Distributed Denial of Service Simulation: Victim, Attackers and Command-and-Control

Vincent Chu
Dept. of Electrical Engineering and Computer Science
York University
Toronto, Ontario, Canada M3J 1P3
cse13261@cse.yorku.ca

ABSTRACT

This report describes the design and implementation decisions for implementing the server and client software for the distributed denial-of-service simulation described in the assignment paper.

The simulation consists of 3 components: a coordinator or command-and-control agent, attacker agents, and a victim agents. The command-and-control agent issues an attack instruction and broadcasts it to all of the registered attackers. This instruction includes: the victim's address, a start date and time, and the duration of the attack. Each attacker keeps a list of active attacks received and has a scheduler that initiates the attacks at the specified time.

PROTOCOL DESIGN

All protocols are a TCP protocol where messages are encoded in ASCII text and terminated by the control characters carriage return `[CR]` immediately followed by linefeed `[LF]`.

The command-and-control component consists of two parts: the registrar server and client to broadcast instructions to attackers.

The registrar server listens for new attackers to subscribe to receive broadcasted instructions and for existing attackers to cancel their subscription to stopped receiving instructions or go offline. The allowed registrar requests are as follows, where *`host`* and *`port`* are the address and listening port of the subscribing attacker:

`SUBSCRIBE` *`host port`*`[CR][LF]`
`CANCEL` *`host port`*`[CR][LF]`

The command-and-control broadcasting or botnet component issues commands to the listening attackers.

The attackers then execute theses commands.

`SYNC` *`time`*`[CR][LF]`
`ATTACK` *`host port start duration`*`[CR][LF]`

If a `SYNC` instruction is received with the command-and-control's *`time`* stamp, the attacker will calculate the time latency between its clock and the command-and-control's clock, and adjust as needed.

If an `ATTACK` instruction is received with the specified *`host`* and *`port`* of the victim server, the *`start`* date and time in ISO 8601 format, and the *`duration`* of the attack in seconds, the attacker adds it to list of attacks that still need to be executed and a scheduler that checks every second whether the attack needs to be initiated at the particular time. Once the attack is initialized a thread is spawned and passed to the attack client to connect to the victim to start the attack.

The victim server implements a simple echo server that also logs the incoming messages to a text file. For the purposes of the simulation, the attacker will send the following message, where *`host`* and *`port`* are the address information of the attacker and `seconds` is the number of seconds the attack has been ongoing for:

`ATTACK` *`seconds`* `=` *`host`*`:`*`port`*`[CR][LF]`

All connections are terminated by the ASCII control character ETX (End of Text), hex value of 0x03.

For the purposes of this simulation, a control instructions between the attackers and the command-and-control are send request and close. No error handling or retry mechanisms are implemented. On error, the request is simply ignored. See Figure 1 for a general flow diagram of the instructions in the simulation and Figure 2 for an example of general schema of the protocols.

