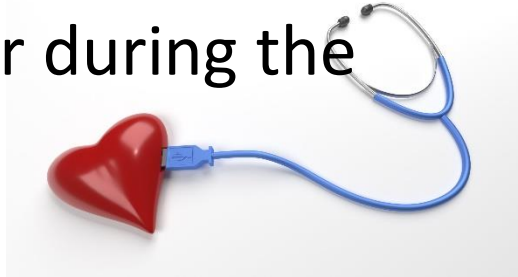
Today’s lab

- We will try to do a DDOS today.
- We will need 3 computers: Attacker, Target, Monitor.
- The attacker will send 10K tcp syn packets to the target machine, in a 100 iterations loop. Total of 1M packets.
- The target machine will run an apache server.
- The monitor will send pings to the server during the attack.



Today's lab

- The attacker will run two types of programs, one in C and one in Python.
- For each program, we will measure:

1. The time the attacker needs to send a packet (avg).
2. The ping's RTT from the monitor. You will have to send one ping every 5 seconds.

Today's lab

- The above measurements would appear in measurement files, in the following scheme:

syns_results_c.txt , containing each syn request (an index of the request or a counter) sent from C program and the time it took to send it, separated by newline. At the end of the file, a measurement of how long it took for the program to send <u>all</u> the packets, and the <u>average time</u> to send a packet.

syns_results_p.txt , same as above, but for the <u>Python program</u>.

Today's lab

- The above measurements would appear in measurement files, in the following scheme:

pings_results_c.txt , containing all the ping requests generated by the Monitor machine, while the Attacker attacked using the C program. Each ping request should be separated with newline. At the end of the file, you should add the average ping's RTT.

pings_results_p.txt , same as above, but while the Attacker used the Python program.

Today’s lab

- You’ll need to create matplotlib graphs describing the 2 metrics for C and for Python.
- *Syn_pkts_c.png & Syn_pkts_p.png* : The graph should describe the time needed for the attacker to send a packet. The x axis should be the time needed to send a packet. The y axis will represent the number of packets sent. One graph for the Python attack and another for the C attack.
- The y axis should be logarithmic, and the x axis should be distinguishable.
- You’ll also need to create a short report describing the results, with STD and average.
- You’ll need to include your understandings of the diffs.

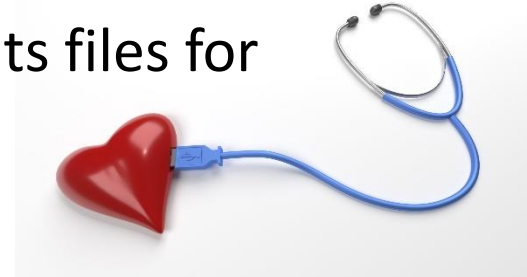
Today’s lab

- You’ll need to create matplotlib graphs describing the 2 metrics for C and for Python.
- *Pings_c.png & Pings_p.png* : The graph should describe the RTT for the ping requests. The x axis should be the RTT in milliseconds. The y axis will represent the number of pings. One graph for the Python attack and another for the C attack.
- The y axis should be logarithmic, and the x axis should be distinguishable.
- You’ll also need to create a short report describing the results, with STD and average.
- You’ll need to include your understandings of the diffs.

Upload box

- You'll upload a ID_measure.zip file including:

1. Attack.py and attack.c – attack files.
2. DDOS.pdf – report of the lab.
3. *Syn_pkts_c.png*, *Pings_c.png* – graphs of C.
4. *syns_results_c.txt*, *pings_results_c.txt* – results files for C.
5. *Syn_pkts_p.png*, *Pings_p.png*– graphs of Python.
6. *syns_results_p.txt*, *pings_results_p.txt* – results files for Python.



Notes

- Some notes:
- The DDOS attacks might run several hours depending on your hardware, do not start this lab 1 hour before the due date.
- After you saved the result files, you can duplicate them and manipulate the text so it will fit for the matplotlib for easier graph generation.
- The Monitor machine shouldn’t consume too much resources since it only sends ping requests and saves them into a file, therefore you can give this machine less memory and put more into the Attacker machine so the attack will finish faster.
- Remember that printing into the console also consumes resources and it is not needed during the attack, but if you insist to see what is happening, you can print the progress into the console every X packets/pings (like every 500 packets for example).

Bonus lab

- Now, we will look deeper to efficiency of script languages:
- Create two lists in python.
- The elements in the first one are random 5-character words.
- The elements of the second one are their sha-256.
- The length of the lists will be 1M.

Bonus lab

- In parallel to the list creation, create a dictionary.
- Every word from the list will be the key, and the value will be its hash.
- Generate 1000 random 5-character words.
- Measure the time you need to find a corresponding hash to a word, in the list vs in the dict.
- Now, create a c correspondent software.

Bonus lab

- Create matplotlib graphs for the dict and list, in python and c.
- The statistics will be to find a single hash for a word- average time and STD.

Bonus lab

- Therefore, the bonus lab will include a measure_additional_ID.zip:

1. lists_dict.py and lists_dict.c – the scripts.
2. lists_c.png, lists_p.png, dict_c.png, dict_p.png – graphs.
3. Additional.pdf – description and insights after running the lab.

- The bonus lab may grant 5 additional points.