

로그 분석 / 스노트 룰 스터디

권문정

Contents

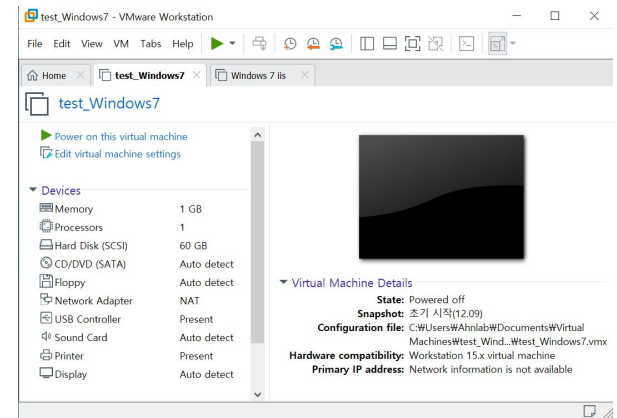
- 01** 웹 공격 로그 분석을 위한 환경 구성
- 02** Apache access.log 분석 – GET
- 03** Live HTTP Headers를 활용한 로그 분석 – POST
- 04** Snort 탐지 룰 학습
- 05** Snort 탐지 룰 만들기

01. 웹 공격 로그 분석을 위한 환경 구성

■ 실습 환경

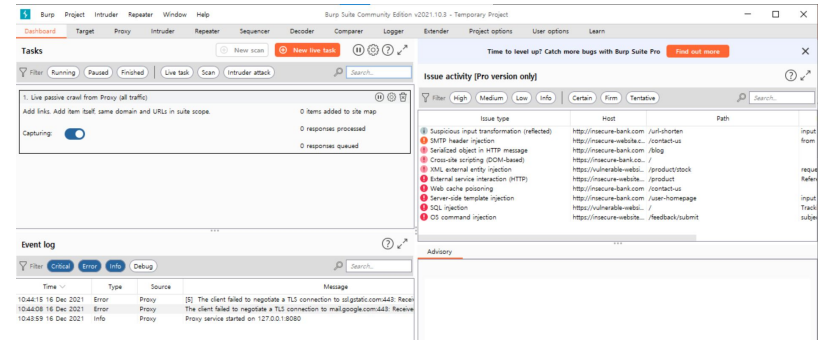
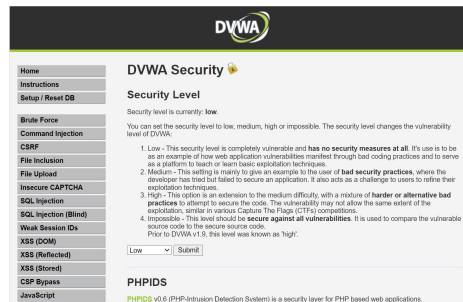
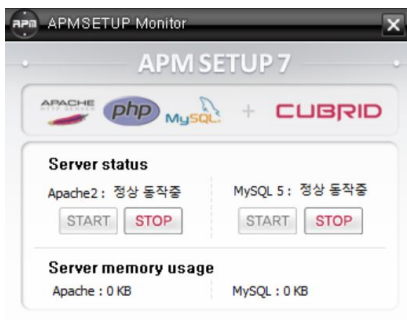
1. 가상 환경 구축

- VMware Workstation Pro(free trial 30일)
- OS : Window10 64 bit



2. 실습 도구

- APMSETUP7 (window 32 bit)
 - Apache(2.2.14)
 - PHP(5.1.41-community)
 - MySQL5.1.41-community MySQL Community Server (GPL)
- Damn Vulnerable Web Application (DVWA)
 - : 취약점 진단/모의해킹을 할 수 있는 웹 어플리케이션. PHP/MySQL 환경. Security Level : Low로 설정
- Burp Suite
 - : 프록시(Proxy)를 사용하여 네트워크에서 통신하는 패킷을 가로채 분석 및 조작, 취약점을 확인하는 도구



02. Apache access.log 분석 – GET

■ Brute Force Attack

- 입력한 비밀번호는 GET 방식으로 password 파라미터에 담겨 전달됨
- password 파라미터에 여러 값들을 입력하여 수십 번의 로그인 시도를 하는 것으로 보아, Username이 'admin'인 계정의 비밀번호를 알아내기 위해 무차별 대입 공격을 하는 것으로 추정

```
127.0.0.1 - - [21/Dec/2021:09:36:51 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=1&Login=Login HTTP/1.1" 200 4241
```

```
127.0.0.1 - - [21/Dec/2021:09:36:51 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=qewr&Login=Login HTTP/1.1" 200 4241
```

```
127.0.0.1 - - [21/Dec/2021:09:36:51 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=1234&Login=Login HTTP/1.1" 200 4241
```

```
127.0.0.1 - - [21/Dec/2021:09:36:52 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=asdf&Login=Login HTTP/1.1" 200 4241
```

```
127.0.0.1 - - [21/Dec/2021:09:36:52 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=abcd&Login=Login HTTP/1.1" 200 4241
```

```
127.0.0.1 - - [21/Dec/2021:09:36:52 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1" 200 4284
```

```
127.0.0.1 - - [21/Dec/2021:09:36:53 +0900] "GET  
/dvwa/vulnerabilities/brute/?username=admin&password=qwer1234&Login=Login HTTP/1.1" 200 4241
```

