**Cyber and Infrastructure Defense, Fall 2021, Haadi Jafarian**

**Assignment 3 – Think Like an Adversary: DDoS Simulation**

**Learning outcomes:**

* Think like an adversary: Understand the structure, goals, or methods of link flooding DDoS attacks with a focus on Crossfire DDoS attack.
* Develop a more thorough understanding of network topologies and its components.
* Learn about network simulation tools and expand your knowledge of network topologies and routing.
* Learn about log file analysis through a case study.

**Overview**: Simulation is the process of learning by doing. Whenever there is something new in the world, we try to analyze it first by examining it and in the process get to learn a lot of things. Network simulation (NS) is one of the types of simulation, which is used to simulate the networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as NS2. It is an object-oriented, discrete event-driven simulator written in C++ and Otcl/tcl. We will only use TCL scripting language in this assignment.

NS-2 can be used to implement network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms and many more. In ns2, C++ is used for detailed protocol implementation and tcl is used for the setup. Learn more about NS2 here: <https://www.geeksforgeeks.org/basics-of-ns2-and-otcltcl-script/>. A more detailed tutorial of NS2 could be found at: <http://nile.wpi.edu/NS/>.

In this assignment, you will use NS2 to simulate the Crossfire DDoS attack on a small network.

**Assignment Steps**

**Step 1. Download resources for assignment 3 from Canvas.**

It consists of the following files:

1. **ns2 application** (**ns.exe**): This executable runs tcl scripts with extension “**.tcl**” which are the OTCL code files. The TCL file defines a simulation scenario over a TCP/IP network written in OTcL/TcL language. Once **ns2** runs the **.tcl** file, it generates two types of outputs: an **out.nam** file which could then be given to **nsnam** tool (see below) to visualize the results of the simulation, and an **out.tr** file which includes a textual trace of the simulation. To execute a “**filename.tcl**” file using **ns2**, just run the command in a terminal (you need to traverse to the project folder).

***ns filename.tcl***

1. **nsnam.exe:**this tool receives an **.nam** file (output of ns2 execution) and visualizes the results. To visualize results, after **ns** execution is finished, in the terminal run

**nam *out.nam***

1. **Crossfire Attack paper ("The Crossfire Attack.pdf")**: This pdf is the paper describing the Crossfire DDoS attack.
2. **topology.pdf:**this is the topology of the network on which you will simulate the Crossfire attack.
3. **sample.tcl:**The file defines a small DDoS simulation scenario with two bots, one user, one router and one webserver. You can use this sample to understand how to define nodes, routers, and traffic sources in NS2, and then use this to develop the topology and simulation scenario. Run the simulation using **ns** and then visualize the scenario using the generated **out.nam** file and **nsnam** tool**.**

Shape

Description automatically generated with medium confidence

**Note:** If you have a Debian-based Linux machine, you can install **ns2** and **nsnam** as packages (see <https://www.geeksforgeeks.org/basics-of-ns2-and-otcltcl-script/>).

If you’re using Mac machine or Linux machines that do not include ns2 and nsnam packages, you can download and use this Debian VM in VirtualBox which has both ns2 and nsnam pre-installed on it: <https://drive.google.com/file/d/1iwQNwf8HbIyaHZw5B_7AOJw65glZZodo/view?usp=sharing>

**Step 2**. Read the “**The Crossfire Attack**” paper in the folder. Understand the general concept of the attack, which is also discussed in class. Specially focus on the concepts of flow density, degradation ratio, and steps of the attack.

**Step 3**. Develop the given network **topology.pdf** document using ns2 tcl scripting language. You can use Microsoft Code for coding, which includes extensions for TCL language. The network consists of several types of components as shown in the figure below:

1. Routers: they are regular nodes with links to other routers or hosts (bots, decoys, users, web server).
2. All links have bandwidth = 950Kb, Link Delay = 5ms, and queuing strategy = RED. You can use other setups as long as the objectives of the assignment do not change, and you mention your assumptions in your report. (Queue length does not need to be mentioned, but you can optionally set it to 5)
3. Bots: On each bot, there is a UDP traffic generator (CBR) that sends traffic to decoys with a rate that you need to determine.
4. Users: On each user node, there is a UDP traffic generator that sends traffic with a rate of **100Kbps** to Web server node. Here, we are using UDP to simulate Web traffic. But to simplify the simulation, we will use UDP traffic.
5. Decoys: Each decoy server has a “null” service and receives the UDP traffic from bot nodes. As mentioned later, you need to determine which bots should send traffic to which decoys and with what rate to achieve maximal degradation ratio.
6. Web server: The webserver has a “null” service and receives the UDP traffic from user nodes. Bots do not send their traffic to Web server directly.

**Step 4**. Using the concept of flooding networks by aggregating flows on *critical* links of the network, you are asked to simulate a simple version of Crossfire attack on the network defined in **topology.pdf**. Ignore the rolling property of the Crossfire attack, i.e., the attacker does not change its target set and only attacks one set of links.