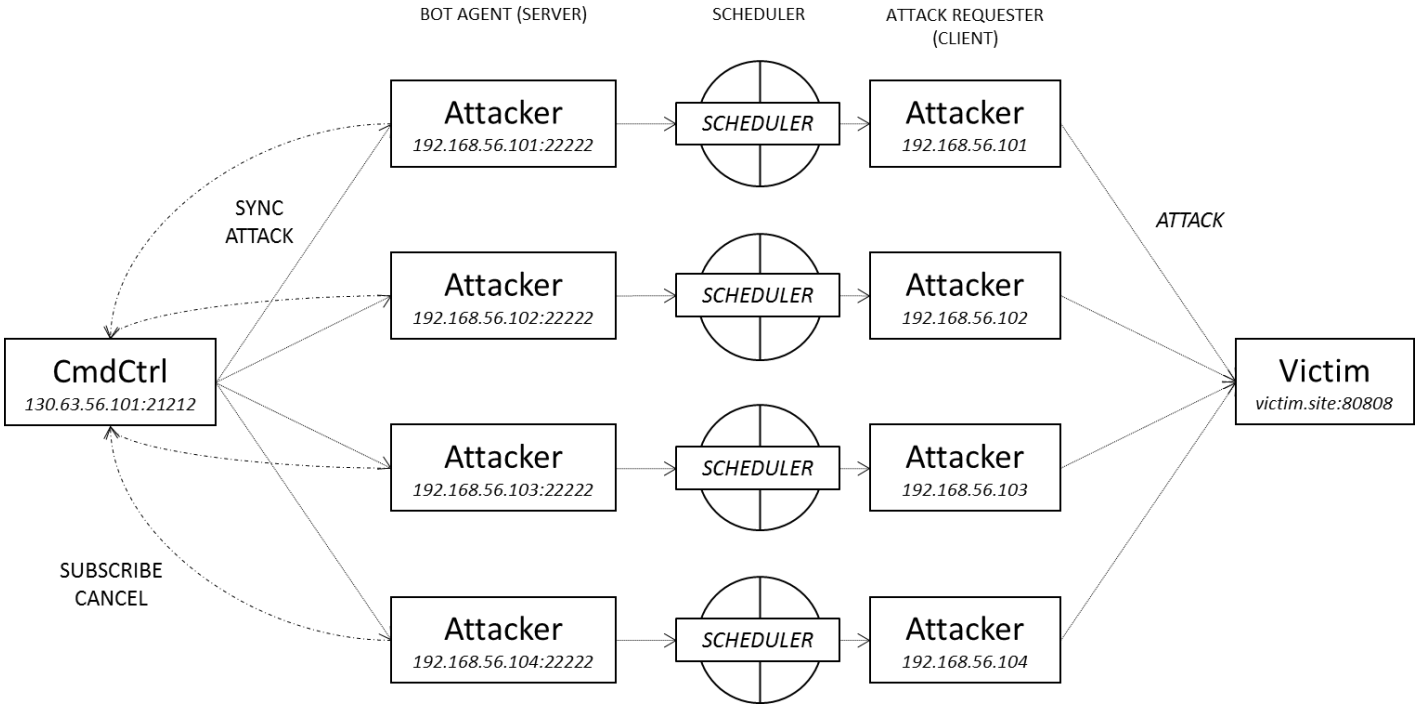

Figure 1 Data flow of the instructions between the components

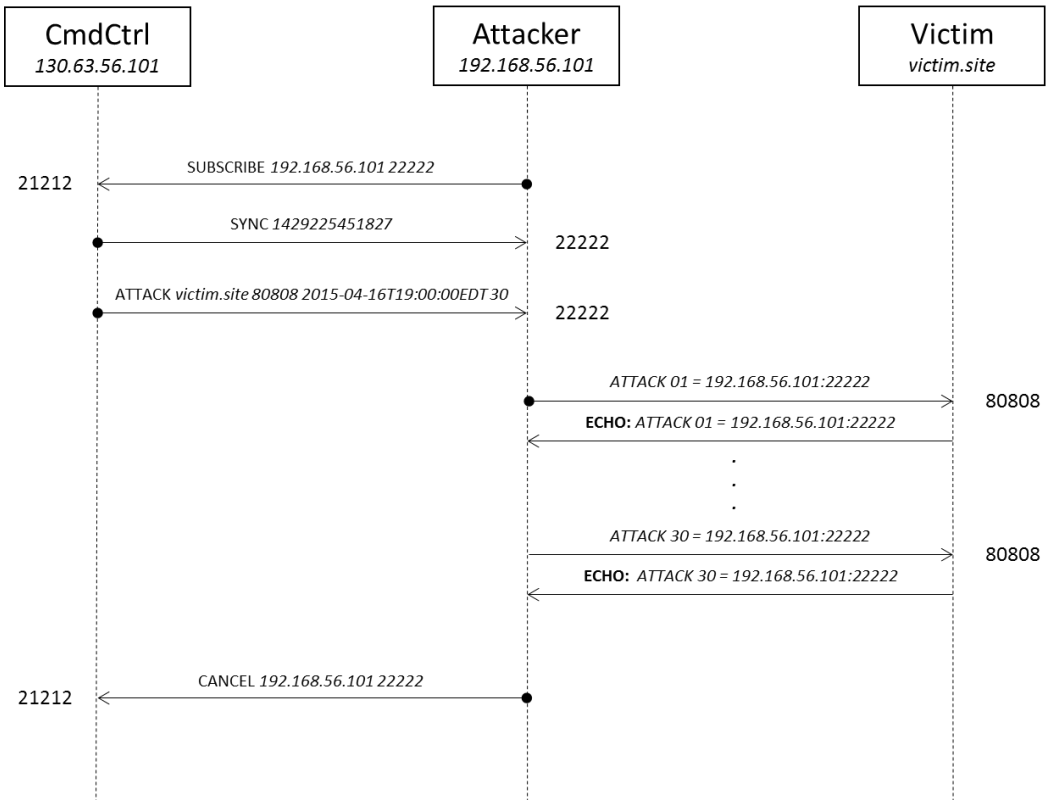

Figure 2 Example of the messages sent between the components

IMPLEMENTATION DESIGN

An implementation of the server and client that adheres to the specified protocol has been created and provided alongside this document.

The implementation has three command-line interface modes, one for the command-and-control, the attackers and the victim. See Figure 3.

All three components share common components that make it possible to easily modify, extend the simulation and write less code. The shared components include static utility classes for debugging and logging, compatibility with Java 1.7, and interfaces that define an Agent, Agent Factory and Error Handling callback.

The Agent interface enables developers to implement protocol logic over the entire session of the connection, without having to worry about state management, similar to the interface provided by Java Servlets. In the server and client stacks, each has a specialize implementation to handle the two roles in the protocol.

The Agent Factory interface enables developers to implement server logic to create new agents for each new incoming request to the server.

The Error Handle interface allows developers to implement a callback when a request to the server encounters an error at the client.

Additionally the Requester and Simple Requester abstract classes implement Agents that provide a mechanism for sending requests to the server from the client.

The Command Executor class is used to implement a read–eval–print loop for the Command Line class and is also used to evaluate the requests received on the server-side.

With the classes described above, a generic client and server are implemented and shared between all three components. A Main class is used to initiate all three components, in their respective modes.

COMPILATION AND USAGE

To compile or running the application, Java Standard JDK 1.7 or later is required. Compilation can be running from the command-line terminal. To compile the entire project, execute the command (from the project directory):

```
javac Main.java
```