02. Apache access.log 분석 – GET

■ CSRF

- 비밀번호가 GET 방식으로 변경됨
- 변경되는 비밀번호의 값이 password_new 와 password_conf 파라미터에 담겨 전달됨
- 서버의 입장에서는 세션/쿠키에 담긴 로그인 정보가 검증 없이 전달되므로, 비밀번호 변경 요청이 사용자의 요청인지, 공격자가 로그인 정보를 가로채서 시도한 요청인지 알 수 없음
- 본 로그는 웹 공격을 위한 환경에서 생성되었으므로 경로명이 표시되어 명시적으로 알 수 있으나, 실제 상황에서 로그가 표시되었을 때 어떻게 판단해야 할지 어려움이 있을 것으로 추정
 - 검증되지 않은 요청을 서버가 받아들이는 것과 마찬가지로 분석자 또한 눈치채지 못할 수 있을 가능성 존재
 - 의심스러운 경우, 사용자의 중요한 상태를 바꾸는 요청 (ex. 비밀번호, 전화번호 변경) 시에 CAPTCHA, 현재 비밀번호 입력 등의 검증 절차가 구비되어 있는지 확인 필요
 - 관리자 계정의 정보 변경 시, 접근 권한과 기록을 주의 깊게 확인할 필요가 있음

127.0.0.1 - - [21/Dec/2021:09:40:59 +0900] "GET /dvwa/vulnerabilities/csrf/?password_new=abcd&password_conf=abcd&Change=Change HTTP/1.1" 200 4494

127.0.0.1 - - [21/Dec/2021:09:47:15 +0900] "GET /dvwa/vulnerabilities/csrf/?password_new=csrf&password_conf=csrf&Change=Change HTTP/1.1" 200 4494

127.0.0.1 - - [21/Dec/2021:10:48:05 +0900] "GET /dvwa/vulnerabilities/csrf/?password_new=csrf&password_conf=csrf&Change=Change HTTP/1.1" 200 4494

127.0.0.1 - - [21/Dec/2021:10:48:22 +0900] "GET /dvwa/vulnerabilities/csrf/?password_new=csrf&password_conf=csrf&Change=Change HTTP/1.1" 200 4494

127.0.0.1 - - [21/Dec/2021:13:08:46 +0900] "GET /dvwa/vulnerabilities/csrf/?password_new=csrfz&password_conf=csrfz&Change=Change HTTP/1.1" 200 4494

02. Apache access.log 분석 – GET

■ SQL Injection

- WHERE 절에서 1=1 라는 참일 수밖에 없는 문장을 OR와 함께 사용하여 모두 참인 문장을 삽입
- 뒷 부분은 #를 사용하여 주석처리 해줌으로써 어떤 비밀번호를 입력하든 결과를 반환 받을 수 있음
- SQL Injection의 가장 기본적인 형태
- GET 방식으로 공격이 진행되므로 값이 URL의 파라미터로 전달 됨. 즉, URL 인코딩이 된 문자열을 확인 가능
→ URL Decoder를 사용하여 문자열 해독 후 해석 필요

127.0.0.1 - - [21/Dec/2021:09:47:28 +0900]

"GET/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+%23&Submit=Submit HTTP/1.1" 200 438

→ id=' OR 1=1 #

127.0.0.1 - - [21/Dec/2021:09:50:12 +0900]

"GET/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+%23&Submit=Submit HTTP/1.1" 200 438

→ id=' OR 1=1 #

127.0.0.1 - - [21/Dec/2021:10:52:44 +0900]

"GET/dvwa/vulnerabilities/sqli/?id=%27+OR+1%3D1+%23&Submit=Submit HTTP/1.1" 200 4388

→ id=' OR 1=1 #

02. Apache access.log 분석 – GET

■ Blind SQL Injection

- id가 1인 사용자가 있는지, 없는지 알아내고 있음
- AND 연산이 실행되고 있는지 확인하여 SQL 쿼리문이 뒤에 있는지 확인하고 있음.
WHY? SQL 쿼리문이 뒤에 있다면 SQL Injection으로 DB의 정보를 캐낼 수 있으므로.

```
127.0.0.1 - - [21/Dec/2021:09:51:29 +0900] "GET /dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit HTTP/1.1" 200 4155
```

