*.conf
```

* IncludeOptional 지시어를 사용하여 CRS 설정 파일과 규칙 파일을 포함

Apache 재시작

```
sudo systemctl restart httpd
```

OWASP CRS 커스터마이징

기본 규칙에서 제외할 규칙 식별

```
SecRuleRemoveById 981176
```

특정 상황에 맞게 규칙 조정

```
SecRule REQUEST_URI "@beginsWith /admin" "id:'200006',phase:2,log,allow,status:200,msg:
'Admin access granted'"
```

실시간 공격 탐지의 이해

실시간 공격 탐지의 원리

- **HTTP 트래픽 분석**
  - WAF는 웹 응용프로그램으로 들어오는 모든 HTTP/HTTPS 요청을 실시간으로 분석하여 잠재적인 공격을 탐지
  - 패턴 매칭 및 이상징후 분석을 통해 비정상적인 트래픽을 탐지
- **로그 모니터링**
  - 실시간으로 생성되는 로그를 분석하여 공격 패턴을 탐지
  - 로그 데이터에서 비정상적인 활동을 식별하고 경고를 생성
- **행동 기반 탐지**
  - 정상적인 사용 패턴을 학습하고, 정상적인 사용자 행동과 비정상적인 행동을 구분하여 공격을 탐지
  - 기계학습 알고리즘을 사용하여 정상적인 트래픽과 공격 트래픽을 구분

실시간 공격 탐지의 이해

실시간 공격 탐지의 중요성

- **신속한 대응**
  - 실시간으로 공격을 탐지하면 빠르게 대응하여 피해를 최소화하는 효과
    - **예 :** SQL Injection 공격이 탐지되면 즉시 차단하여 데이터베이스를 보호
- **지속적인 보호**
  - 실시간 모니터링을 통해 지속적으로 시스템을 보호하는 효과
    - **예 :** 지속적인 XSS 공격 시도를 실시간으로 탐지하고 차단하여 웹 애플리케이션을 안전하게 유지
- **위험 완화**
  - 잠재적인 보안 위협을 조기에 발견하고 대응함으로써 위험을 완화하는 효과
    - **예 :** CSRF 공격 시도를 실시간으로 감지하여 무단 요청을 차단합니다.

WAF의 실시간 탐지 및 차단

WAF의 실시간 탐지 및 차단의 원리

- **패턴 매칭**
  - 사전에 정의된 규칙을 기반으로 HTTP 요청을 분석하고, 악성 패턴이 일치하면 차단
  - 알려진 공격 패턴을 실시간으로 감지하고 차단하여 웹 응용프로그램을 보호
  - SQL Injection, XSS, CSRF 등의 공격 패턴을 탐지하고 차단
- **이상징후 분석**
  - 비정상 트래픽을 탐지하여 비정상적인 활동을 차단
  - 비정상적인 트래픽을 실시간으로 감지하고 차단하여 웹 응용프로그램을 보호
  - 정상적인 사용자 트래픽을 기준으로 비정상적인 활동을 감지하고 차단

ELK Stack 이해

ELK Stack

Elasticsearch, Logstash, Kibana의 약자로, 실시간 로그 분석 및 시각화를 위한 통합 솔루션

구성요소

- **Elasticsearch**
  - 분산형 검색 및 분석 엔진으로, 수집된 로그 데이터를 저장하고 검색할 수 있는 기능 제공
    - **예 :** 실시간 검색, 분산 저장, 고성능 데이터 분석
- **Logstash**
  - 로그 및 이벤트 데이터 처리 파이프라인 도구로, 다양한 소스로부터 데이터를 수집하여 필터링 및 변환한 후 Elasticsearch로 전송
    - **예 :** 데이터 수집, 처리, 변환 및 출력
- **Kibana**
  - Elasticsearch 데이터를 시각화하는 도구로, Elasticsearch에 저장된 데이터를 기반으로 대시보드와 시각화를 제공
    - **예 :** 데이터 시각화, 대시보드 생성, 실시간 모니터링

ELK Stack 설치

Elasticsearch 설치

```
sudo rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
sudo sh -c 'echo "[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md" > /etc/yum.repos.d/elasticsearch.repo'
sudo yum install elasticsearch
sudo systemctl enable elasticsearch
sudo systemctl start elasticsearch
```

ELK Stack 설치

Logstash 설치

```
sudo yum install logstash
sudo systemctl enable logstash
sudo systemctl start logstash
```

Kibana 설치

```
sudo yum install kibana
sudo systemctl enable kibana
sudo systemctl start kibana
```

ELK Stack 설정

Logstash 설정

```
input {
  file {
    path => "/var/log/httpd/modsec_audit.log"
    start_position => "beginning"
  }
}
filter {
  grok {
    match => { "message" => "%{COMMONAPACHELOG}" }
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

Kibana 설정

- 대시보드를 통해 실시간 로그 모니터링 및 알림 설정