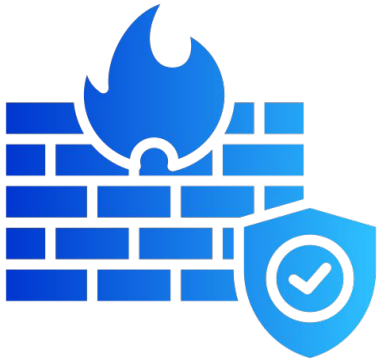
Web Application Firewall Development

안수현 / 이우진 / 임정은 / 정채원



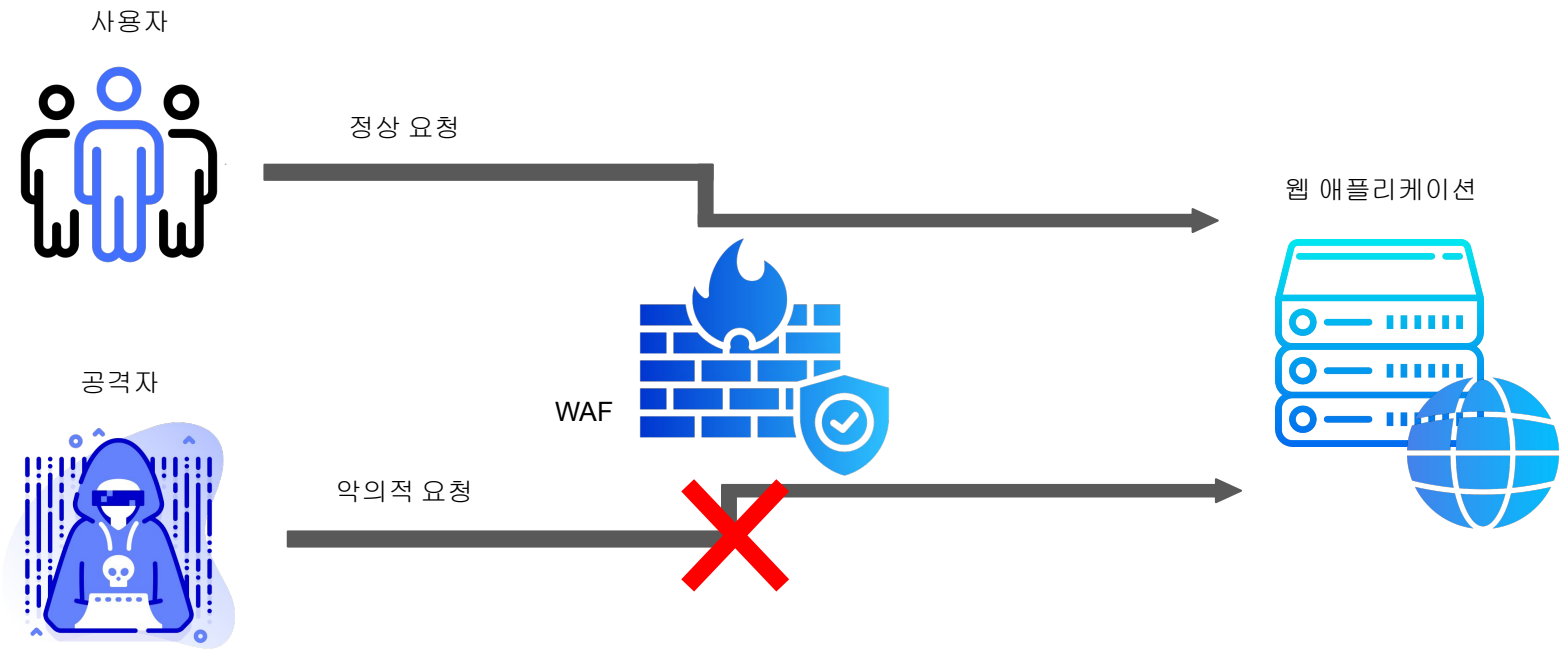
INDEX

Web Application Firewall

WAF 웹 애플리케이션 방화벽 솔루션

- 웹 애플리케이션을 해킹 공격으로부터 보호
- 웹 애플리케이션으로 가는 트래픽을 모니터링 및 필터링
- OSI 모형 중 7계층(애플리케이션 계층)에서 동작

Architecture



TODO

- HTTP 프록시 구현
- 정규표현식으로 탐지 규칙 작성
- 규칙 파싱 및 적용
- easy?