→ id=1

```
127.0.0.1 - - [21/Dec/2021:09:51:41 +0900] "GET /dvwa/vulnerabilities/sqli_blind/?id=1%27+AND+1%3D2&Submit=Submit HTTP/1.1" 200 5425
```

→ id=1' AND 1=2

```
127.0.0.1 - - [21/Dec/2021:09:51:56 +0900] "GET /dvwa/vulnerabilities/sqli_blind/?id=1%27+AND+1%3D2+%23&Submit=Submit HTTP/1.1" 404 4161
```

→ id=1' AND 1=2 #

02. Apache access.log 분석 – GET

■ Blind SQL Injection

- 앞 부분의 문장을 모두 참으로 만들고, 뒷부분의 문장에 `database()`, `length()`함수를 통해 데이터베이스 이름의 길이를 확인하고 있음.
- 숫자가 4에서 멈춘 것으로 보아 데이터베이스의 이름은 4글자인 듯 함.

* `database()` : 데이터베이스 이름을 반환해주는 함수

* `length()` : 문자열의 길이를 반환해주는 함수

```
127.0.0.1 - - [21/Dec/2021:10:44:14 +0900] "GET
```

```
/dvwa/vulnerabilities/sqli_blind/?id=+1%27+and+length%28database%28%29%29+%3D+5%23&Submit=Submit HTTP/1.1" 404 4161
```

→ `id= 1' and length(database()) = 5#`

```
127.0.0.1 - - [21/Dec/2021:10:44:17 +0900] "GET
```

```
/dvwa/vulnerabilities/sqli_blind/?id=+1%27+and+length%28database%28%29%29+%3D+4+%23&Submit=Submit HTTP/1.1" 200 4155
```

→ `id= 1' and length(database()) = 4 #`

02. Apache access.log 분석 – GET

■ Blind SQL Injection

- 앞 부분의 문장을 모두 참으로 만들고, 뒷부분의 문장에 `database()`, `substr()` 함수를 통해 데이터베이스 이름의 철자를 확인하고 있음.
- * `substr()` : 반환하는 문자를 알려주는 함수
- * `substr(database(),1,1)`는 `database()`가 반환하는 문자열의 첫 번째 글자를 반환한다.
 - 즉, 데이터베이스 이름의 첫 번째 글자를 알아낸다.
 - 이런 식으로 4번째 글자까지 알아낸다.
 - 데이터베이스 이름을 확인하는 시도가 어디에서 멈췄는지 확인하여 1~4번째 글자가 d, v, w, a임을 알아냈다.

```
127.0.0.1 - - [21/Dec/2021:10:44:45 +0900] "GET
/dvwa/vulnerabilities/sqli_blind/?id=+1%27+and+substr%28database%28%29%2C1%2C1%29%3D+%27a%2
7+%23&Submit=Submit HTTP/1.1" 404 4161
→ id= 1' and substr(database(),1,1)= 'a' #
```

```
127.0.0.1 - - [21/Dec/2021:10:44:49 +0900] "GET
/dvwa/vulnerabilities/sqli_blind/?id=+1%27+and+substr%28database%28%29%2C1%2C1%29%3D+%27b%2
7+%23&Submit=Submit HTTP/1.1" 404 4161
→ sqli_blind/?id= 1' and substr(database(),1,1)= 'b' #
```

```
127.0.0.1 - - [21/Dec/2021:10:44:53 +0900] "GET
/dvwa/vulnerabilities/sqli_blind/?id=+1%27+and+substr%28database%28%29%2C1%2C1%29%3D+%27c%27
+%23&Submit=Submit HTTP/1.1" 404 4161
→ id= 1' and substr(database(),1,1)= 'c' #
```

02. Apache access.log 분석 – GET

■ Blind SQL Injection

- users 테이블의 user가 'admin'인 첫 번째 사용자의 password 첫 글자를 확인
- 이런 식으로 password의 문자열 자리 수마다 문자를 대입하여 확인해 봄

```
127.0.0.1 - - [21/Dec/2021:10:53:17 +0900] "GET  
/dvwa/vulnerabilities/sql_i_blind/?id=1%27+AND+substr%28+%28SELECT+password+FROM+users+WHERE+  
user%3D%27admin%27+LIMIT+0%2C1%29%2C1%2C1%29+%3D+%27a%27+%23&Submit=Submit  
HTTP/1.1" 404 4161
```

