Lab 2: Detect SQL Injection Vulnerabilities using Various SQL Injection Detection Tools

Lab Scenario

By now, you will be familiar with various types of SQL injection attacks and their possible impact. To recap, the different kinds of SQL injection attacks include authentication bypass, information disclosure, compromised data integrity, compromised availability of data and remote code execution (which allows identity spoofing), damage to existing data, and the execution of system-level commands to cause a denial of service from the application.

As an ethical hacker or pen tester, you need to test your organization's web applications and services against SQL injection and other vulnerabilities, using various approaches and multiple techniques to ensure that your assessments, and the applications and services themselves, are robust.

In the previous lab, you learned how to use SQL injection attacks on the MSSQL server database to test for website vulnerabilities.

In this lab, you will learn how to test for SQL injection vulnerabilities using various other SQL injection detection tools.

Lab Objectives

- Detect SQL injection vulnerabilities using DSSS
- Detect SQL injection vulnerabilities using OWASP ZAP

Overview of SQL Injection Detection Tools

SQL injection detection tools help to discover SQL injection attacks by monitoring HTTP traffic, SQL injection attack vectors, and determining if a web application or database code contains SQL injection vulnerabilities.

To defend against SQL injection, developers must take proper care in configuring and developing their applications in order to make them robust and secure. Developers should use best practices and countermeasures to prevent their applications from becoming vulnerable to SQL injection attacks.

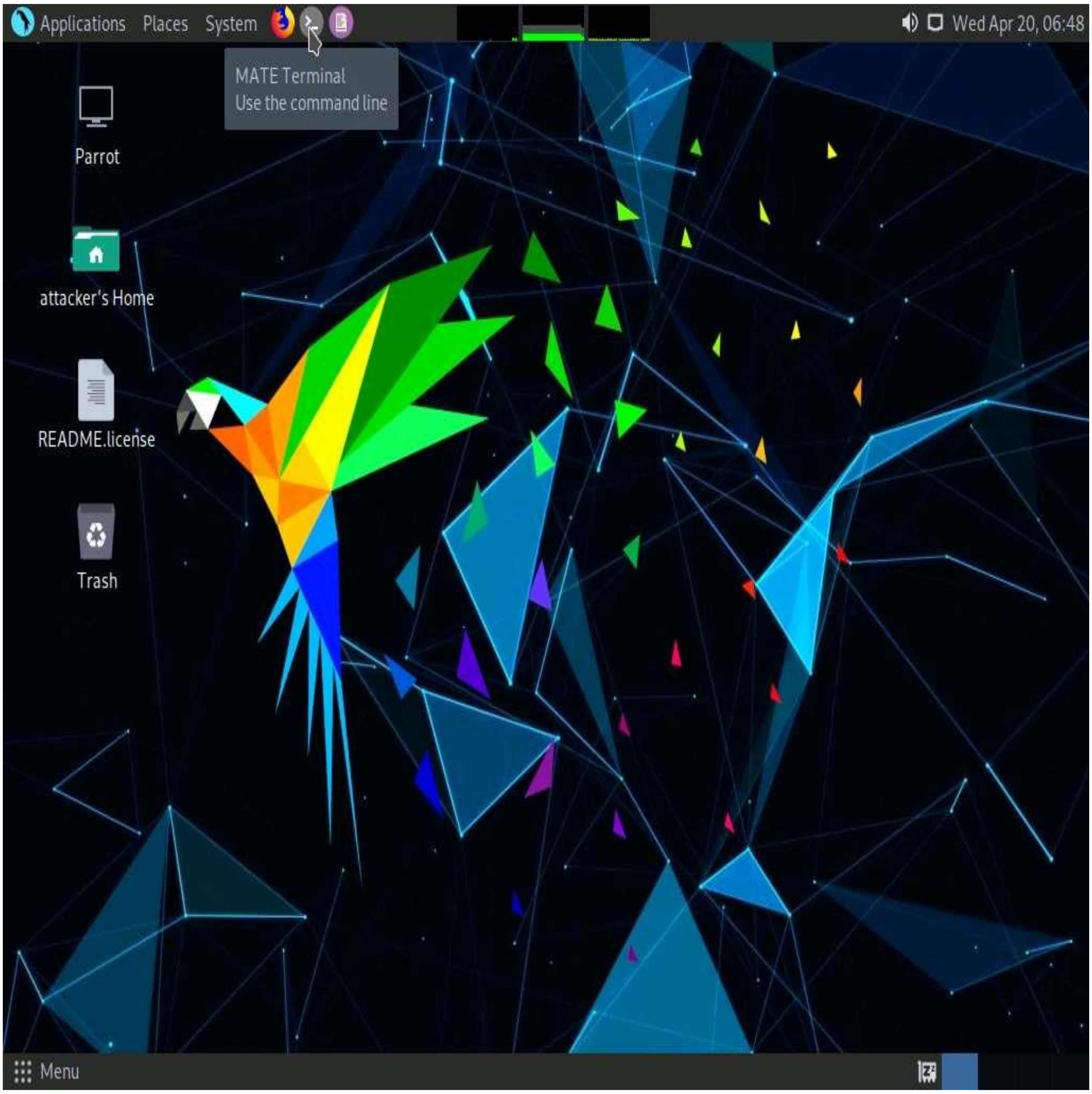
Task 1: Detect SQL Injection Vulnerabilities using DSSS

Damn Small SQLi Scanner (DSSS) is a fully functional SQL injection vulnerability scanner that supports GET and POST parameters. DSSS scans web applications for various SQL injection vulnerabilities.

Here, we will use DSSS to detect SQL injection vulnerabilities in a web application.

We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** machine.

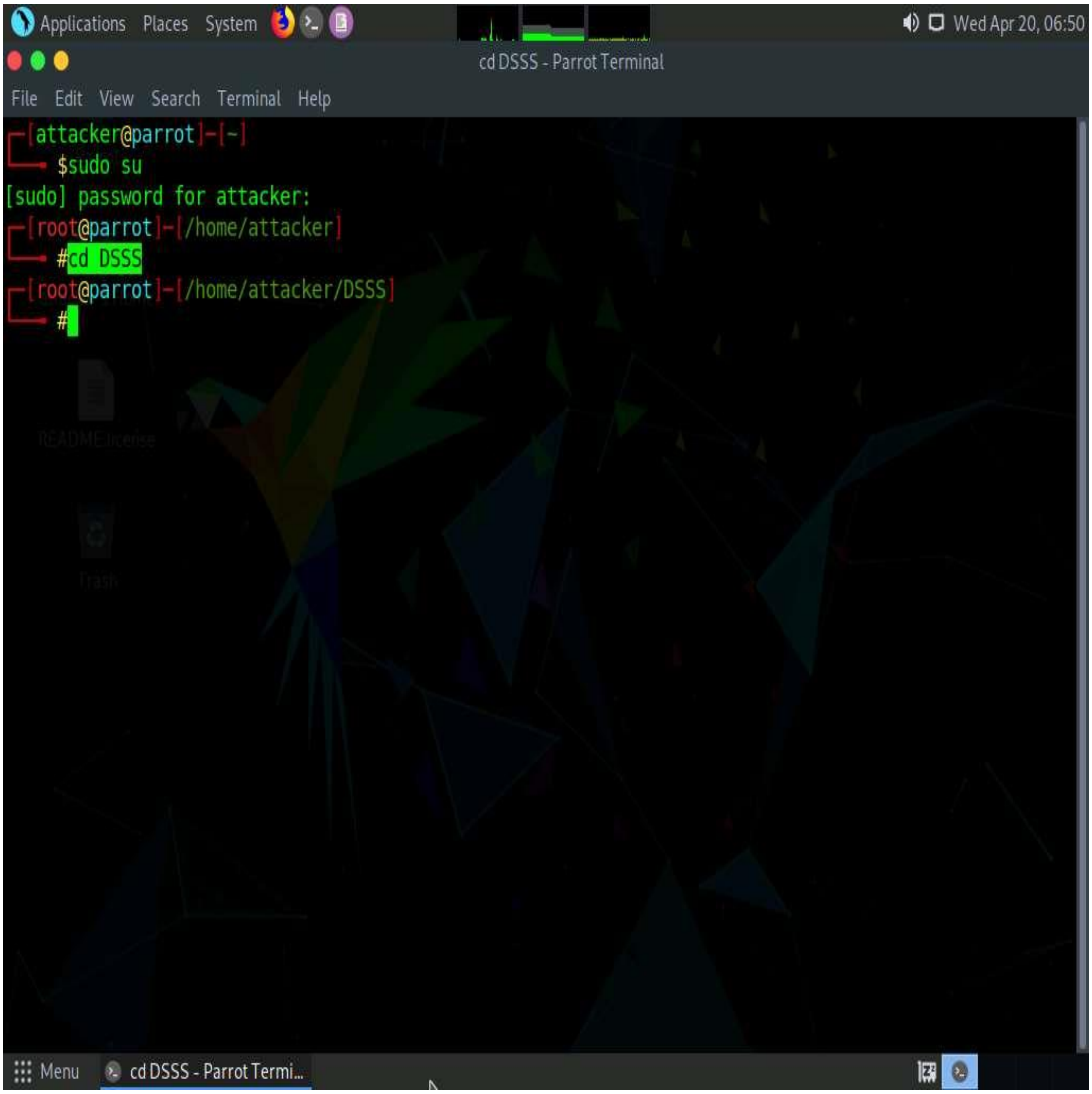
1. ☐ On the **Parrot Security** machine, click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Parrot Terminal** window.



2. ☐ A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. ☐ In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

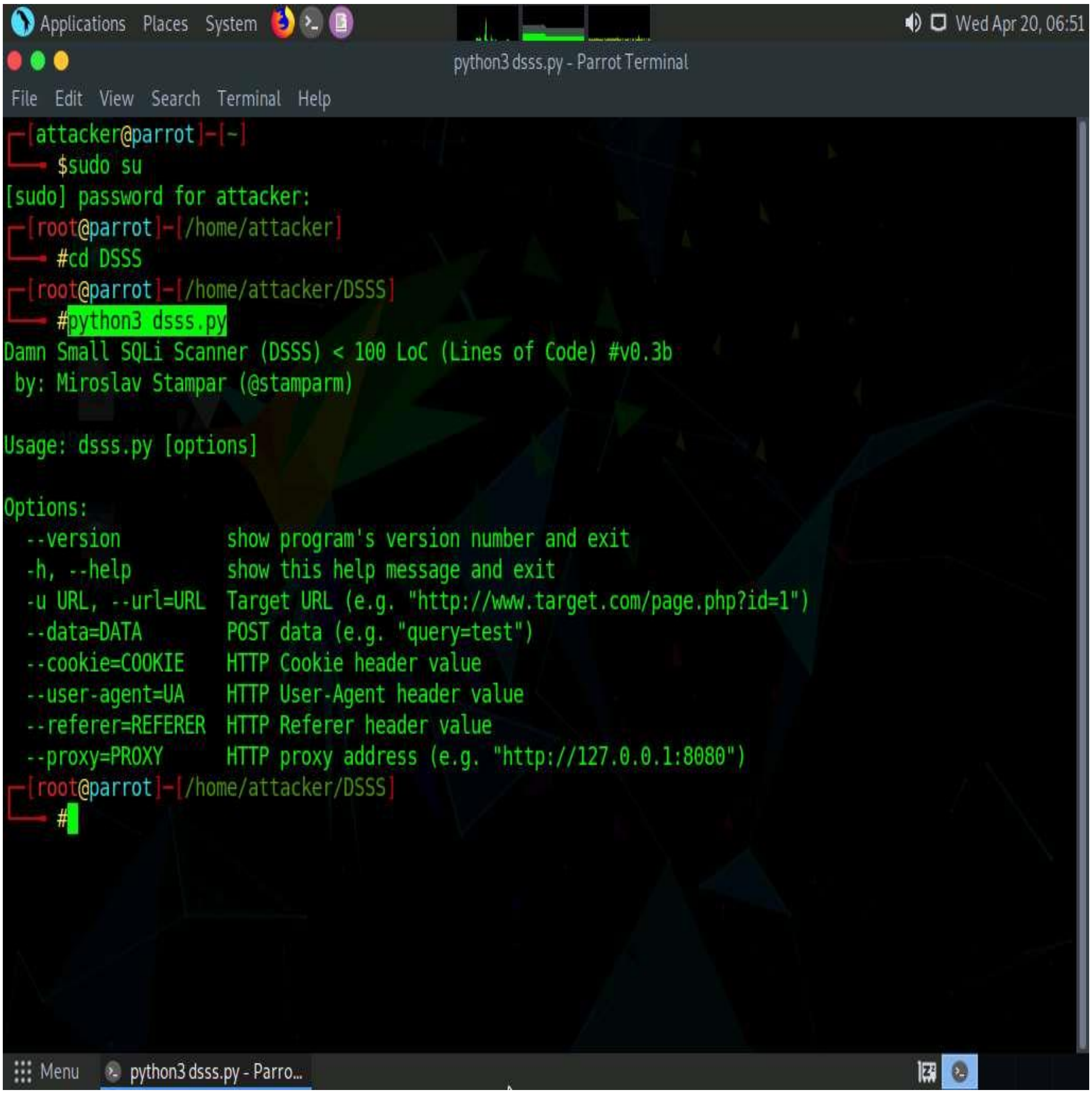
The password that you type will not be visible.

