ITIS 6167/8167: Network Security

Project 2, DDoS attack implementation

Due Date: Nov 5, 2023 at 11:59 pm

Points: 100 points for full credit plus 50 points bonus

In project 2, we're going to explore how to implement DDoS attacks. The project will be divided in three parts: Socket programing (**required**), TCP SYN flood attack (**required**), and HTTP flood attack (**bonus**).

The instruction is based on **Python language**. You can also use C/C++, if you are more familiar with them. Ubuntu Operating systems are recommended to run the experiments. But macOS and Windows are also sufficient to run the experiments as long as you can run Python programs and installed required Libraries successfully.

We assume that everyone could run Python3 programs, if you have any issues installing python programs, please refer:

https://www.python.org/
https://realpython.com/installing-python/

In these two websites, you can find information of how to install and run the latest Python language.

To launch DDoS attacks, especially TCP SYN flood attack, we need to craft our own network packets. We could write it from scratch, but that requires a lot of technical details about the network layers and specifications. We stand on the shoulders of others in this step and using a packet crafting library named as **Scapy**. For more information about Scapy, please refer to:

https://scapy.net/

To use Scapy in our Python program, we need to install it in our system. You can refer to the following website for more information.

http://phaethon.github.io/kamene/api/installation.html

Please make sure that you can run the Scapy using **root** permission. If not, you may have a problem of importing libraries from Scapy at the beginning of the program.

The project consists of seven files:

main.py
multithreading_programming_example
main_server.py
socket_programming_example_client.py
socket_programming_example_server.py
tcp_syn_flood_attack.py
http_flood_attack.py

The main.py file includes five major sections of codes: Hello World example, multi-threading programming examples in Python, Socket programming examples, TCP syn flood attack examples and HTTP flood attack examples. Hello World example and multi-threading programming examples are ready to run. If you have cleared all the errors, you should be able to run them in the main.py file. Please remove the commented quotes of a particular section if you want to run that section.

You need to fledge the rest three files: Socket programming examples, TCP syn flood attack examples and HTTP flood attack examples.

Socket programming examples include two files, socket_programming_example_client.py and socket_programming_example_server.py
Client side: Client is to send message to server.
Server side: Server is to listen and receive message from client.
For both sides, we need to create socket in order to send or receive message.
Here are some useful guidelines: https://realpython.com/python-sockets/

1) Successfully create sockets (20 pts)
2) Successfully send and receive message (20 pts)
   Note: at least 30 pts will be deducted if the program cannot run successfully

In **TCP syn flood attack**, we need to implement a syn flooding attack. In order to flood the server, the client should be able to fabricate a number of syn packets with forged source IP addresses and port numbers to the server. Scapy is needed to create syn packet. Loopback address 127.0.0.1 should be used.

1) Successfully create syn packet (10 pts)
2) Randomly generate IP address and port number for each packet (10 pts)
3) The program can generate as many packets as we want (10 pts)
4) Using Wireshark to track the packets (10 pts)
   Note: at least 30 pts will be deducted if the program cannot run successfully

(**Bonus**) In **HTTP flood attack**, we need to implement a HTTP flooding attack. We consider a scenario of multiple clients, where multi-threading should be used to simulate different clients.

Each client needs to send a number of forged http requests with random URL, IP addresses, and port numbers. Loopback address 127.0.0.1 should be used.

1) Successfully create multiple threads (10 pts)
2) Successfully create HTTP packets with random URLs (10 pts)
3) Randomly generate IP address and port number for each packet (10 pts)
4) The program can generate as many packets as we want (10 pts)
5) Using Wireshark to track the packets (10 pts)
   Note: at least 40 pts will be deducted if the program cannot run successfully

Report (20 pts): Your report (in PDF) should be at most 5 pages in 10 point, double column format. Your write-up should not re-describe the assignment. The paper must be written using proper English grammar and should have no spelling mistakes. It should include:

- **Title:** The title should be descriptive and fit in one line across the page.
- **Author(s):** This should be right under the title, says who you are.
- **Abstract:** This is the paper in brief and should state the basic contents and conclusions of the paper. In general, the abstract is an advertisement that should draw the reader into reading your paper, without being misleading. It should be complete enough to understand what will be covered in the paper. Do not be afraid of giving away the ending!
- **Introduction:** This is a short overview of what you did, and what you learnt. This should contain more motivation than the abstract. Again, please make sure you include your main conclusions.
- **Methodology:** This should answer questions such as, what attacks you have implemented, for each tasks, explain clearly and in details how you went by doing it (what approach you used). Please include explanations about your timer accuracy, as well as a description of the platform you used to the level of detail such that someone else could reproduce the same experiments elsewhere.
- **Results:** This section should mainly consist of tables, figures (screenshot from Wireshark), each addressing the questions above. Also include code snippets with each plot so that the reader can follow your idea (avoid copying and pasting extra pieces of code).
- **Conclusions:** Summarize the conclusions here and discuss things you have learnt during this project.

Submission: Your submission should include **1)** Report in PDF, **2)** code that implements the objectives of this project **3)** README file describing how to run your project (e.g., arguments) and your team partner (if any). Compress all files in a tar file and submit the tar file via Canvas by due date (Nov 5, 11:59 pm)

Running a real-world programming example successful can be stressful and there have various problems that might prevent us from doing so. Please try your best to make these examples run successful on your computer.