→ **id=1' AND substr((SELECT password FROM users WHERE user='admin' LIMIT 0,1),1,1) = 'a' #**

```
127.0.0.1 - - [21/Dec/2021:10:53:20 +0900] "GET  
/dvwa/vulnerabilities/sql_i_blind/?id=1%27+AND+substr%28+%28SELECT+password+FROM+users+WHERE+  
user%3D%27admin%27+LIMIT+0%2C1%29%2C1%2C1%29+%3D+%27b%27+%23&Submit=Submit  
HTTP/1.1" 404 4161
```

→ **id=1' AND substr((SELECT password FROM users WHERE user='admin' LIMIT 0,1),1,1) = 'b' #**

```
127.0.0.1 - - [21/Dec/2021:10:53:23 +0900] "GET  
/dvwa/vulnerabilities/sql_i_blind/?id=1%27+AND+substr%28+%28SELECT+password+FROM+users+WHERE+  
user%3D%27admin%27+LIMIT+0%2C1%29%2C1%2C1%29+%3D+%27c%27+%23&Submit=Submit  
HTTP/1.1" 200 4155
```

→ **id=1' AND substr((SELECT password FROM users WHERE user='admin' LIMIT 0,1),1,1) = 'c' #**

02. Apache access.log 분석 – GET

■ XSS

- URL에 HTML태그가 포함된 것으로 보아 XSS임을 유추할 수 있음
- 'xss', 'script'라는 내용을 표시하는 alert 창 띄우기, 쿠키 내용을 알려주는 alert창 띄우기 등의 로그가 표시됨

```
127.0.0.1 - - [21/Dec/2021:10:47:37 +0900] "GET  
/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%27xss%27%29%3C%2Fscript%3E HTTP/1.1"  
200 4244  
→ name=<script>alert('xss')</script>
```

```
127.0.0.1 - - [21/Dec/2021:10:48:02 +0900] "GET  
/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%27script%27%29%3C%2Fscript%3E HTTP/1.1"  
200 4247  
→ name=<script>alert('script')</script>
```

```
127.0.0.1 - - [21/Dec/2021:09:48:00 +0900] "GET  
/dvwa/vulnerabilities/xss_r/?name=%3CSCRiPt%3Ealert%28document.cookie%29%3C%2Fscript%3E  
HTTP/1.1" 200 4254  
→ name=<SCRiPt>alert(document.cookie)</script>
```

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ POST 방식으로 전송되는 데이터 확인

- DVWA 로그인 시에 데이터를 HTTP POST 방식으로 주고받는 것을 알 수 있음
- POST는 GET과 다르게 access.log에 기록이 상세히 남지 않아 분석이 어려움
→ Chrome의 Live HTTP Headers를 설치하여 바디의 폼 데이터 확인 필요

localhost/DVWA/index.php

DevTools is now available in Korean! Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network Performance Memory Application >> 2

Filter ☐ Invert ☐ Hide data URLs

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies ☐ Blocked Requests

☐ 3rd-party requests

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110

Name ☐ Headers ☐ Payload ☐ Preview ☐ Response ☐ Initiator ☐ Timing ☐ Cookies

☒ add_event_listeners.js

☒ add_event_listeners.js

☒ dvwaPage.js

☒ index.php

☒ login.php

☒ logo.png

☒ main.css

General

Request URL: http://localhost/DVWA/login.php

Request Method: POST

Status Code: 302 Found

Remote Address: 127.0.0.1:80

Referrer Policy: strict-origin-when-cross-origin

Response Headers View source

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Connection: Keep-Alive

Content-Length: 0

Content-Type: text/html

Date: Mon, 20 Dec 2021 07:44:24 GMT

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Keep-Alive: timeout=5, max=100