Development

Ruleset parsing

YAML 또는 JSON으로 작성된 규칙들을 파싱

```python
class Rule():
    def __init__(self, ruleset: str):
        rule = yaml.safe_load(ruleset)

        assert check(rule, "name", str)
        assert check(rule, "description", str)
        assert check(rule, "severity", str)
        assert check(rule, "action", str)
        assert check(rule, "definition", dict)

        self.name = rule["name"]
        self.description = rule["description"]
        self.severity = Severity.serialize(rule["severity"])
        self.action = Action.serialize(rule["action"])
        self.definition = Definition(rule["definition"])

        self.detector = Detector(self.definition)
        self.detect = self.detector.detect
```

규칙 설명 파싱

```python
def __init__(self, definition: dict):
    self.ipv4 = Definition._ipv4(definition, "ipv4Network")
    self.ipv6 = Definition._ipv6(definition, "ipv6Network")
    self.method = Definition._str(definition, "method")
    self.header = Definition._header(definition, "header")
    self.cookie = Definition._group(definition, "cookie")
    self.url_resource = Definition._str(definition, "urlResource")
    self.query_string = Definition._str(definition, "queryString")
    self.query_parameter = Definition._group(definition, "queryParameter")
    self.body = Definition._str(definition, "body")
    self.json_body = Definition._group(definition, "jsonBody")
```

탐지 규칙 파싱

HTTP Proxy

요청을 처리할 HTTP 프록시를 구현

```python
async def _session(self, reader, writer):
    try:
        ip, port = writer.get_extra_info("peername")

        data = await HttpProxy._http(reader)
    except Exception as e:
        return

    result = self.handler(data, ip)

    try:
        if result:
            client = await asyncio.open_connection(self.rhost, self.rport)
            await self._proxy(writer, client, data, ip)
        else:
            await self._block(writer)

        await writer.drain()

        writer.close()
    except Exception as e:
        return
```

HTTP 요청 전달

```python
    if mode == None:
        return data
    elif mode == TE:
        while True:
            i = 0
            chunk = b""
            while i < HEADER_MAX_SIZE:
                chunk += await r.read(1)
                i += 1
                if chunk[-2:] == b"\r\n":
                    length = int(chunk.strip(), 16)
                    break
            else:
                return False
            if length == 0:
                return data
            data += await r.read(length)
            await r.read(2)
        return data
    elif mode == CL:
        data += await r.read(length)
        return data
    else:
        return False # Unreachable
```

HTTP 요청 길이 처리

```python
if "x-forwarded-for" in self.header:
    ip_list.extend([x.strip() for x in self.headers["x-forwarded-for"].split(",")])

ip_list.append(ip)

for ip_addr in ip_list:
    try:
        self.ipv4.append(IPv4Address(ip_addr))
    except:
        self.ipv6.append(IPv6Address(ip_addr))

if "cookie" in self.headers:
    self.cookie = HttpRequest._cookie(self.headers["cookie"])
else:
    self.cookie = None

self.url_resource = unquote(urlparse(self.path).path)
self.query_string = urlparse(self.path).query

self.query_parameter = HttpRequest._query_parameter(self.query_string)

self.query_string = unquote(self.query_string)

self.body = unquote(self.rfile.read().decode("latin1"))

try:
    self.json_body = json.loads(self.body)
except ValueError:
    self.json_body = None
```