4. ☐ In the **MATE Terminal** type **cd DSSS** and press **Enter** to navigate to the DSSS folder which is already downloaded.



```
┌─[attacker@parrot]─[~]
└──╼ $sudo su
[sudo] password for attacker:
┌─[root@parrot]─[/home/attacker]
└──╼ #cd DSSS
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #
```

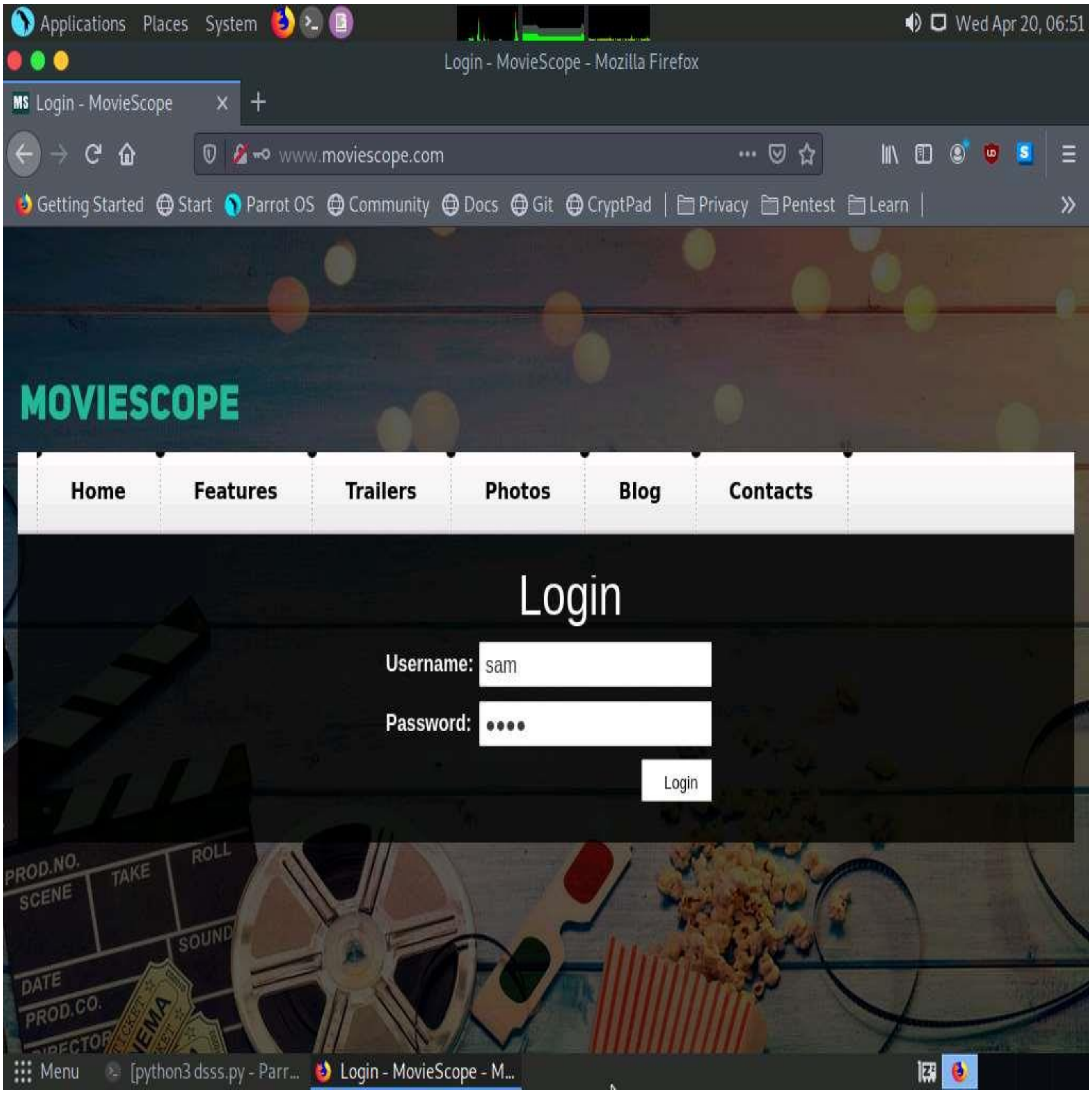
5. ☐ In the terminal window, type **python3 dsss.py** and press **Enter** to view a list of available options in the DSSS application, as shown in the screenshot.



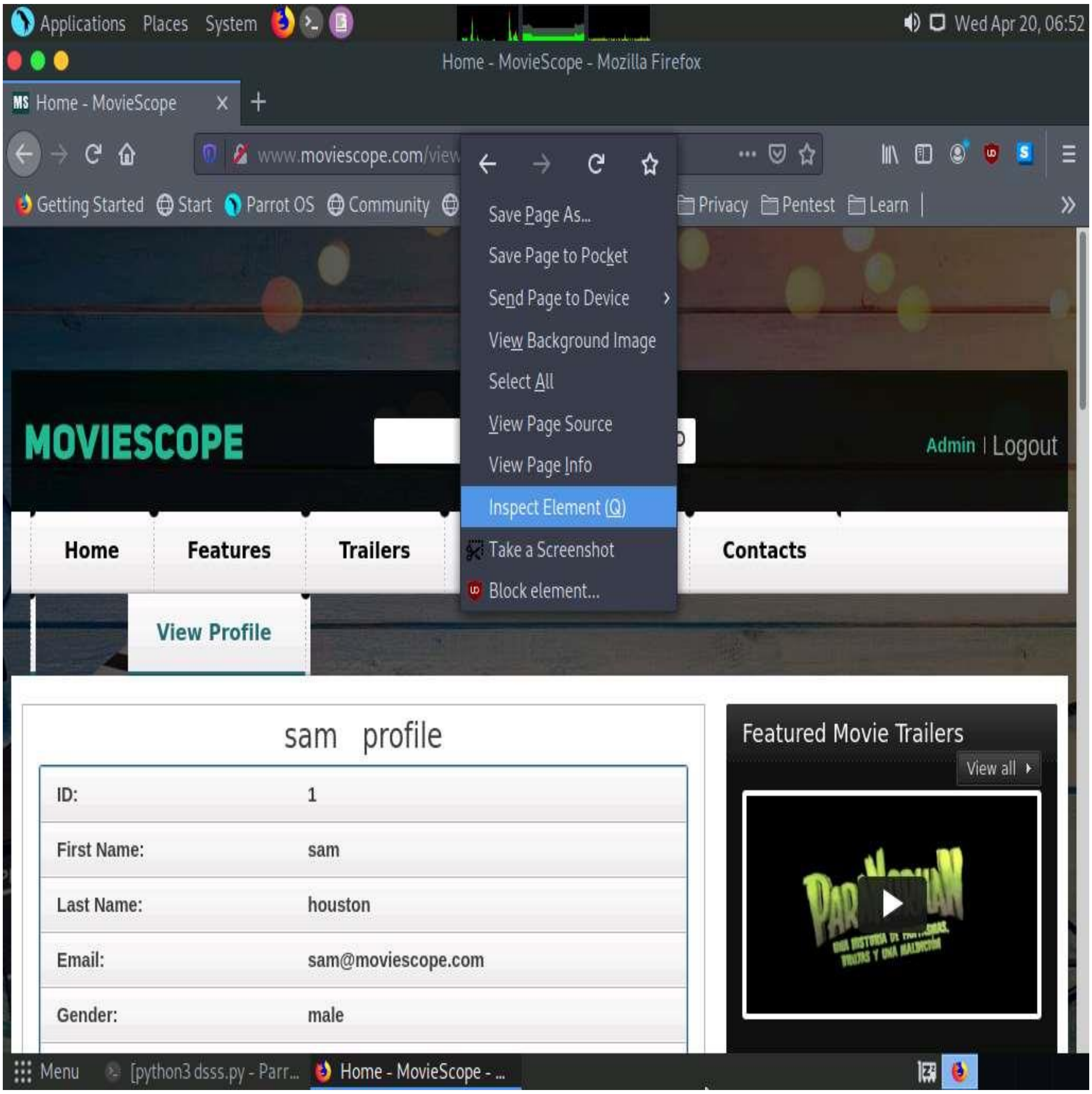
```
┌─[attacker@parrot]─[~]
└──╼ $sudo su
[sudo] password for attacker:
┌─[root@parrot]─[/home/attacker]
└──╼ #cd DSSS
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

Usage: dsss.py [options]

Options:
  --version          show program's version number and exit
  -h, --help         show this help message and exit
  -u URL, --url=URL  Target URL (e.g. "http://www.target.com/page.php?id=1")
  --data=DATA        POST data (e.g. "query=test")
  --cookie=COOKIE    HTTP Cookie header value
  --user-agent=UA    HTTP User-Agent header value
  --referer=REFERER  HTTP Referer header value
  --proxy=PROXY      HTTP proxy address (e.g. "http://127.0.0.1:8080")
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #
```

6. ☐ Now, minimize the **Terminal** window and click on the **Firefox** icon in the top section of **Desktop** to launch Firefox.
7. ☐ In the **Mozilla Firefox** window, type **http://www.moviescope.com/** in the address bar and press **Enter**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don’t Save**.


Applications Places System
Wed Apr 20, 06:51
Login - MovieScope - Mozilla Firefox
Login - MovieScope
www.moviescope.com
Getting Started Start Parrot OS Community Docs Git CryptPad Privacy Pentest Learn
MOVIESCOPE
Home Features Trailers Photos Blog Contacts
Login
Username: sam
Password: ••••
Login
Menu [python3 dsss.py - Parr... Login - MovieScope - M...

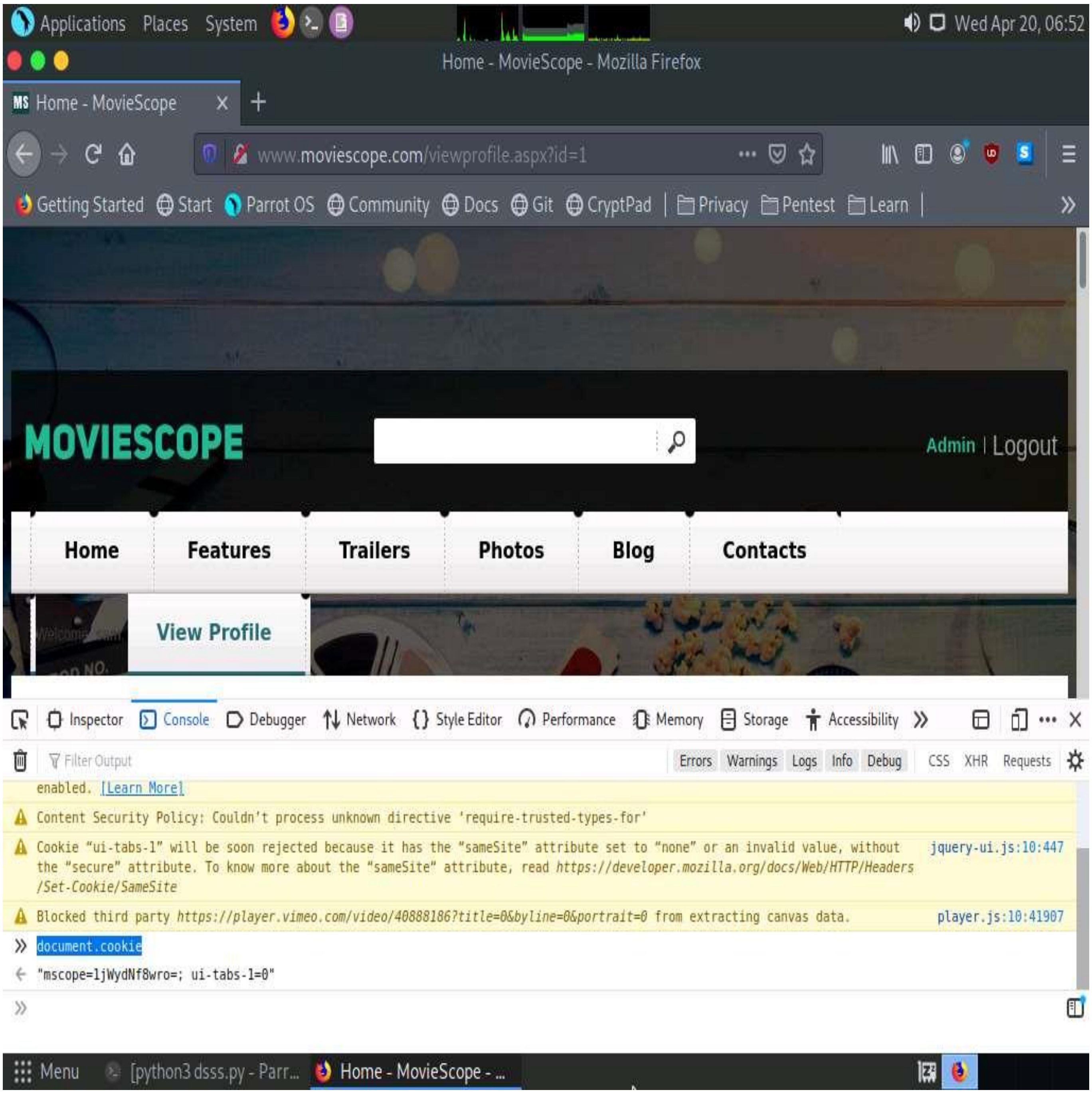
8. ☐ Once you are logged into the website, click the **View Profile** tab from the menu bar; and when the page has loaded, make a note of the URL in the address bar of the browser.
9. ☐ Right-click anywhere on the webpage and click **Inspect Element (Q)** from the context menu, as shown in the screenshot.