To simulate the Crossfire attack you must determine a scenario such that maximum degradation ratio is achieved. Maximum degradation ratio is achieved when the maximum number of packets from the user nodes to the web server is dropped. Therefore, the objective is (a) to determine the target links and then (b) the rate with which each bot must communicate with each decoy server, such that the target links become congested and the number of benign packets that are dropped is maximized. That is, you must first determine the optimal set of links to flood (target links) based on flow density metric, and then generate the traffic rates from bots to decoy servers such that only those *target links* but *not* any other upstream links are congested/flooded. Specifically, you need to do the following:

1. Determine three target links in the topology and label them as target links 1, 2, 3 in the TCL code. These target links are the links with top 3 highest “flow density”.
2. For each bot, you must determine two things:
3. Which decoy server(s) it should communicate with? (bot to decoy assignment)
4. What should be the rate of traffic from each bot to each decoy server?

**Step 5**. Add the above scenario to your TCL code; that is, add code to send traffic from bots to decoys as identified in the responses to above questions. In a terminal go to the project directory where **ns.exe** is, simply run "**ns yourcode.tcl**". You can use **nsnam**tool to see your simulation visually. Note that you may need to play with your traffic rates to achieve the scenario with the highest degradation ratio. This process usually requires running multiple simulations (e.g. with different rates of CBR traffic) to achieve the desired effect.

**Step 6.** Using the guidelines below, analyze the trace file to calculate and plot a graph that shows the total number of packets dropped from each user node in every second during the simulation. Finally, calculate the degradation ratio as the percentage of total dropped packets for all users over time. So, your report will have one figure with 5 lines, 4 showing drop-rates per second for each of the 4 users, and one showing the total degradation ratio for all user traffic over time.

To analyze the output, you need to do processing on the output file with extension **out.*tr***which includes the raw simulated network traffic. In your analysis, you need to count the total number of packets sent by a user node, and then count how many of those packets are dropped.

For example, in **out.tr** file, we may see the following trace line for a packet with ID=326.

- 0.54768 0 2 cbr 210 ------- 2 0.0 4.0 163 326

This table defines the format of every trace line:

Table

Description automatically generated

For example, the trace line above includes the following information:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 0.54768 | 0 | 2 | cbr | 210 | 2 | 0.0 | 4.0 | 0.0 | 326 |
| Action taken | Time | Layer-2 source (source MAC) | Layer-2 destination  (destination MAC) | Traffic type | Flow ID | Layer-3 source  (source IP) | Source port | Layer-3 destination  (dest. IP) | Destination port | Packet ID |

This trace line shows a packet with ID=326 that has been sent from sourceIP=0 to destIP=4, but this trace line shows delivery of this packet from sourceMAC=0 to destMAC=2.

In fact, in the trace file, we see log of every layer-2 frame. Let’s look at all the trace lines related to this packet with ID=326 in our example:

+ 0.54768 0 2 cbr 210 ------- 2 0.0 4.0 163 326 (node 0 enqueues packet 326)

- 0.54768 0 2 cbr 210 ------- 2 0.0 4.0 163 326 (node 0 dequeues this packet and send it to next hop which is node 2).

…

r 0.554448 0 2 cbr 210 ------- 2 0.0 4.0 163 326 (node 2 receives the packet)

+ 0.554448 2 4 cbr 210 ------- 2 0.0 4.0 163 326 (node 2 queues the packet)

d 0.554448 2 4 cbr 210 ------- 2 0.0 4.0 163 326 (node 2 drops the packet)

Use the references online (e.g. <http://nsnam.sourceforge.net/wiki/index.php/NS-2_Trace_Formats>) to understand the structure of ns2 trace files in more details.

For every packet, you need to identify (1) which node sends the packet (source IP), (2) when (at which second) the packet was originally sent (round times up to seconds 1 to 10; ceil(0.554448) = 1) and (3) whether the packet is dropped or not. If a trace line with action “**d**” exists in the trace file, then it means the packet was dropped along the way; if not, it means the packet arrived successfully. For example, in the above example, packet 326 is sent by node 0 at second 1 and it is dropped.

There are various analyzer tools online that can expedite the process of analyzing your trace file. For Linux, the following link provides useful information about tools for analyzing ns2 trace file: <http://nile.wpi.edu/NS/analysis.html>

For Windows and Linux, you can also use the AWK scripts online.

You can also develop your own analyzer (with any programming language) to parse ".tr" file and extract the necessary information.

**Step 7.** Write a short report and describe your additional assumptions, your answers to steps 4 and 6, and your observations and conclusion.

**Upload only your tcl file, along with your report in Canvas. Please Zip the files.**

**Rubric for Grading**

Simulation of the DDoS Attack: max 50%

* Complete simulation of network topology, bots and clients, and traffic rates: 50%
* Missing some components of network topology, but working simulation: 35%
* Incomplete simulation: 20%

Degradation of clients’ reachability: max 20%

* High degradation, 20%
* Partial or minor degradation: 10%

Detailed analysis of DDoS impact, 30%

* Complete report including packet drops over time, 30%
* Report missing discussion of packet drops: 10%