Module **app_lpd**

Trabalho de LPD @author:João Estevão

Functions

def active_connections()

This function aims to determine the active connections on the machine. It uses the command "netstat -a" to get information about the active connections and stores this information in a variable called "netstat". It then breaks this information into lines and prints them to the screen.

Parameters

netstat : netstat -a list

Returns

active connections on the machine

def authentication(user, passwd)

Authenticates a user using argon2 as the password-hashing function.

Parameters

user : string
The user that is trying to authenticate.

passwd : string
The password inputted.

Returns

is_authenticated : boolean
Returns True if the user/password combo is in the db, otherwise returns False.

def chat()

The chat() function is responsible for implementing the client side of a messaging application. It allows the user to choose a server to send an RSA-encrypted message to and also read all messages sent to the server. The function presents a number of options for the user, such as sending an encrypted message, reading messages from the server or exiting the application. When choosing option 1, the user is prompted for the server's IP address and port, and then the function checks if there are already RSA keys generated to encrypt the message. If not, the user is prompted to generate a new key. The message is then encrypted and sent to the server over a socket connection. Option 2 allows the user to read the messages sent to the server by again entering the server's IP address and port. The function uses the sqlite3 library to connect to the database and retrieve the user ID. The function also uses the socket library to establish the connection to the server and the RSA library to encrypt and decrypt the messages.

Parameters

ip : string

port : int

choice : string

times : int

threads : int

Returns

udp or tcp flood() atack

def flood()

The "udpflood" function is a denial-of-service (DoS) attack script that sends UDP or TCP packets to a specific host on the specified on the specified port. The script prompts the user for information such as the IP address of the host, the port, whether to send UDP or TCP packets, and the number of packets to be sent over a single connection. It also allows the user to specify the number of threads they want to use to send the packets. It does this by creating multiple threads with the "Thread" method of the "threading" library and then starts these threads with the "start()" method. The script also has a "new" function that allows the user to start the attack again with the same settings without having to rerun the whole script.

Parameters

ip : string

port : int

choice : string

times : int

threads : int

Returns

udp or tcp flood() atack

def generate_pdf(filename, ip_country, title)

The above function is used to generate a PDF file based on three arguments: the filename a list of tuples with IP, country and timestamp information, and a title. The first step is to print a message that the report is being created. Next a SimpleDocTemplate object is created with the file name and page size. A "Flowable" object container is created and a title is added to the report using the Paragraph object and object and a specific font style. Next, a table is created from the list of tuples passed as argument and added to the element container. Finally, the document is built and saved with the specified name. The function also prints a message that the PDF has been successfully saved.

def log_manager()

The function log_manager() is responsible for analyzing the logs provided by the teacher and creating a heatmap for each log. It presents a menu interface for the user to choose between analyzing the UFW or SSH logs. If the UFW option is chosen, the function opens the corresponding log file, extracts all IP addresses and stores them in a list. It also uses the geoip2 library to get the geographic coordinates of these IP addresses and print out the countries from which these IPs were blocked. It then generates and saves a heatmap with these coordinates and creates a PDF file with information about the blocked IPs. If the SSH option is is chosen, the function runs a command to create a file with all logs of failed login attempts, extracts the IP addresses and times of these events and stores them in a list. It also uses the geoip2 library to get the geographic coordinates of these IP addresses and generates and saves a heatmap with these coordinates and creates a PDF file with information about the failed login attempts. failures. The function also has an option to exit the menu. Parameters

---

Choose a option to read log files

Returns

`generate_pdf() with with all logs of failed login() attempts, extracts the IP addresses and times of these events and stores them in a list anf geo location`

def login()

Parameters

user : string

passwd : string
The password inputted.

Returns

is_authenticated : boolean
Returns True and the user name.

def port_scanner()

This function aims to perform a port scan on a specific target. It uses Python's socket module to connect to different ports on the target and check if they are open or closed. It starts by asking the user to enter the target to be scanned and gets the IP address of that target. Then it defines a function called "portscan" that tries to connect to a specific port on the target and, if the connection is successful, prints out the open port. If the connection is not successful, the function passes. It also defines a function called "threader" that creates multiple threads to perform the port scan in parallel, which speeds up the process. It uses a queue to store the ports to be scanned and measures the time spent on the scan.

```
 Parameters
-------
target : string

Returns
-------
open ports in target and Time taken
```

Index

Functions