sam profile

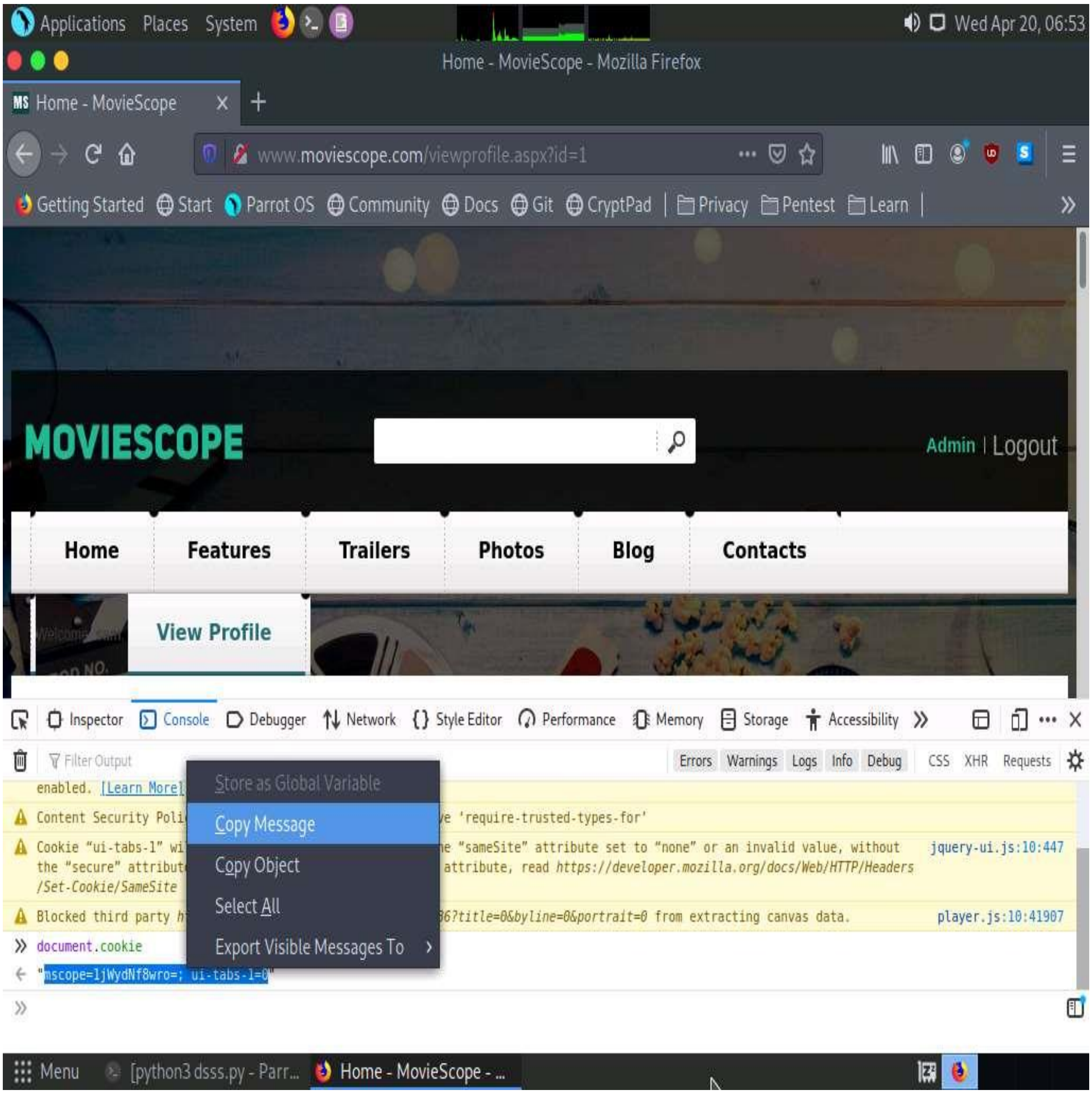
ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male

10. ☐ The **Developer Tools** frame appears in the lower section of the browser window. Click the **Console** tab, type **document.cookie** in the lower-left corner of the browser, and press **Enter**.


Applications Places System
Wed Apr 20, 06:52
Home - MovieScope - Mozilla Firefox
Home - MovieScope
www.moviescope.com/viewprofile.aspx?id=1
Getting Started Start Parrot OS Community Docs Git CryptPad Privacy Pentest Learn
MOVIESCOPE
Admin | Logout
Home Features Trailers Photos Blog Contacts
View Profile
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility
Filter Output
Errors Warnings Logs Info Debug CSS XHR Requests
Menu [python3 dsss.py - Parr... Home - MovieScope - ...

```
enabled. [Learn More]
Content Security Policy: Couldn't process unknown directive 'require-trusted-types-for'
Cookie "ui-tabs-1" will be soon rejected because it has the "sameSite" attribute set to "none" or an invalid value, without    jquery-ui.js:10:447
the "secure" attribute. To know more about the "sameSite" attribute, read https://developer.mozilla.org/docs/Web/HTTP/Headers
/Set-Cookie/SameSite
Blocked third party https://player.vimeo.com/video/40888186?title=0&byline=0&portrait=0 from extracting canvas data.    player.js:10:41907
» document.cookie
← "mscope=1jWydNf8wro=; ui-tabs-1=0"
»
```

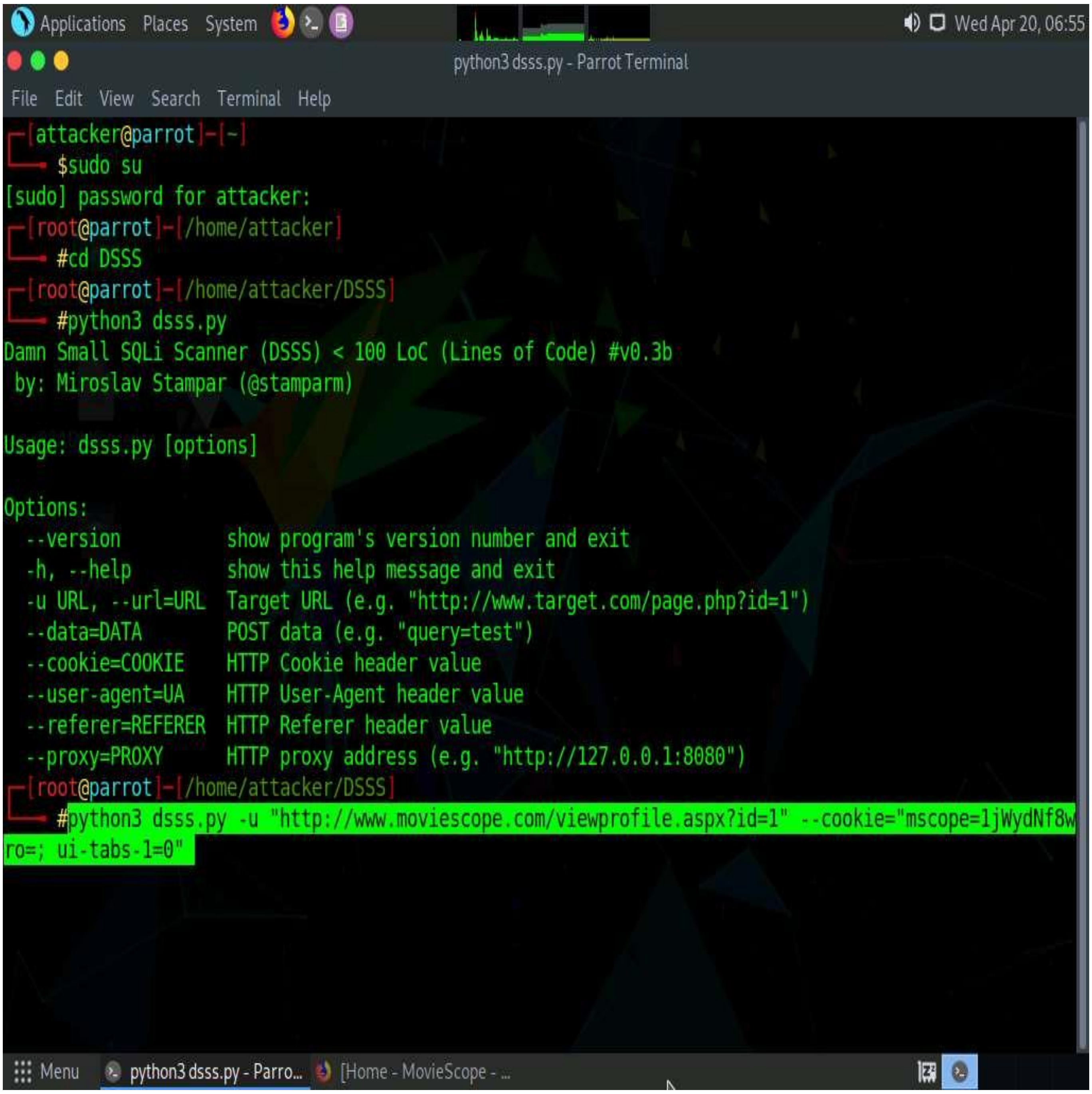
11. ☐ Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser.