HTTP 요청 파싱

Writing rules

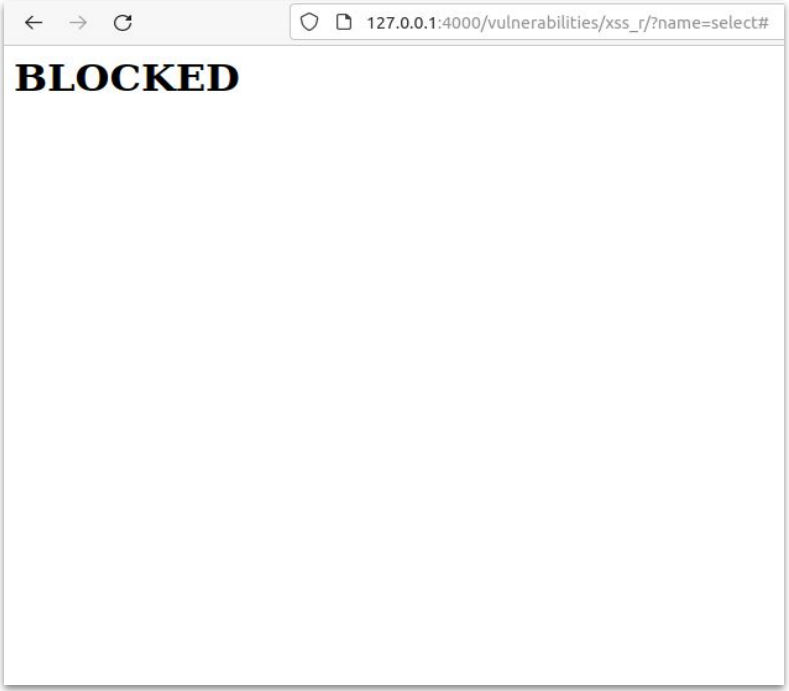
Writing rules

웹 해킹 공격을 탐지하고 차단할 규칙을 작성 고려할 점이 생각보다 많다

- **오탐**이나 **미탐**이 많은가? -> 하지만 둘 다 줄이기 힘들다
- 탐지를 위한 정규표현식이 너무 무거운건 아닌가?

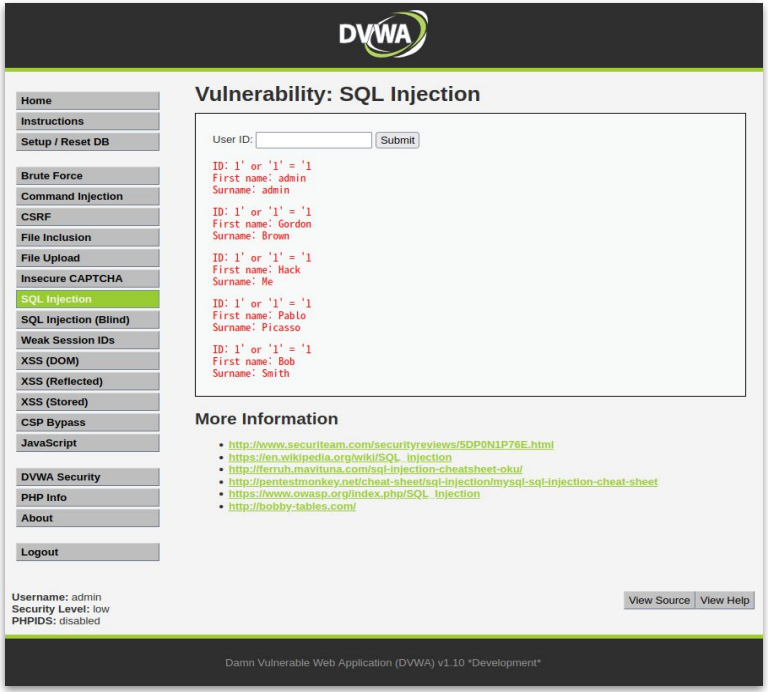
False positive

오탐 공격 시도가 아님에도 차단하는 경우



False negative

미탐 공격 시도를 탐지하지 못하는 경우



SQL INJECTION

ex) 1’ or ‘1’ = ‘1

SSRF

ex) file:///etc/passwd

XSS

ex) <script>alert(1);</script>

```
name: SQL_INJECTION_QUERY
description: SQL injection attack on query parameters
severity: HIGH
action: BLOCK

definition:
  queryParameter:
    "*": |
      (?i)(['"]|/\*|\*/|--|#|
      SELECT|INSERT|UPDATE|DELETE|
      WHERE|FROM|ORDER BY|GROUP BY|
      HAVING|UNION|SLEEP|SUBSTR)
```

```
name: SSRF_QUERY
description: SSRF attack on query parameters
severity: HIGH
action: BLOCK

definition:
  queryParameter:
    "*": |
      (?i)(https?://[^a-z]|
      0\.0\.0\.0|127\..*|
      ::1|localhost|jar:|
      [^https?]://)
```

```
name: XSS_QUERY
description: XSS attack on query string
severity: MEDIUM
action: BLOCK

definition:
  queryString: |
    (?i)(javascript|
    on[a-z]+=|style=|
    <script>|eval)
```

Conclusion

- WAF 등의 보안 솔루션을 개발하는 것은 생각보다 고려할 점이 많다
- 오탐과 미탐을 동시에 줄이는 것은 상당히 어렵다
- 오탐과 미탐을 줄이기 위해서는 패턴 기반을 넘어서 Ai 기술의 도입이 필요하다

Resources

- 정탐, 미탐, 오탐 이란?
- 웹 방화벽의 개념과 원리
- 웹 애플리케이션 방화벽 AWS WAF

Github

https://github.com/movptr06/Web-Security

Thank you

This presentation uses pixabay and flaticon.

Q&A