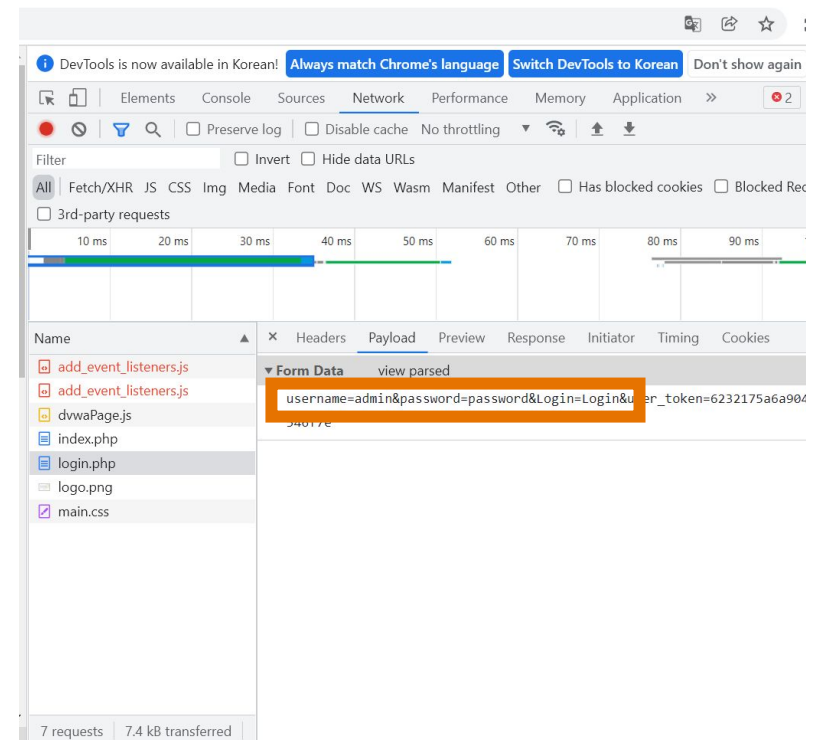
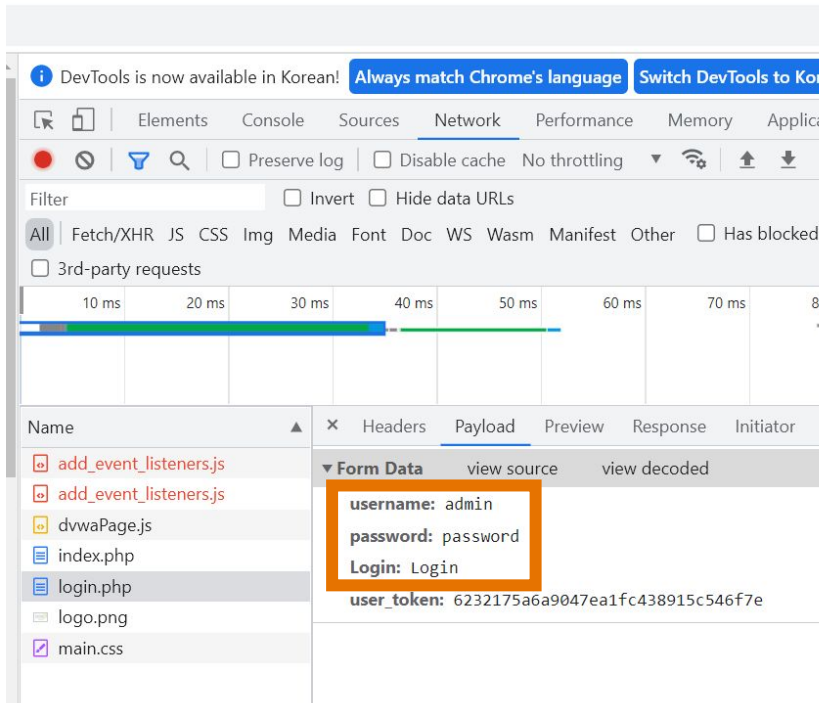
Location: index.php

7 requests 7.4 kB transferred

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ POST 방식으로 전송되는 데이터 확인

- 폼 데이터를 확인하여 로그인할 때 사용되는 **username**과 **password**를 알아낼 수 있음
- POST 방식은 데이터를 헤더가 아닌, 본문에 담아서 보내므로 URL으로 조작 불가



02. Live HTTP Headers를 활용한 로그 분석 – POST

■ CSRF

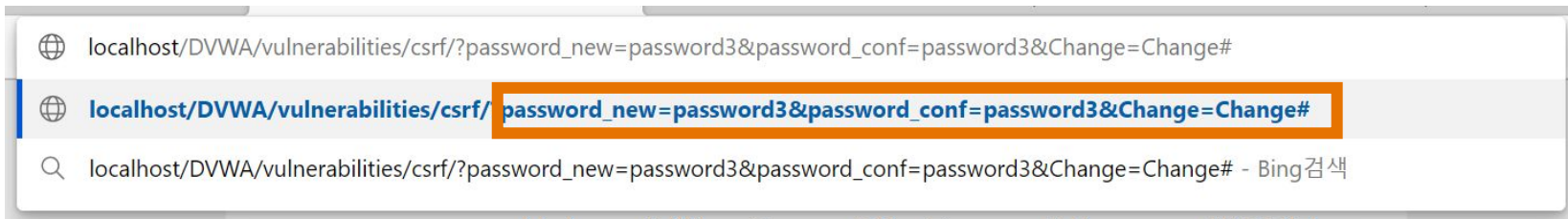
- 사용자가 로그인한 상태에서 비밀번호를 바꾸는 요청을 수행
세션/쿠키가 남아있는 상태에서 비밀번호 검증 없이 비밀번호를 수정
- 공격자의 입장에서 비밀번호를 수정하자, 비밀번호의 값이 GET 방식으로 전달되어 URL에 그대로 노출됨