Applications Places System
Wed Apr 20, 06:53
Home - MovieScope - Mozilla Firefox
Home - MovieScope
www.moviescope.com/viewprofile.aspx?id=1
Getting Started Start Parrot OS Community Docs Git CryptPad Privacy Pentest Learn
MOVIESCOPE
Admin | Logout
Home Features Trailers Photos Blog Contacts
View Profile
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility
Filter Output
Errors Warnings Logs Info Debug CSS XHR Requests
Store as Global Variable
Copy Message
Copy Object
Select All
Export Visible Messages To
Menu [python3 dsss.py - Parr... Home - MovieScope - ...

```
enabled. [Learn More]
Content Security Poli                      e 'require-trusted-types-for'
Cookie "ui-tabs-1" wi                      e "sameSite" attribute set to "none" or an invalid value, without    jquery-ui.js:10:447
the "secure" attribut                      attribute, read https://developer.mozilla.org/docs/Web/HTTP/Headers
/Set-Cookie/SameSite
Blocked third party h                      86?title=0&byline=0&portrait=0 from extracting canvas data.           player.js:10:41907
>> document.cookie
<- "mscope=1jWydNf8wro=; ui-tabs-1=0"
```

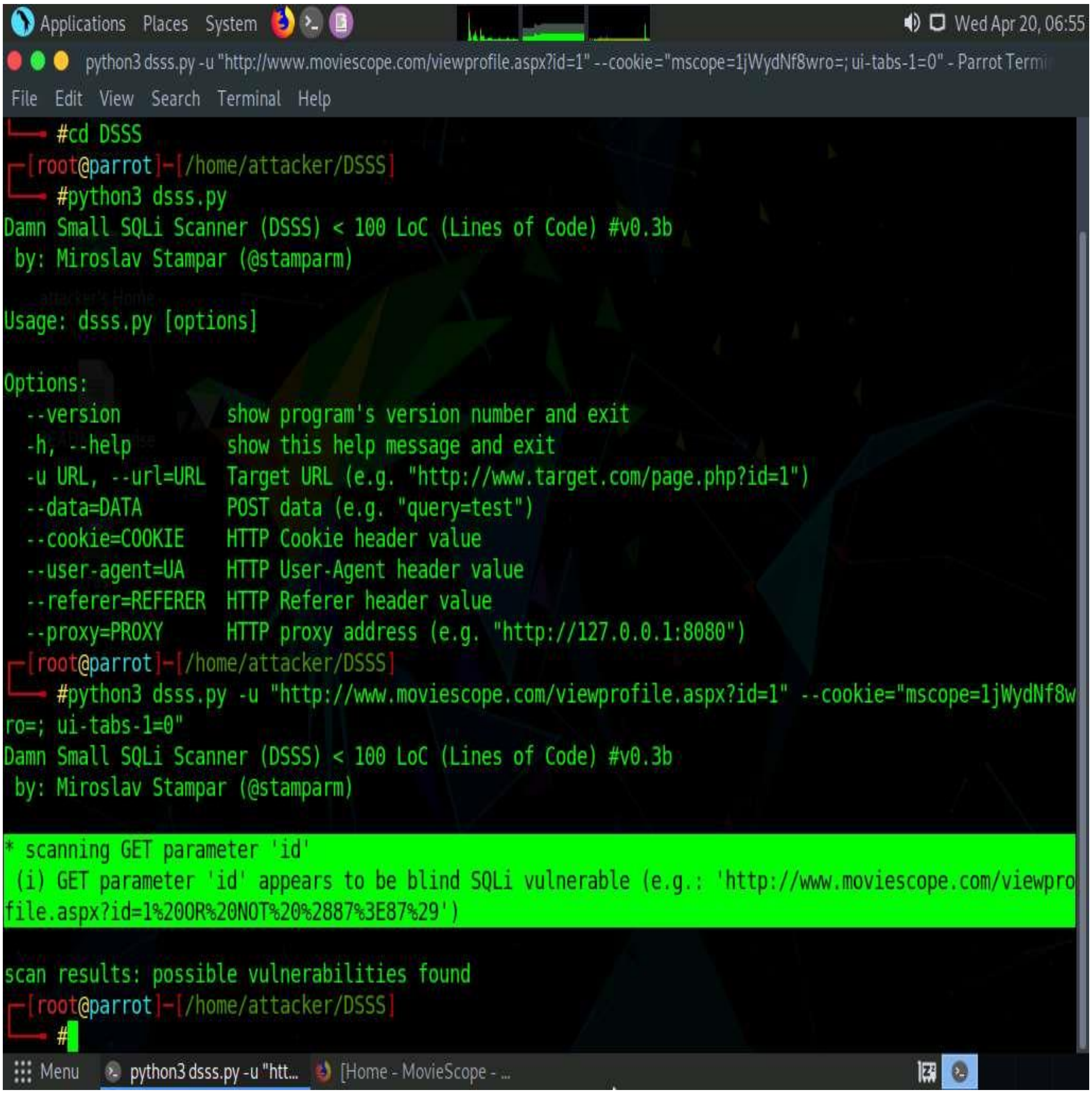
12. ☐ Switch to a terminal window and type **python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step 11]"** and press Enter.

In this command, **-u** specifies the target URL and **--cookie** specifies the HTTP cookie header value.


Applications Places System
Wed Apr 20, 06:55
python3 dsss.py - Parrot Terminal
File Edit View Search Terminal Help
Menu
python3 dsss.py - Parro...
[Home - MovieScope - ...

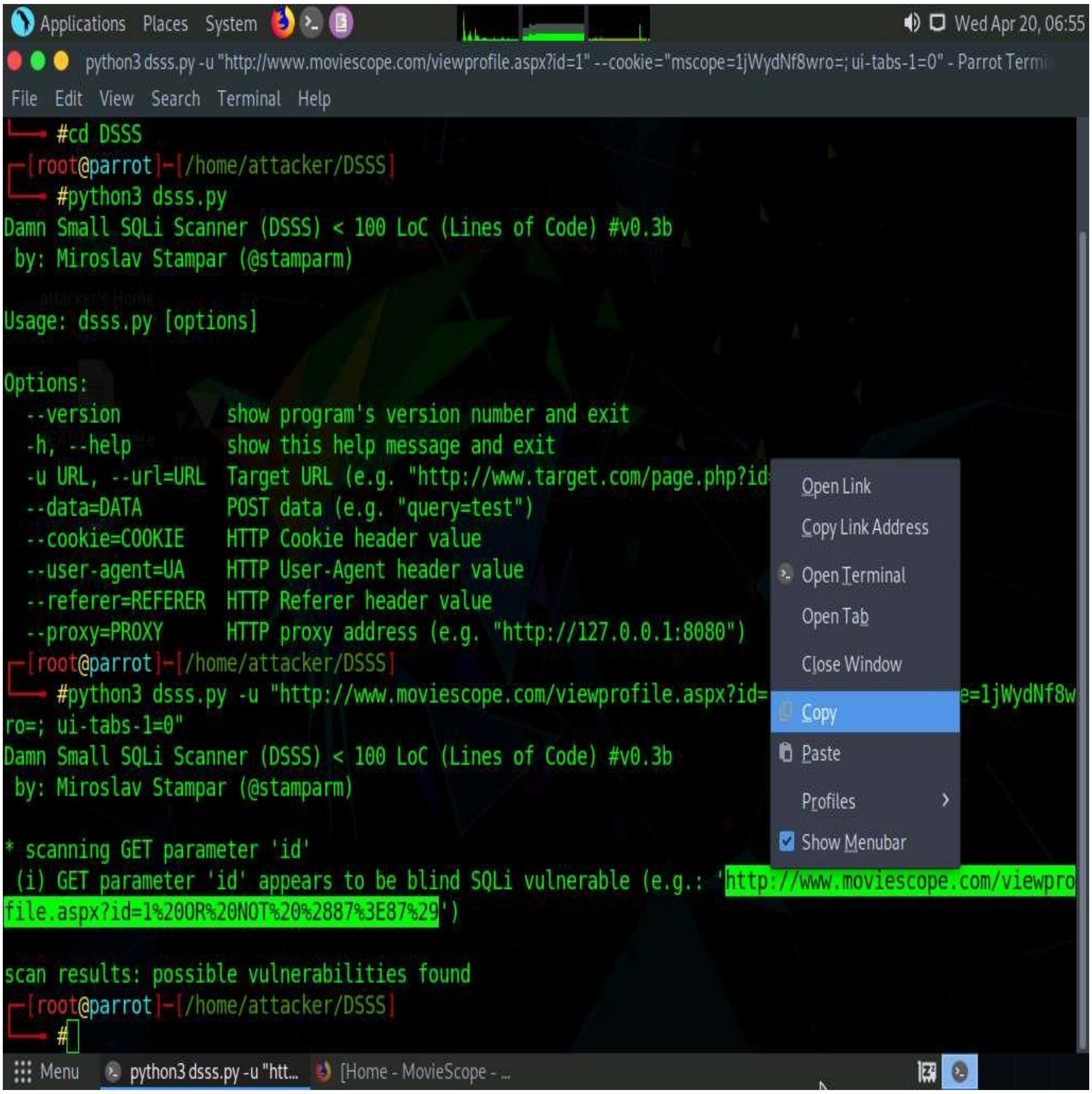
```
┌─[attacker@parrot]─[~]
└──╼ $sudo su
[sudo] password for attacker:
┌─[root@parrot]─[/home/attacker]
└──╼ #cd DSSS
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

Usage: dsss.py [options]

Options:
  --version          show program's version number and exit
  -h, --help         show this help message and exit
  -u URL, --url=URL  Target URL (e.g. "http://www.target.com/page.php?id=1")
  --data=DATA        POST data (e.g. "query=test")
  --cookie=COOKIE    HTTP Cookie header value
  --user-agent=UA    HTTP User-Agent header value
  --referer=REFERER  HTTP Referer header value
  --proxy=PROXY      HTTP proxy address (e.g. "http://127.0.0.1:8080")
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8w
ro=; ui-tabs-1=0"
```

13. ☐ The above command causes DSSS to scan the target website for SQL injection vulnerabilities.
14. ☐ The result appears, showing that the target website (**www.moviescope.com**) is vulnerable to blind SQL injection attacks. The vulnerable link is also displayed, as shown in the screenshot.


Applications Places System
Wed Apr 20, 06:55
python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" - Parrot Term
File Edit View Search Terminal Help
Menu
python3 dsss.py -u "htt...
[Home - MovieScope - ...

```
└──╼ #cd DSSS
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

Usage: dsss.py [options]

Options:
  --version           show program's version number and exit
  -h, --help          show this help message and exit
  -u URL, --url=URL   Target URL (e.g. "http://www.target.com/page.php?id=1")
  --data=DATA         POST data (e.g. "query=test")
  --cookie=COOKIE     HTTP Cookie header value
  --user-agent=UA     HTTP User-Agent header value
  --referer=REFERER   HTTP Referer header value
  --proxy=PROXY       HTTP proxy address (e.g. "http://127.0.0.1:8080")
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8w
ro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
 (i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.: 'http://www.moviescope.com/viewpro
file.aspx?id=1%20OR%20NOT%20%2887%3E87%29')

scan results: possible vulnerabilities found
┌─[root@parrot]─[/home/attacker/DSSS]
└──╼ #
```

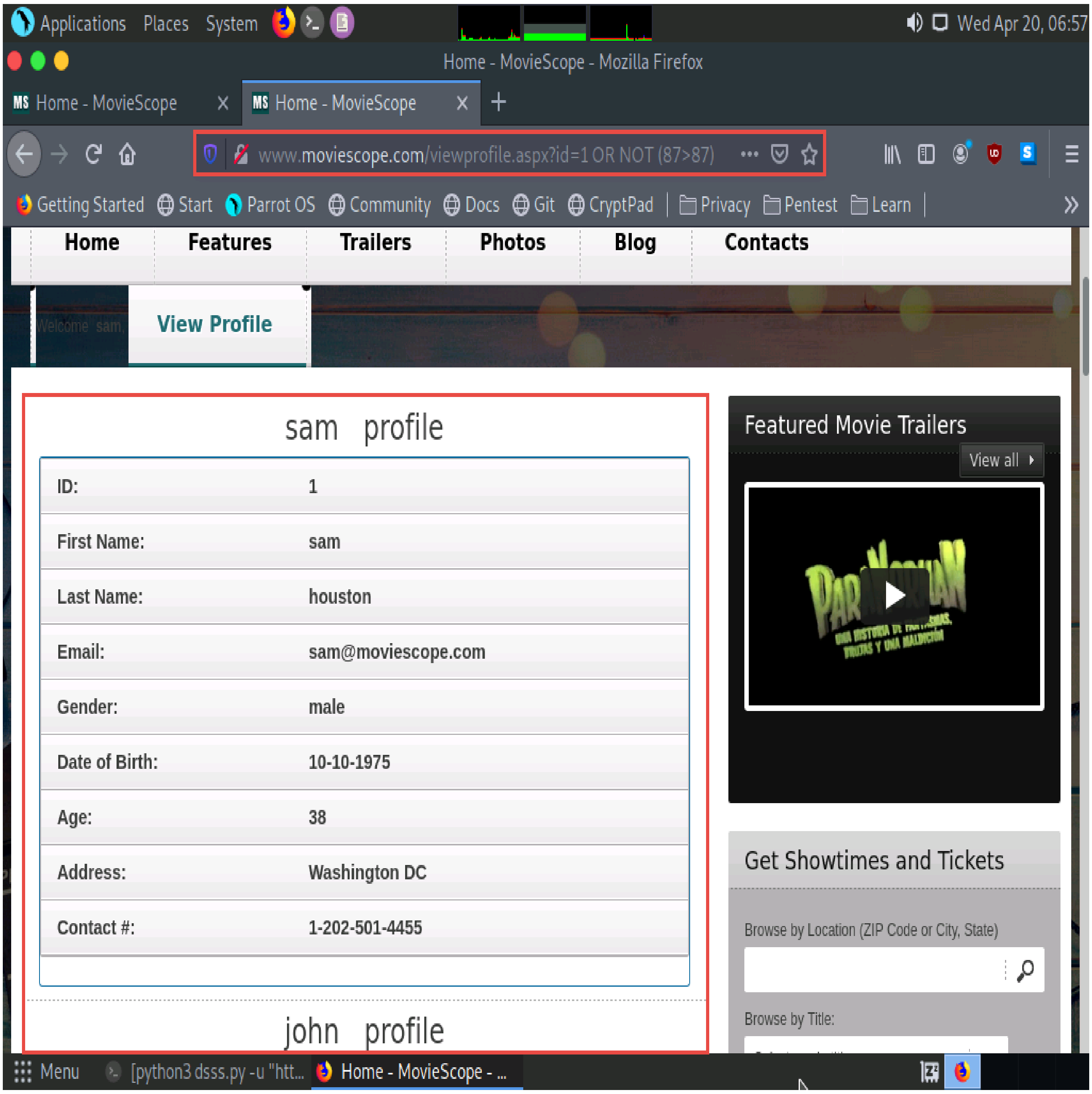
15. ☐ Highlight the vulnerable website link, right-click it, and, from the options, click **Copy**.