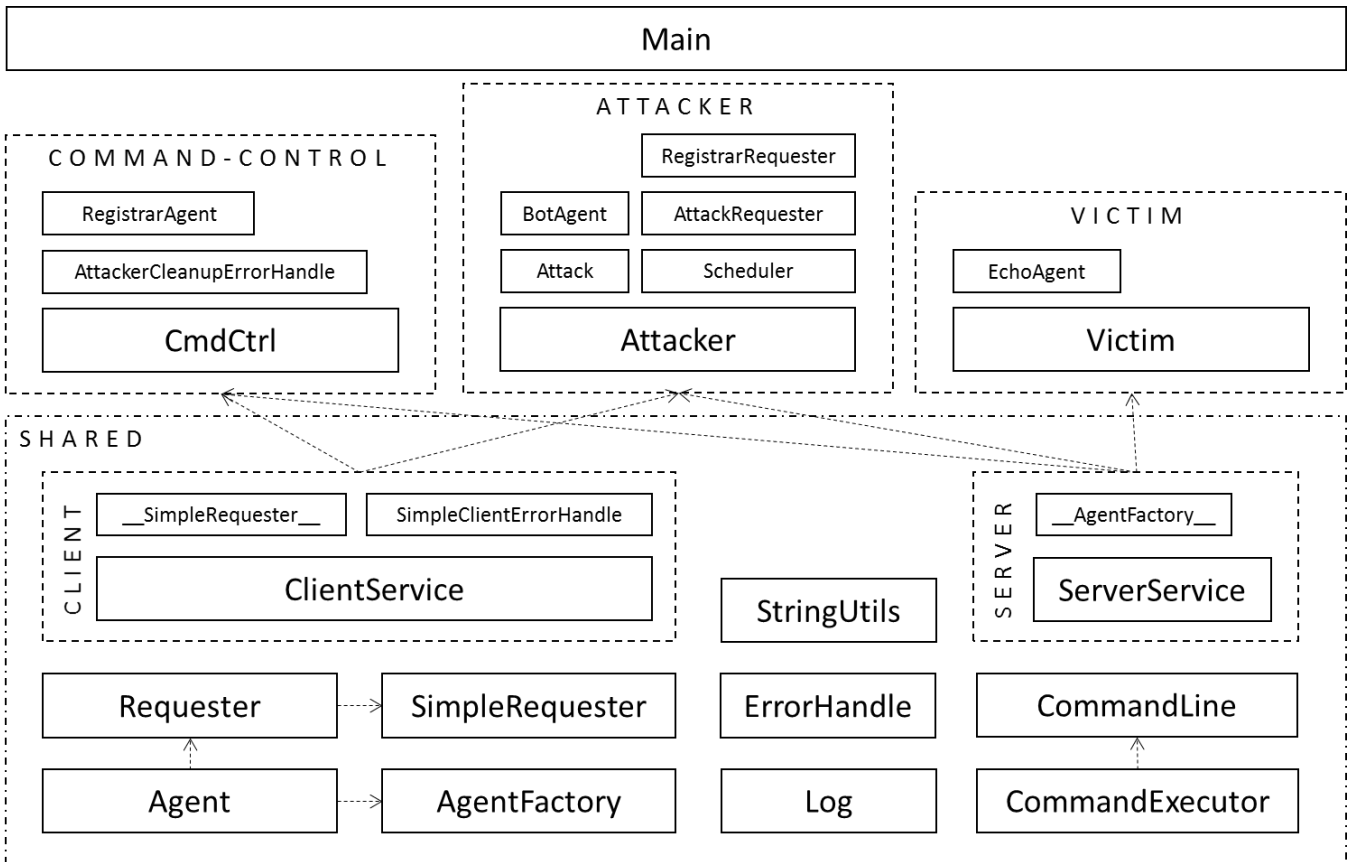

Figure 3 Simulation's Software Architecture

To start a command-and-control instance, open a separate terminal window and execute the command below (from the project directory):

```
java Main CmdCtrl
```

The `CmdCtrl` CLI provides a small set of commands that can be enter to perform certain operations. For a list of the available commands and their associated usages, see Table 1.

Table 1 Available commands for CmdCtrl CLI

START *port*

Starts the server and listens to registration from attackers at the given *port* number.

STOP

Stops the registrar server.

LIST

Prints all the registered attackers.

SYNC

Broadcasts an instruction to synchronize all of the attackers' clocks.

ATTACK *host port start duration*

Broadcasts an attack instruction to all the attackers, given the *host* name and *port* number of the victim server, the attack *start* date and time in ISO 8601 format and the attack *duration* in seconds.

HELP

Print this usage output.

EXIT

Terminates the program.

To start an attacker instance, open a separate terminal window and execute the command below (from the project directory), where *host* and *port* are the address of the command-and-control server:

```
java Main Attacker host port
```

The `Attacker` CLI provides a small set of commands that can be enter to perform certain operations. For a list of the available commands and their associated usages, see Table 2.

Table 2 Available commands for Attacker CLI

START *port*

Starts the server and listens for instructions from the command-and-control system at the given *port* number. Requires that command-and-control server is running.

STOP

Stops the botnet server.

LIST

Prints all the attacks scheduled.

DELAY

Prints the time delay in the clock relative to the command-and-control server, disregarding the networking delays.

ATTACK *host port start duration*

Adds an attack to the list of attacks for scheduling to the attack the victims, given the *host* name and *port* number of the victim server, the attack *start* date and time in ISO 8601 format and the attack *duration* in seconds. Requires that the attacker server is started.

HELP

Print this usage output.

EXIT

Terminates the program.

To start a victim instance, open a separate terminal window and execute the command below (from the project directory), where *logfile* is the file path to the file where requests are saved:

```
java Main Victim logfile
```

The `Victim` CLI provides a small set of commands that can be enter to perform certain operations. For a list of the available commands and their associated usages, see Table 3.

Table 3 Available commands for Victim CLI

START *port*

Starts the server and listens for incoming requests at the given *port*.

STOP

Stops the server.

HELP

Print this usage output.

EXIT

Terminates the program.