02. Live HTTP Headers를 활용한 로그 분석 – POST

■ CSRF

- URL 창에서 비밀번호 파라미터에 맞게 값을 바꿔주고 **Enter**키를 누르자, 비밀번호를 바꾸는 페이지에서 비밀번호가 변경되었다는 안내 메시지가 표시됨



02. Live HTTP Headers를 활용한 로그 분석 – POST

■ CSRF

- 로그인 테스트 페이지는 POST 방식으로 데이터를 주고 받음

The screenshot displays a web browser window with the URL `localhost/DVWA/vulnerabilities/csrf/test_credentials.php`. The page title is "Test Credentials" and the subtitle is "Vulnerabilities/CSRF". It contains a login form with fields for "Username" (filled with "admin") and "Password" (filled with "*****"), and a "Login" button. A green message above the form states "Valid password for 'admin'".

Overlaid on the right is the Chrome DevTools Network tab. It shows a list of requests, with the selected request being `test_credentials.php` (source: `source.css`). The "Headers" sub-tab is active, showing the following details:

- Request Method:** POST (highlighted with an orange box)
- Status Code:** 200 OK
- Remote Address:** 127.0.0.1:80
- Referrer Policy:** strict-origin-when-cross-origin

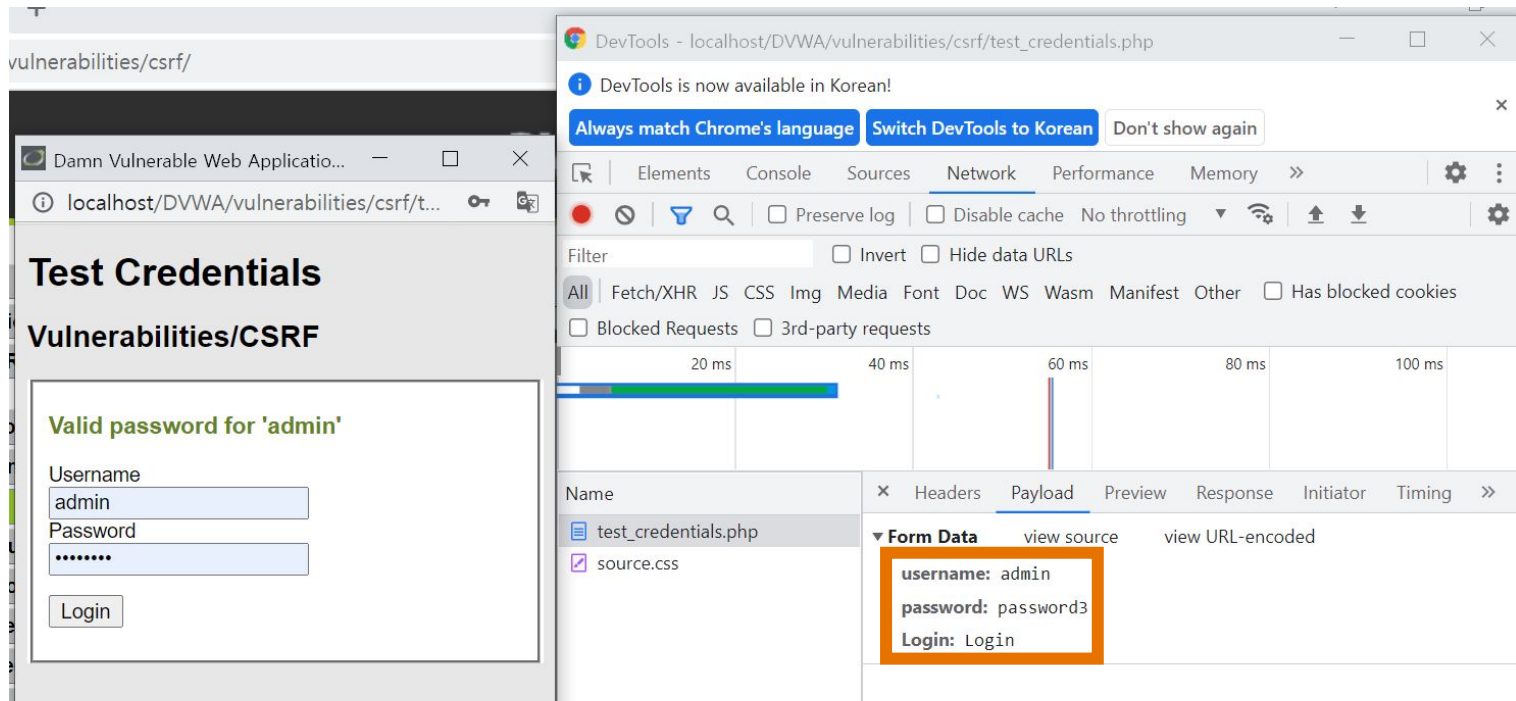
Below the headers, the "Response Headers" section is visible, showing:

- Cache-Control:** no-cache, must-revalidate
- Connection:** Keep-Alive
- Content-Length:** 1107
- Content-Type:** text/html; charset=utf-8
- Date:** Tue, 21 Dec 2021 00:37:42 GMT

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ CSRF

- 로그인 테스트 페이지에서 방금 URL에서 수정한 비밀번호를 입력하고 로그인 시도를 하자 비밀번호를 올바르게 입력했다고 안내함
- 폼 데이터에서 비밀번호를 방금 공격자가 위조한 값으로 입력한 것을 확인할 수 있음.