Applications Places System
Wed Apr 20, 06:55
python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" - Parrot Term
File Edit View Search Terminal Help
Open Link
Copy Link Address
Open Terminal
Open Tab
Close Window
Copy
Paste
Profiles
Show Menubar
Menu
python3 dsss.py -u "htt...
[Home - MovieScope - ...

```
└── #cd DSSS
┌─[root@parrot]─[/home/attacker/DSSS]
└── #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

Usage: dsss.py [options]

Options:
  --version          show program's version number and exit
  -h, --help         show this help message and exit
  -u URL, --url=URL  Target URL (e.g. "http://www.target.com/page.php?id
  --data=DATA        POST data (e.g. "query=test")
  --cookie=COOKIE    HTTP Cookie header value
  --user-agent=UA    HTTP User-Agent header value
  --referer=REFERER  HTTP Referer header value
  --proxy=PROXY      HTTP proxy address (e.g. "http://127.0.0.1:8080")
┌─[root@parrot]─[/home/attacker/DSSS]
└── #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=                e=1jWydNf8w
ro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
 by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
 (i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.: 'http://www.moviescope.com/viewpro
file.aspx?id=1%20OR%20NOT%20%2887%3E87%29')

scan results: possible vulnerabilities found
┌─[root@parrot]─[/home/attacker/DSSS]
└── #
```

16. ☐ Switch to **Mozilla Firefox**; in a new tab, paste the copied link in the address bar and press **Enter**.
17. ☐ You will observe that information regarding available user accounts appears under the **View Profile** tab.



Applications Places System Wed Apr 20, 06:57

Home - MovieScope - Mozilla Firefox

```
www.moviescope.com/viewprofile.aspx?id=1 OR NOT (87>87)
```

Home Features Trailers Photos Blog Contacts

Welcome sam View Profile

sam profile

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	1-202-501-4455

john profile

Featured Movie Trailers

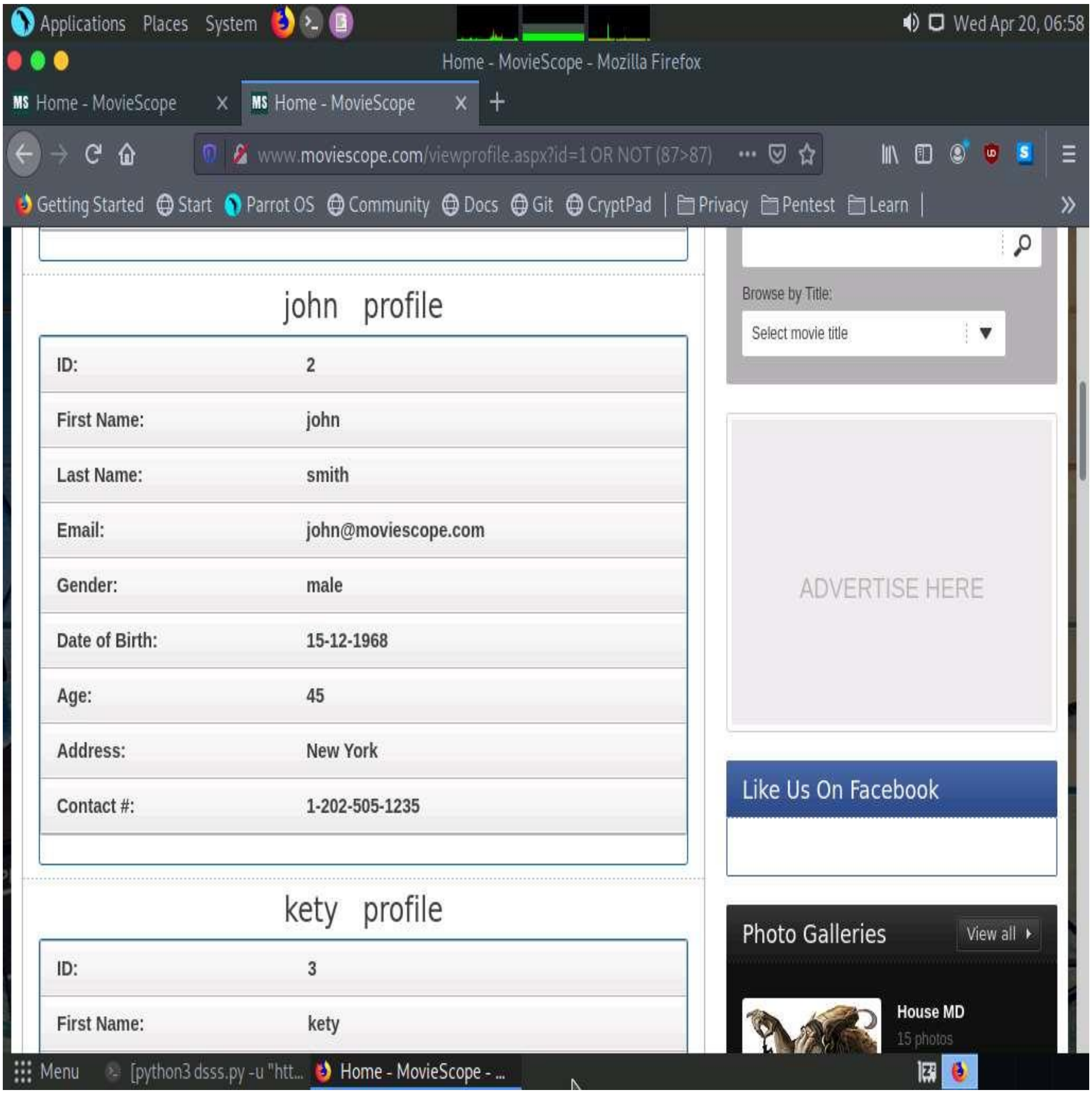
Get Showtimes and Tickets

Browse by Location (ZIP Code or City, State)

Browse by Title:

Menu [python3 dsss.py -u "htt... Home - MovieScope - ...

18. ☐ Scroll down to view the user account information for all users.



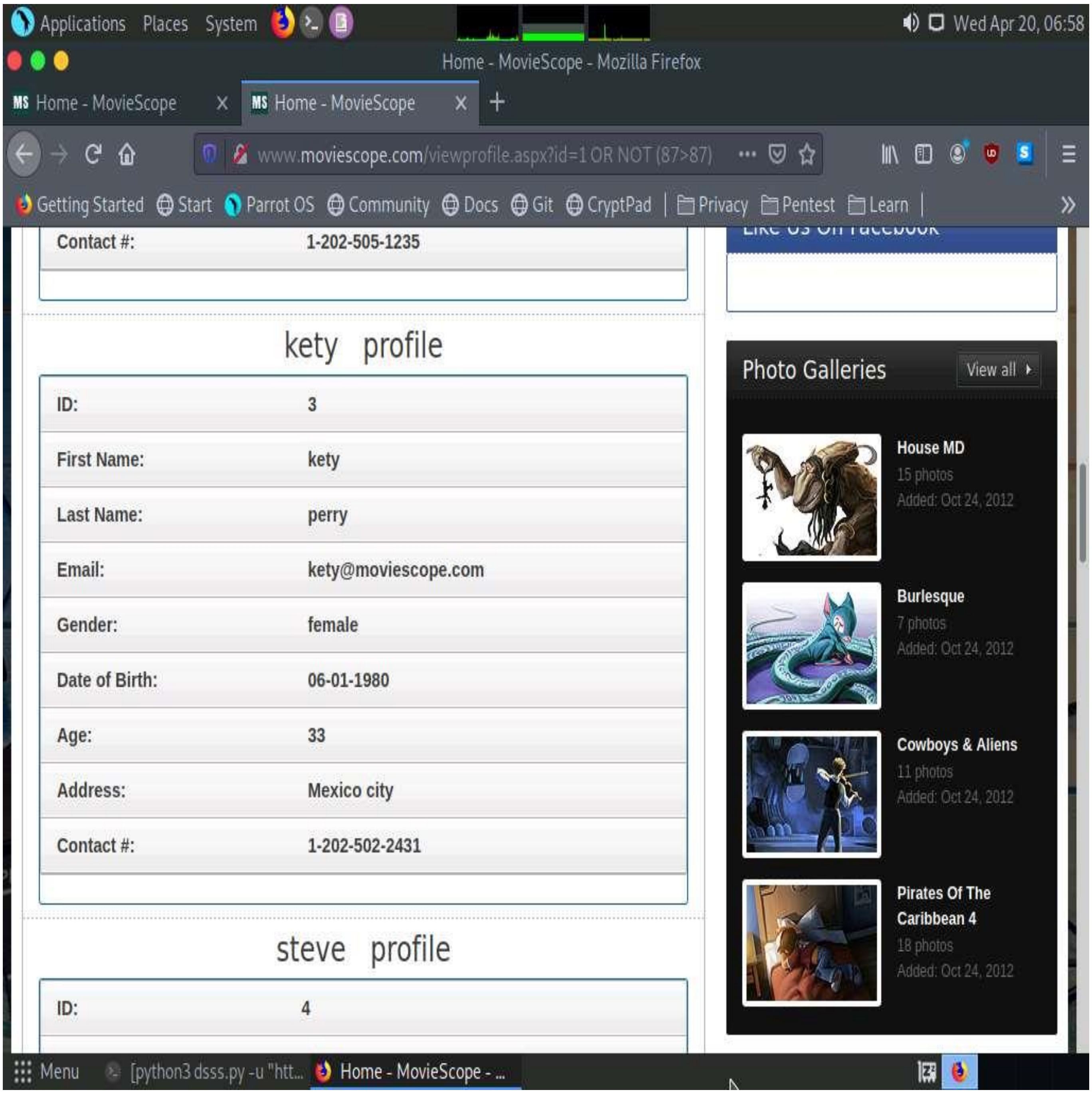
www.moviescope.com/viewprofile.aspx?id=1 OR NOT (87>87)

john profile

ID:	2
First Name:	john
Last Name:	smith
Email:	john@moviescope.com
Gender:	male
Date of Birth:	15-12-1968
Age:	45
Address:	New York
Contact #:	1-202-505-1235

kety profile

ID:	3
First Name:	kety



Home - MovieScope - Mozilla Firefox

www.moviescope.com/viewprofile.aspx?id=1 OR NOT (87>87)

Contact #:	1-202-505-1235

kety profile

ID:	3
First Name:	kety
Last Name:	perry
Email:	kety@moviescope.com
Gender:	female
Date of Birth:	06-01-1980
Age:	33
Address:	Mexico city
Contact #:	1-202-502-2431

steve profile

ID:	4

In real life, attackers use blind SQL injection to access or destroy sensitive data. Attackers can steal data by asking a series of true or false questions through SQL statements. The results of the injection are not visible to the attacker. This type of attack can become time-intensive, because the database must generate a new statement for each newly recovered bit.

19. ☐ This concludes the demonstration of how to detect SQL injection vulnerabilities using DSSS.
20. ☐ Close all open windows and document all the acquired information.

Task 2: Detect SQL Injection Vulnerabilities using OWASP ZAP

OWASP Zed Attack Proxy (ZAP) is an integrated penetration testing tool for finding vulnerabilities in web applications. It offers automated scanners and a set of tools that allow you to find security vulnerabilities manually. It is designed to be used by people with a wide range of security experience, and as such is ideal for developers and functional testers who are new to penetration testing.

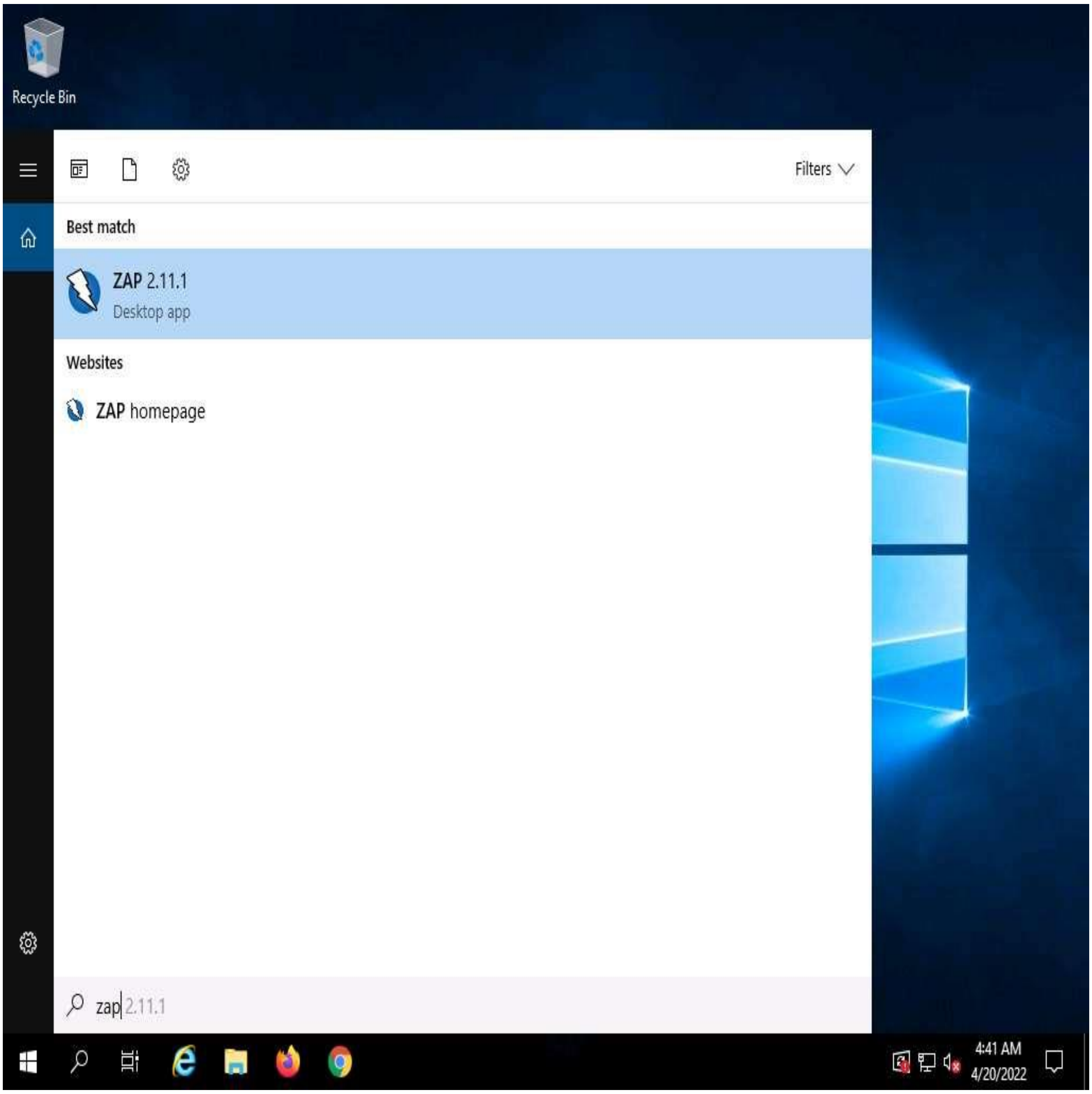
In this task, we will use OWASP ZAP to test a web application for SQL injection vulnerabilities.

We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** machine.

1. ☐ Click Windows Server 2019 to switch to the **Windows Server 2019** machine.

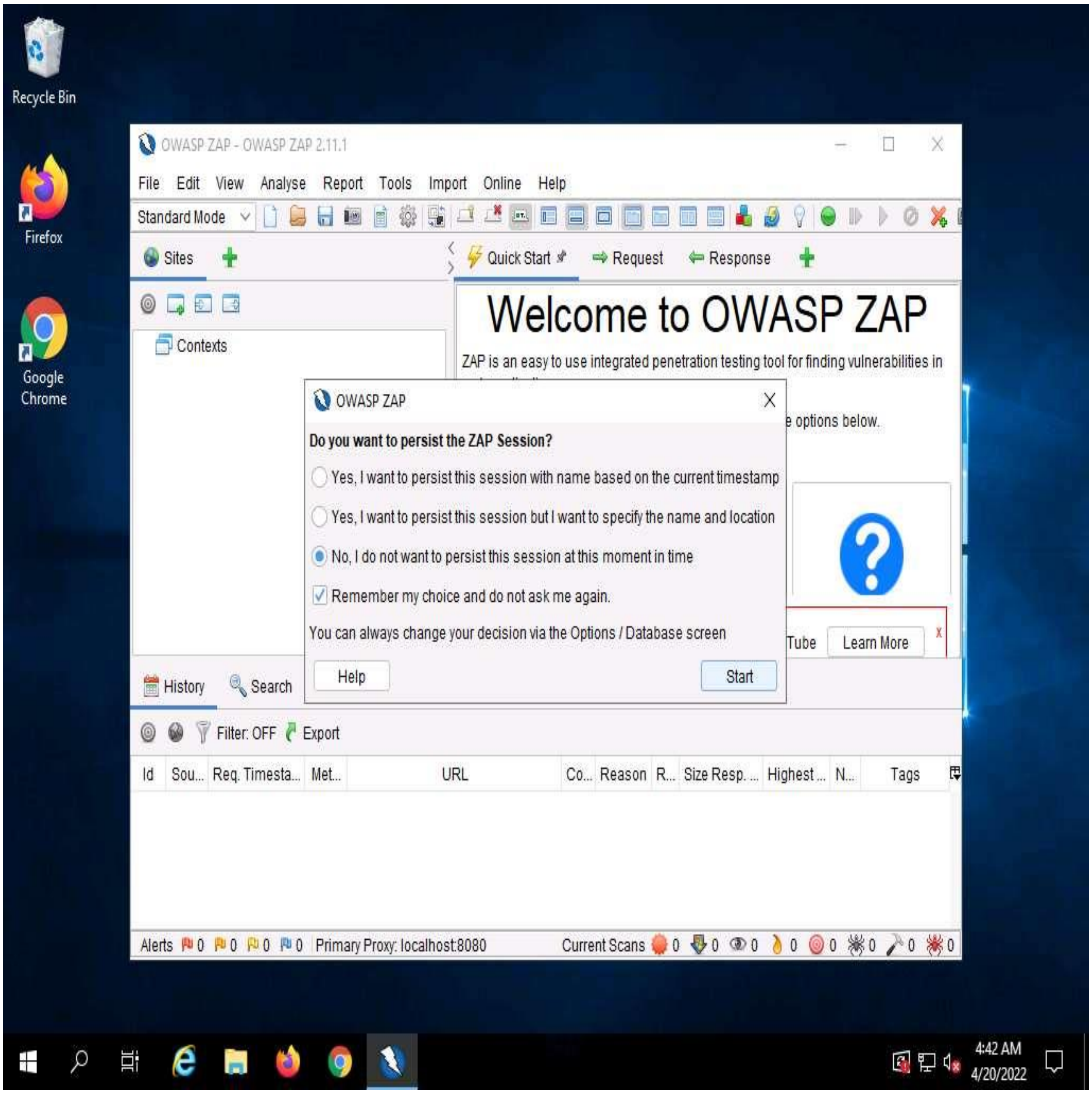
   If you are logged out of the **Windows Server 2019** machine, click Ctrl+Alt+Delete, then login into **Administrator** user profile using **Pa$$w0rd** as password.

2. ☐ Click **Type here to search** icon ( ) on the **Desktop**. Type **zap** in the search field, the **Zap 2.11.1** appears in the results, press **Enter** launch it.



3. ☐ OWASP ZAP initialized and a prompt that reads **Do you want to persist the ZAP Session?** appears; select the **No, I do not want to persist this session at this moment in time** radio button, and click **Start**.

If a **Manage Add-ons** window appears, close it.



Welcome to OWASP ZAP

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in

OWASP ZAP

Do you want to persist the ZAP Session?

- Yes, I want to persist this session with name based on the current timestamp
- Yes, I want to persist this session but I want to specify the name and location
- [✓ No, I do not want to persist this session at this moment in time]
- [✓] Remember my choice and do not ask me again.

You can always change your decision via the Options / Database screen

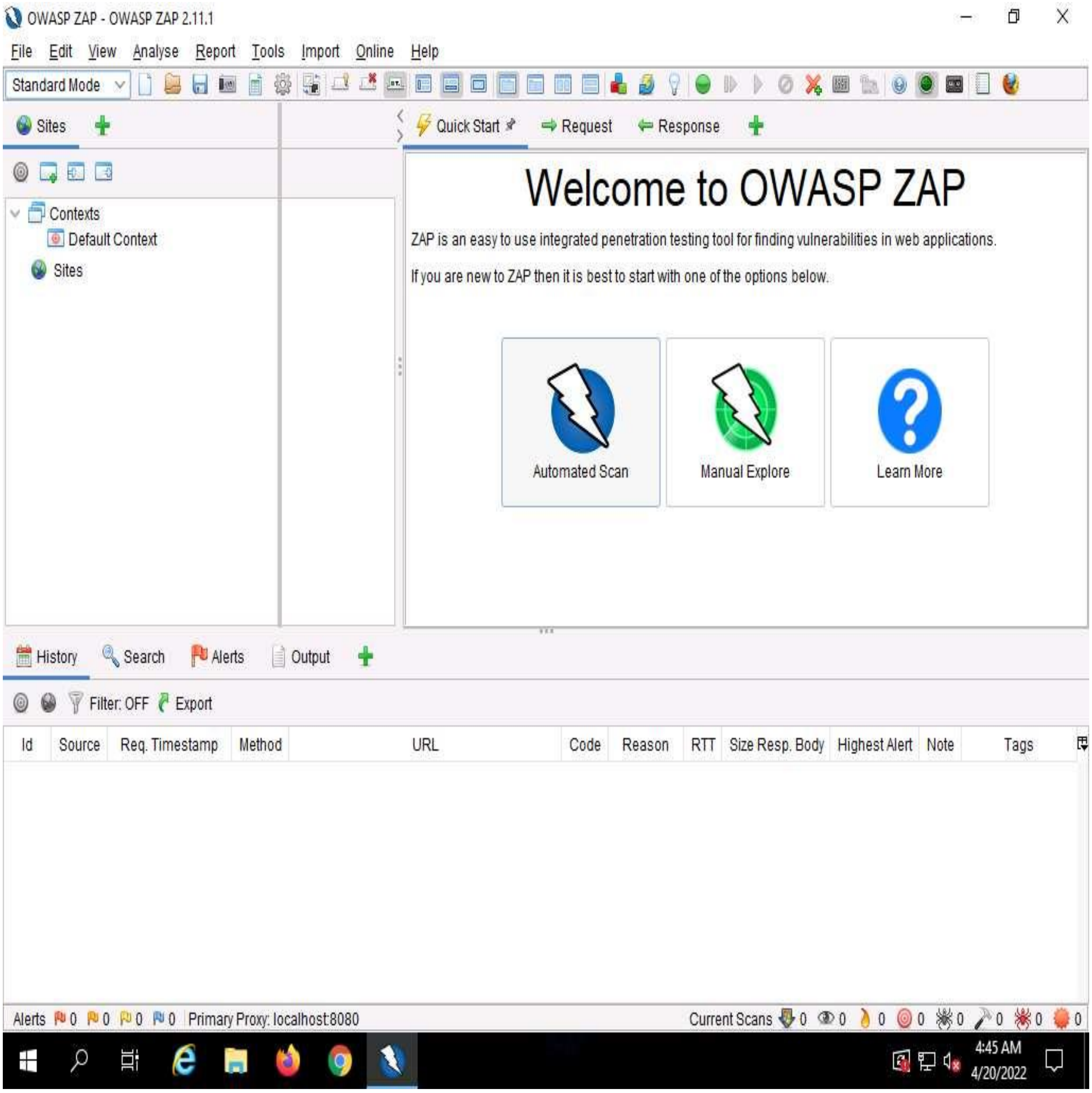
Help | Start

Id	Sou...	Req. Timesta...	Met...	URL	Co...	Reason	R...	Size Resp. ...	Highest ...	N...	Tags

Alerts 0 0 0 0 Primary Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

4. ☐ The **OWASP ZAP** main window appears; under the **Quick Start** tab, click the **Automated Scan** option.

If OWASP ZAP alert pop-up appears, click **OK** in all the pop-ups.