02. Live HTTP Headers를 활용한 로그 분석 - POST

■ Command Injection

- IP를 입력하는 창에 자신의 IP를 입력하고 Submit 버튼을 누르면, 그 값이 POST 방식으로 전달되어 수행됨
- 웹페이지에 CMD창에서 ping + IP 명령어를 입력하는 것과 동일한 결과가 표시됨
→ ping 명령어를 실행하는 웹 페이지라고 추측할 수 있음

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface with the 'Command Injection' vulnerability selected. The 'Ping a device' section has an input field containing '172.29.32.1' and a 'Submit' button. The output shows a successful ping command execution. The Chrome DevTools Network tab is open, showing a POST request to 'localhost/DVWA/vulnerabilities/exec/'. The 'Form Data' section of the request is highlighted, showing the 'ip' parameter set to '172.29.32.1' and the 'Submit' button value.

172.29.32.1

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Ping 172.29.32.1 32바이트 데이터 사용:
172.29.32.1의 응답: 바이트=32 시간=4ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=3ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=3ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=2ms TTL=255

172.29.32.1에 대한 Ping 통계:
패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
왕복 시간(밀리초):
최소 = 2ms, 최대 = 4ms, 평균 = 3ms

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>

Network

Filter: ☐ Invert ☐ Hide data URLs

☐ Blocked Requests ☐ 3rd-party requests

Headers

General

Request URL: http://localhost/DVWA/vulnerabilities/exec/
Request Method: POST
Status Code: 200 OK
Remote Address: 127.0.0.1:80
Referrer Policy: strict-origin-when-cross-origin

Form Data

ip: 172.29.32.1
Submit: Submit

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ Command Injection

- ping 명령어를 입력한 뒤 윈도우 운영체제에서 명령어 여러 개를 실행할 수 있게 해주는 &&를 사용하여 ipconfig 명령어를 입력하자, ping 명령어가 실행된 후 ipconfig 명령어가 함께 실행 됨
→ Command Injection 취약점이 존재함을 알 수 있음. POST 방식으로 전달되므로 폼 데이터를 통해 확인 가능

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface at the URL `localhost/DVWA/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection" and the sub-header is "Ping a device". There is a text input field labeled "Enter an IP address:" and a "Submit" button. An orange callout bubble points to the input field containing the text `172.29.32.1 && ipconfig`.

The page content displays the results of a ping command: "Ping 172.29.32.1 32바이트 데이터 사용: 172.29.32.1의 응답: 바이트=32 시간=3ms TTL=255". Below this, it shows statistics for the ping: "172.29.32.1에 대한 Ping 통계: 패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실), 왕복 시간(밀리초): 최소 = 3ms, 최대 = 4ms, 평균 = 3ms".

The DevTools Network tab is open, showing a list of resources. The "exec/" resource is selected, and the "Headers" tab is active. The "Request Method" is "POST". The "Response Headers" tab is also visible, showing "Status Code: 200 OK" and "Content-Type: text/html; charset=utf-8".

The "Payload" tab is also open, showing the "Form Data" section. The "ip" field contains the value `172.29.32.1 && ipconfig` and the "Submit" field contains the value `Submit`.

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ Command Injection

- ping 명령어를 입력한 뒤 **&&**를 사용하여 **dir** 명령어를 입력하자, ping 명령어가 실행된 후 **dir** 명령어가 함께 실행되어 현재 위치와 현재 위치에 있는 디렉토리 및 파일 목록들을 보여줌
→ **Command Injection** 공격으로 서버의 구조와 파일 내용을 알아내려고 시도하였음. 본 공격에 대한 데이터가 POST 형식으로 전달되므로, 공격이 시도 되었음을 폼 데이터를 통해 알 수 있음

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface with the 'Vulnerability: Command Injection' section selected. The 'Ping a device' form is visible, with the input field containing '172.29.32.1 && dir'. The 'Submit' button is highlighted. The page displays the results of the ping command, showing the IP address and the output of the 'dir' command. The DevTools Network tab is open, showing the request to 'exec/' with the payload 'ip: 172.29.32.1 && dir' and 'Submit: Submit'.