OWASP ZAP - OWASP ZAP 2.11.1
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
Quick Start
Request
Response
History
Search
Alerts
Output
Filter: OFF
Export
Id Source Req. Timestamp Method URL Code Reason RTT Size Resp. Body Highest Alert Note Tags
Alerts 0 0 0 0 Primary Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0
4:45 AM
4/20/2022

Welcome to OWASP ZAP

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

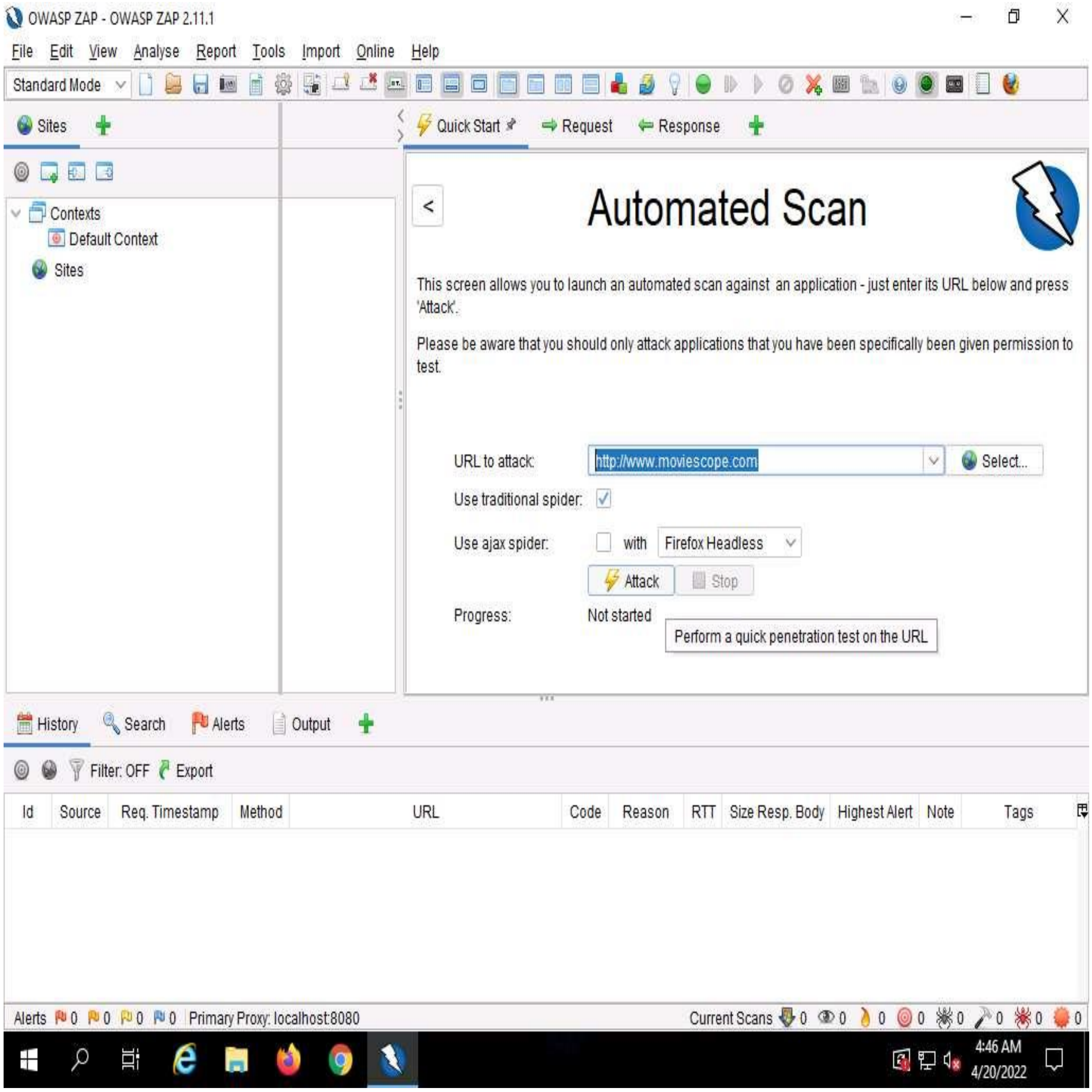
If you are new to ZAP then it is best to start with one of the options below.

Automated Scan

Manual Explore

Learn More

5. ☐ The **Automated Scan** wizard appears, enter the target website in the **URL to attack** field (in this case, **http://www.moviescope.com**). Leave other options set to default, and then click the **Attack** button.


OWASP ZAP - OWASP ZAP 2.11.1
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
Sites

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Use traditional spider: ✓

Use ajax spider: ☐ with Firefox Headless

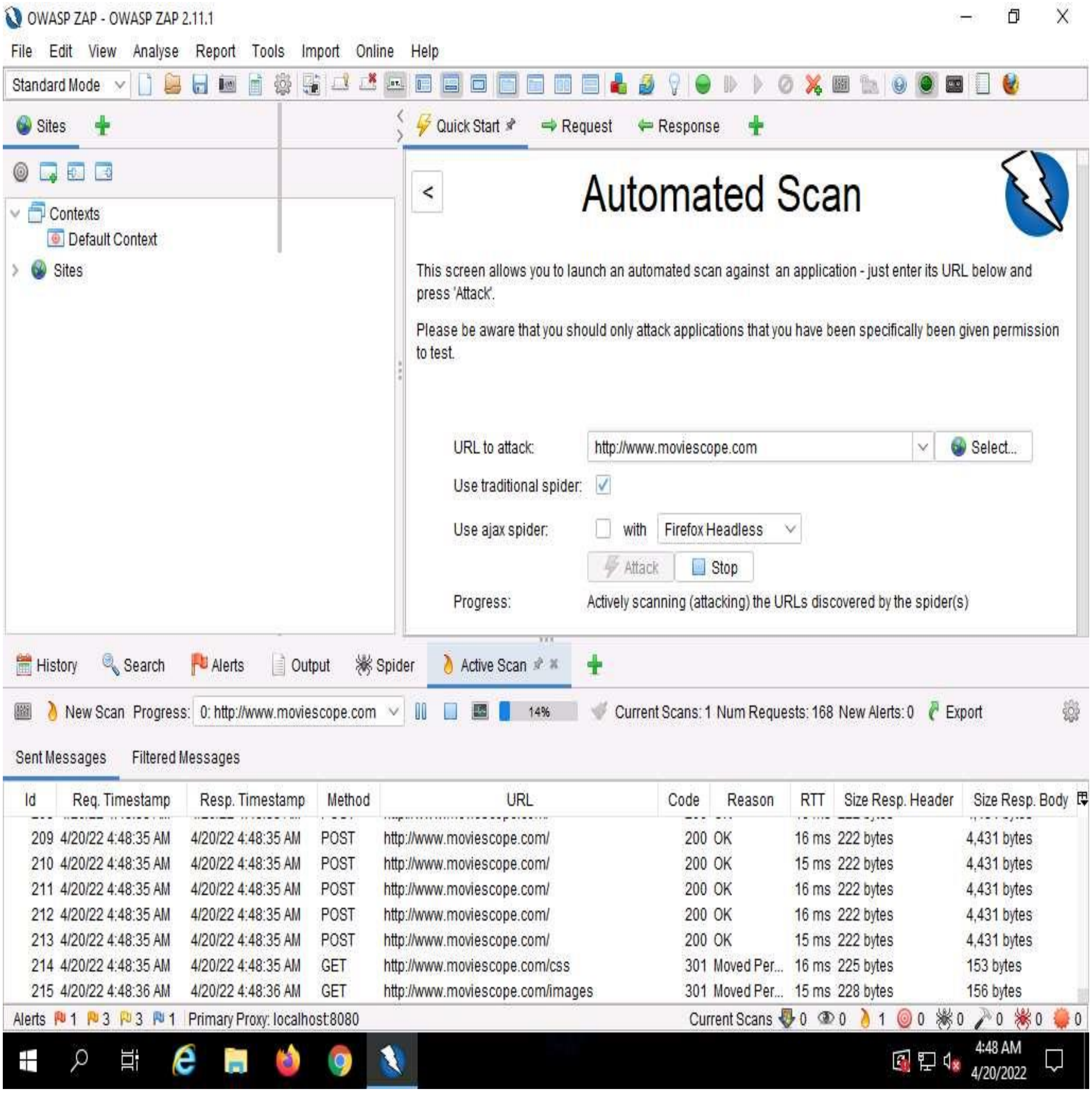
Progress: Not started

Perform a quick penetration test on the URL

Id	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags

Alerts 0 0 0 0 Primary Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

6. ☐ **OWASP ZAP** starts performing **Active Scan** on the target website, as shown in the screenshot.


OWASP ZAP - OWASP ZAP 2.11.1

File Edit View Analyse Report Tools Import Online Help

Standard Mode

Sites

Contexts
Default Context
Sites

Quick Start | Request | Response

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Use traditional spider: ✓

Use ajax spider: ☐ with Firefox Headless

Attack | Stop

Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

History | Search | Alerts | Output | Spider | Active Scan

New Scan Progress: 0: http://www.moviescope.com 14% Current Scans: 1 Num Requests: 168 New Alerts: 0 Export

Sent Messages | Filtered Messages

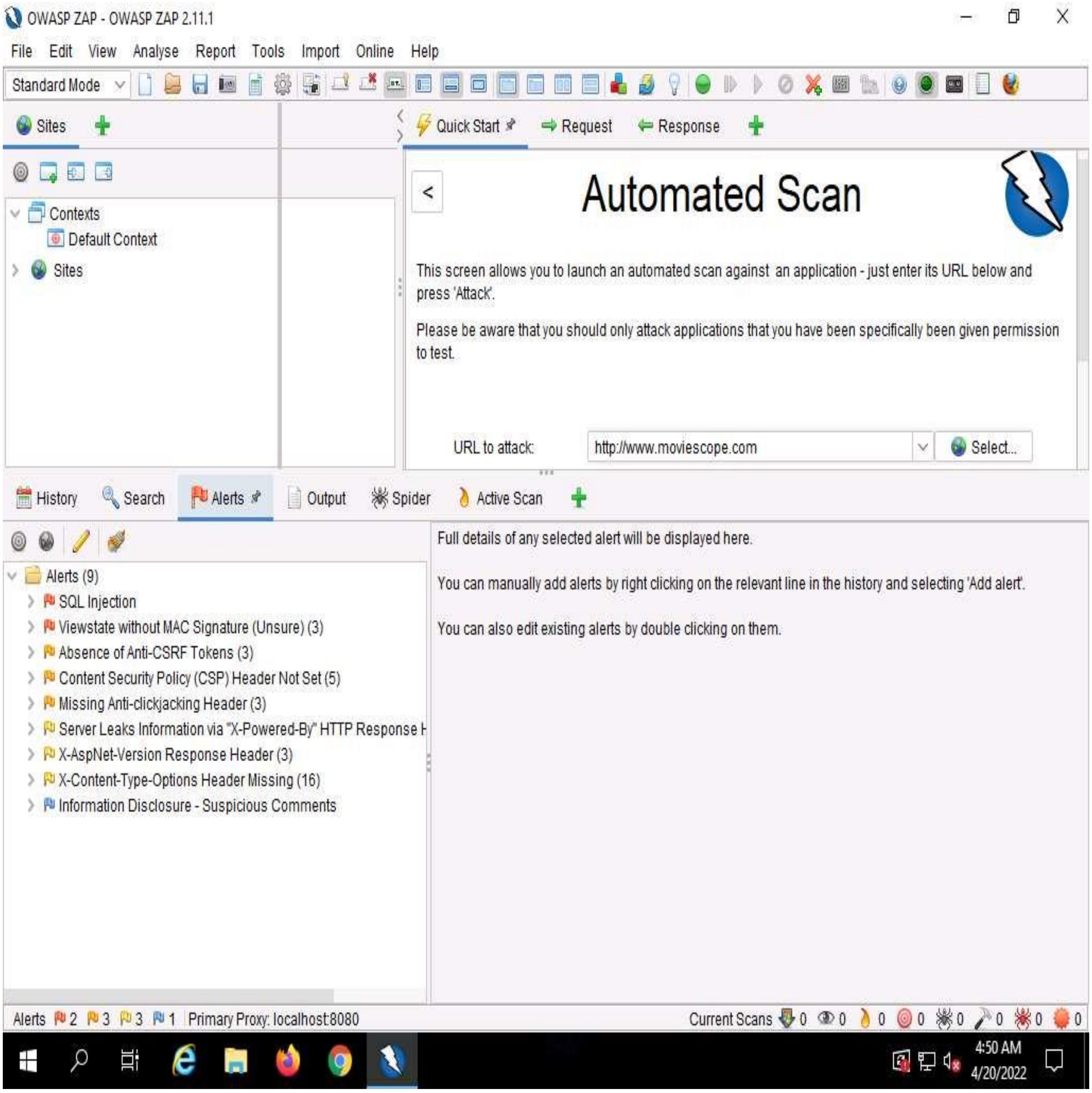
Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
209	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
210	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	15 ms	222 bytes	4,431 bytes
211	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
212	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
213	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	15 ms	222 bytes	4,431 bytes
214	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	GET	http://www.moviescope.com/css	301	Moved Per...	16 ms	225 bytes	153 bytes
215	4/20/22 4:48:36 AM	4/20/22 4:48:36 AM	GET	http://www.moviescope.com/images	301	Moved Per...	15 ms	228 bytes	156 bytes

Alerts 1 3 3 1 Primary Proxy: localhost:8080 Current Scans 0 0 1 0 0 0 0 0

4:48 AM 4/20/2022

7. ☐ After the scan completes, **Alerts** tab appears, as shown in the screenshot.
8. ☐ You can observe the vulnerabilities found on the website under the **Alerts** tab.

The discovered vulnerabilities might differ when you perform this task.


OWASP ZAP - OWASP ZAP 2.11.1
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
Quick Start
Request
Response
History Search Alerts Output Spider Active Scan
URL to attack:
Select...
Current Scans
4:50 AM
4/20/2022

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

Alerts (9)
- SQL Injection
- Viewstate without MAC Signature (Unsure) (3)
- Absence of Anti-CSRF Tokens (3)
- Content Security Policy (CSP) Header Not Set (5)
- Missing Anti-clickjacking Header (3)
- Server Leaks Information via "X-Powered-By" HTTP Response H
- X-AspNet-Version Response Header (3)
- X-Content-Type-Options Header Missing (16)
- Information Disclosure - Suspicious Comments

Full details of any selected alert will be displayed here.

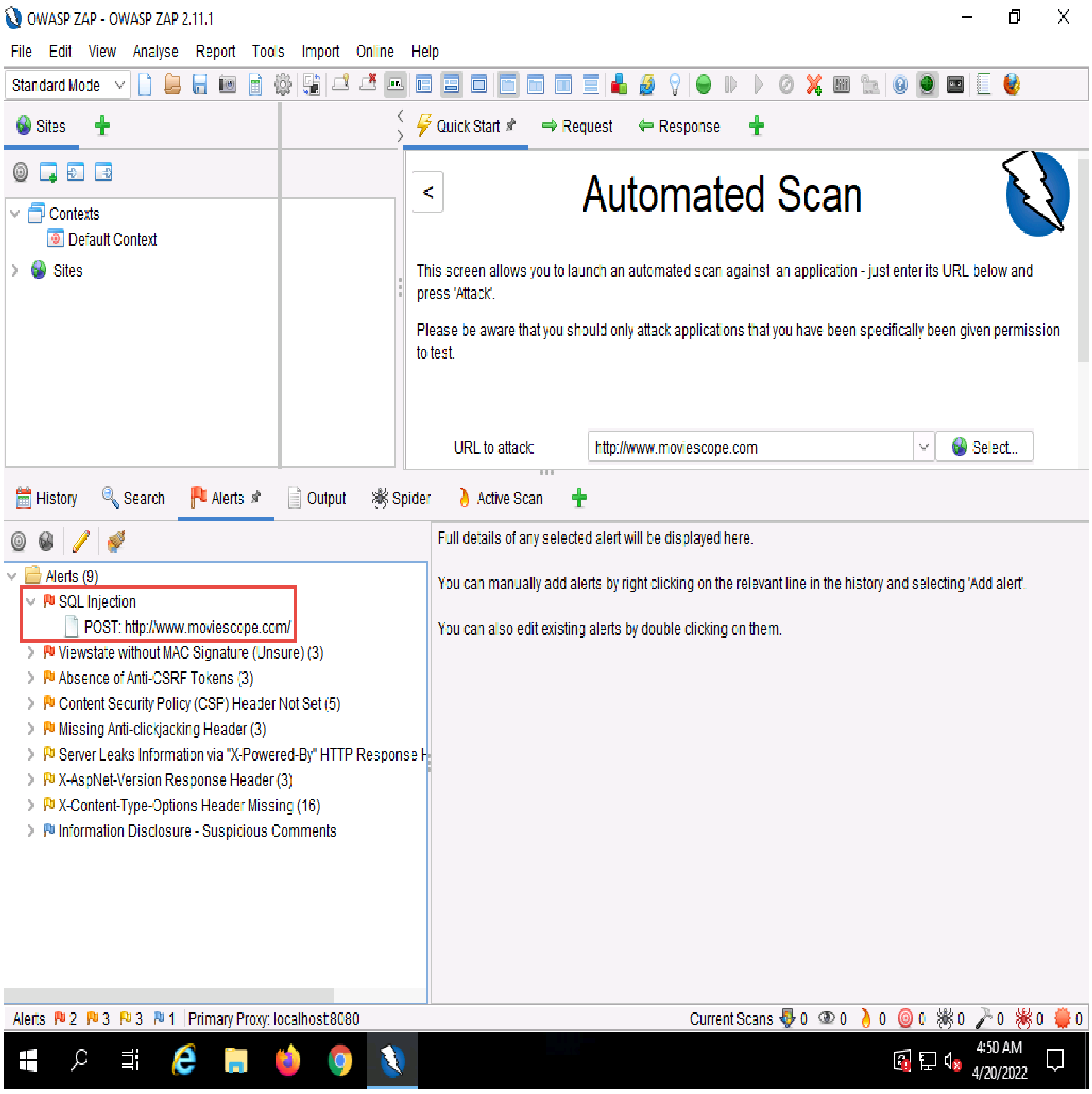
You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 2 3 3 1 Primary Proxy: localhost:8080

9. ☐ Now, expand the **SQL Injection** vulnerability node under the **Alerts** tab.

If you do not see SQL Injection vulnerability under the Alerts tab, perform


OWASP ZAP - OWASP ZAP 2.11.1

File Edit View Analyse Report Tools Import Online Help

Standard Mode

Sites

Contexts
Default Context
Sites

Quick Start Request Response

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://www.moviescope.com

History Search Alerts Output Spider Active Scan

Alerts (9)
- SQL Injection
  - POST: http://www.moviescope.com/
- Viewstate without MAC Signature (Unsure) (3)
- Absence of Anti-CSRF Tokens (3)
- Content Security Policy (CSP) Header Not Set (5)
- Missing Anti-clickjacking Header (3)
- Server Leaks Information via "X-Powered-By" HTTP Response H
- X-AspNet-Version Response Header (3)
- X-Content-Type-Options Header Missing (16)
- Information Disclosure - Suspicious Comments

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 2 3 3 1 Primary Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

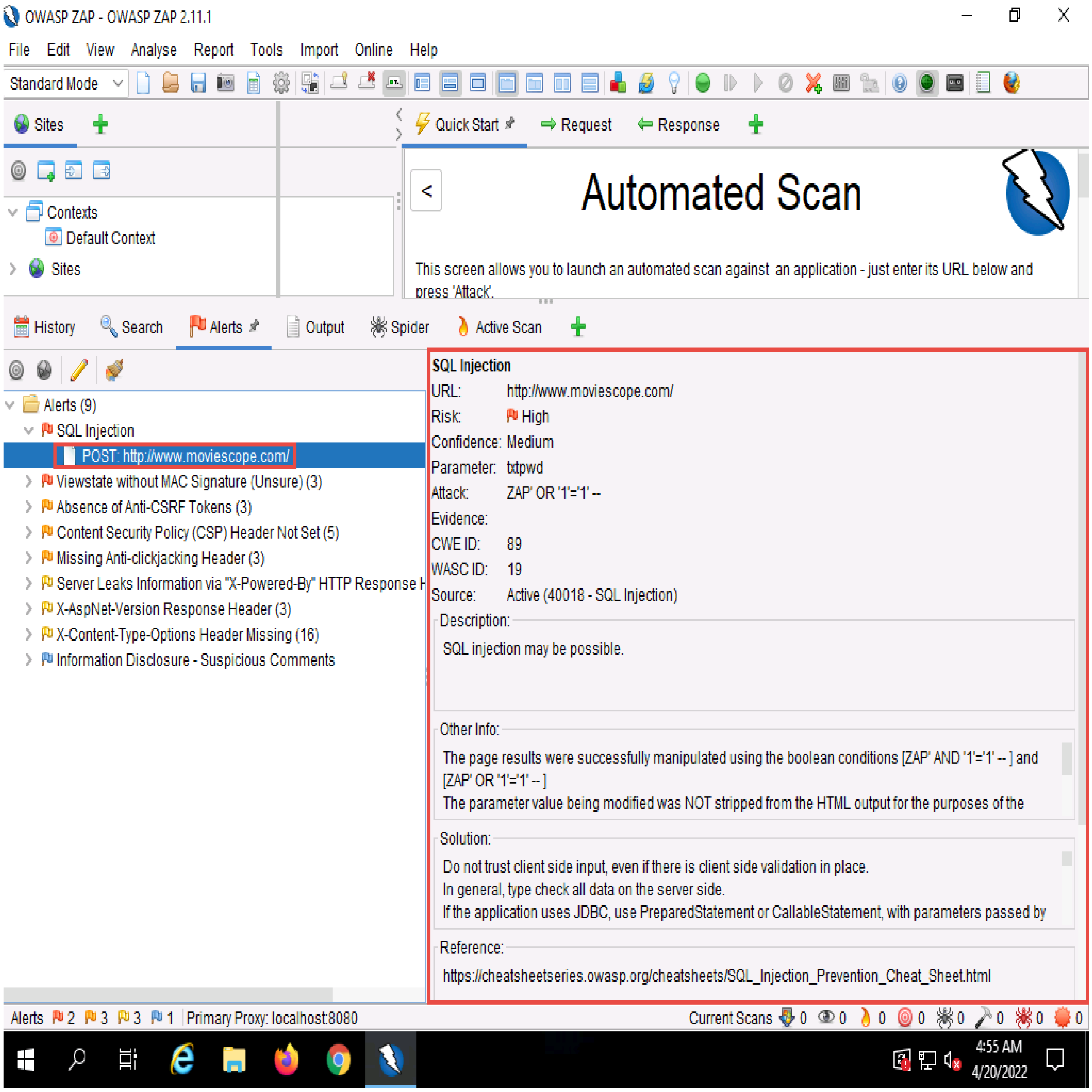
4:50 AM 4/20/2022

10. ☐ Click on the discovered **SQL Injection** vulnerability and further click on the vulnerable URL.

11. ☐ You can observe the information such as **Risk**, **Confidence**, **Parameter**, **Attack**, etc., regarding the discovered SQL Injection vulnerability in the lower right-bottom, as shown in the screenshot.

The risks associated with the vulnerability are categorized according to severity of risk as Low, Medium, High, and Informational alerts. Each level of risk is represented by a different flag color:

- **Red Flag**: High risk
- **Orange Flag**: Medium risk
- **Yellow Flag**: Low risk
- **Blue Flag**: Provides details about information disclosure vulnerabilities


OWASP ZAP - OWASP ZAP 2.11.1
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
Quick Start
Request
Response
Automated Scan
History
Search
Alerts
Output
Spider
Active Scan
Alerts (9)
SQL Injection
POST: http://www.moviescope.com/
Viewstate without MAC Signature (Unsure) (3)
Absence of Anti-CSRF Tokens (3)
Content Security Policy (CSP) Header Not Set (5)
Missing Anti-clickjacking Header (3)
Server Leaks Information via "X-Powered-By" HTTP Response H
X-AspNet-Version Response Header (3)
X-Content-Type-Options Header Missing (16)
Information Disclosure - Suspicious Comments
Alerts 2 3 3 1 Primary Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0
4:55 AM
4/20/2022

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

SQL Injection

URL:	http://www.moviescope.com/
Risk:	High
Confidence:	Medium
Parameter:	txtpwd
Attack:	ZAP' OR '1'='1' --
Evidence:	
CWE ID:	89
WASC ID:	19
Source:	Active (40018 - SQL Injection)

Description:

SQL injection may be possible.

Other Info:

The page results were successfully manipulated using the boolean conditions [ZAP' AND '1'='1' --] and [ZAP' OR '1'='1' --]
The parameter value being modified was NOT stripped from the HTML output for the purposes of the

Solution:

Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

12. ☐ This concludes the demonstration of how to detect SQL injection vulnerabilities using OWASP ZAP.

13. ☐ Close all open windows and document all the acquired information.

14. ☐ You can also use other SQL injection detection tools such as **Acunetix Web Vulnerability Scanner** (https://www.acunetix.com), **Snort** (https://snort.org), **Burp Suite** (https://www.portswigger.net), **w3af** (https://w3af.org), to detect SQL injection vulnerabilities.