172.29.32.1 && dir

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

Ping 172.29.32.1 32바이트 데이터 사용:
172.29.32.1의 응답: 바이트=32 시간=6ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=4ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=4ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=6ms TTL=255

172.29.32.1에 대한 Ping 통계:
패킷: 보낸 = 4, 받은 = 4, 손실 = 0 (0% 손실),
왕복 시간(밀리초):
최소 = 4ms, 최대 = 6ms, 평균 = 4ms
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 987E-B6CF

C:\APM_Setup\htdocs\DVWA\vulnerabilities\exec 디렉터리

2021-12-15 오전 09:03

2021-12-15 오전 09:03

2021-12-15 오전 09:03

help
2021-12-01 오후 05:59 1,839 인
2021-12-15 오전 09:03

source
1개 파일
4개 디렉터리 167,693.6

DevTools is now available in Korean!

Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network Performance Memory >> ⚙ ⋮

Filter ☐ Preserve log ☐ Disable cache No throttling ☐ Invert ☐ Hide data URLs

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies

☐ Blocked Requests ☐ 3rd-party requests

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms

Name

exec/
main.css
dvwaPage.js
logo.png
add_event_listeners.js

Form Data view source view URL-encoded

ip: 172.29.32.1 && dir
Submit: Submit

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ Command Injection

- ping 명령어와 함께 현재 열린 포트를 확인하는 netstat 명령어를 사용한 Command Injection 공격이 시도되었음을 폼 데이터에서 확인 가능. 이렇게 포트 스캔 후에 취약점을 찾으려고 했음을 알 수 있음
(ex. 22, 3389 포트가 열려있는 경우, 원격 접속과 관련한 공격을 시도해볼 수 있음)

The screenshot shows the DVWA 'Vulnerability: Command Injection' page. The 'Ping a device' form has been submitted with the IP address '172.29.32.1' and the command '&& netstat'. The page displays the output of the ping command and a netstat table. The DevTools Network tab is open, showing the 'exec/' request with the payload 'ip: 172.29.32.1 && netstat -a' highlighted in an orange box.

172.29.32.1 && netstat

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Ping 172.29.32.1 32바이트 데이터 사용:
172.29.32.1의 응답: 바이트=32 시간=4ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=3ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=2ms TTL=255
172.29.32.1의 응답: 바이트=32 시간=4ms TTL=255

172.29.32.1에 대한 Ping 통계:
패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
왕복 시간(밀리초):
최소 = 2ms, 최대 = 4ms, 평균 = 3ms

활성 연결

프로토콜	로컬 주소	외부 주소	상태
TCP	0.0.0.0:80	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:135	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:443	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:445	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:902	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:912	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:3306	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:5040	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:5700	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:29919	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49664	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49665	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49666	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49667	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49668	DESKTOP-IKCLUKM:0	LISTENING
TCP	0.0.0.0:49676	DESKTOP-IKCLUKM:0	LISTENING
TCP	127.0.0.1:80	DESKTOP-IKCLUKM:64598	TIME_WAIT
TCP	127.0.0.1:80	DESKTOP-IKCLUKM:64640	TIME_WAIT
TCP	127.0.0.1:80	DESKTOP-IKCLUKM:64691	ESTABLISHED
TCP	127.0.0.1:80	DESKTOP-IKCLUKM:64692	ESTABLISHED

Network Tab:

Filter: ☐ Invert ☐ Hide data URLs

☐ Has blocked cookies ☐ Blocked Requests ☐ 3rd-party requests

20000 ms 40000 ms 60000 ms 80000 ms 100000 ms 120000 ms 1

Form Data view source view URL-encoded

ip: 172.29.32.1 && netstat -a
Submit: Submit

02. Live HTTP Headers를 활용한 로그 분석 – POST

■ Command Injection

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA...
localhost/DVWA/vulnerabilities/view_source.php?id=exec&security=low

Command Injection Source

vulnerabilities/exec/source/low.php

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];
    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }
    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

\$target = \$_REQUEST['ip']; 입력받은 IP주소가 target에 저장

\$cmd = shell_exec('ping ' . \$target); target을 포함해 시스템 명령어를 내림
ex) ping [IP주소] 로 시스템 명령어 실행

References

- SQL Injection 공격 구문/패턴
 - https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=is_king&logNo=221402635339%20ca%20
 - <https://woojoong2.tistory.com/39>
- Blind SQL Injection 공격 구문/패턴
 - <https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=okopok5019&logNo=221543684263%20%20%27%20UNION%20SELECT%201%20#>
 - <https://idh5202.tistory.com/640>
- ASCII Code 일람표
<https://m.blog.naver.com/ansdbtIs4067/220624120433>
- URL 인코더/디코더
<https://www.convertstring.com/ko/EncodeDecode/UrlDecode>