***Nitin Nagavel***

***CPRE 431***

***M04 HW***

**Assignments will be submitted in PDF format via Canvas.**

Please submit your homework online through Canvas. Late homework will not be accepted.

Important: Your submission must be in .pdf format ONLY!

Please ensure that you support all your answers with the correct screenshots showing your solutions.

Layer 7 DoS attack with slowloris

This experiment explores slowloris, a denial of service attack 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](http://groups.geni.net/geni/raw-attachment/wiki/RUP/RUP.pdf).

To reproduce this experiment on GENI, you will need an account on the [GENI Portal](http://groups.geni.net/geni/wiki/SignMeUp), and you will need to have [joined a project](http://groups.geni.net/geni/wiki/JoinAProject). You should have already [uploaded your SSH keys to the portal and know how to log in to a node with those keys](http://groups.geni.net/geni/wiki/HowTo/LoginToNodes).

## 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.

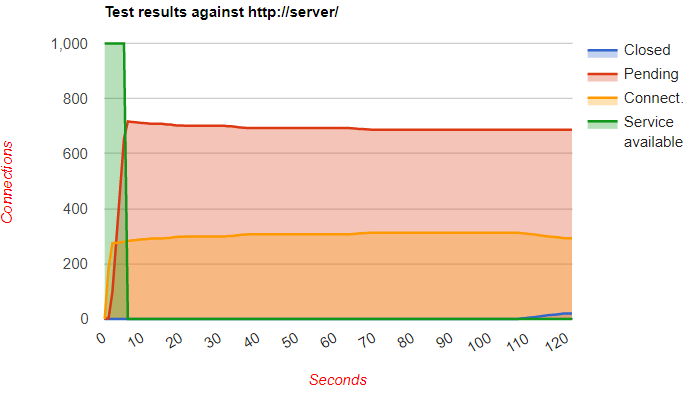
Slowloris is a type of denial of service attack that operates at Layer 7 (the application layer). It exploits a design approach of many web servers, allowing a single machine to take down another machine's vulnerable web server with minimal bandwidth.

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.

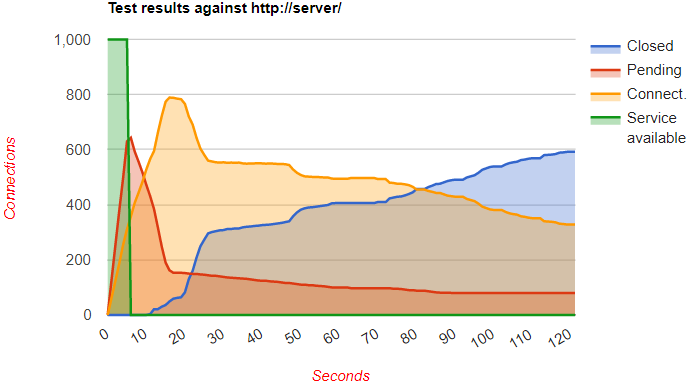
Slowloris was used in 2009 against Iranian government servers during protests related to the elections that year.

## Results

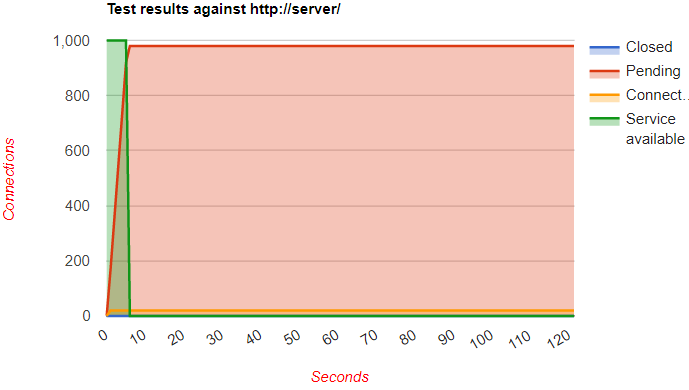
The following image shows the response of an Apache webserver to a slowloris attack. We see that when there are a large number of established connections, the service becomes unavailable (green line goes to zero.)



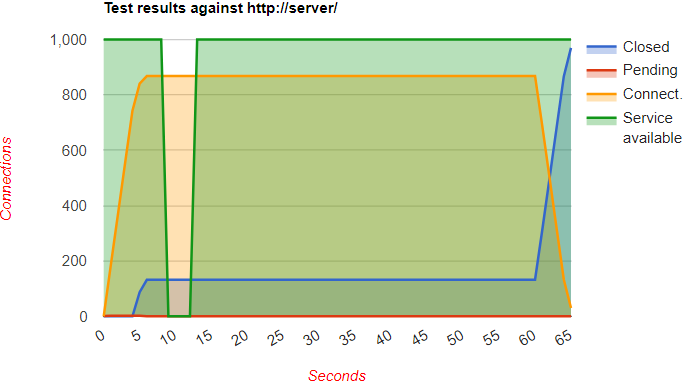
When we limit the rate of traffic from the attacker to 100 kbps, the attack is still successful:



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:



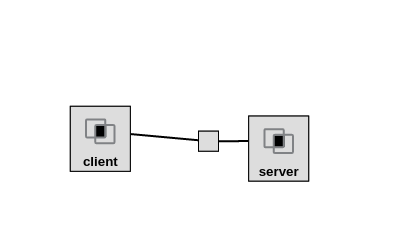
Finally, we found that the nginx web server is resistant to slowloris (even without a firewall limiting the number of connections per host) because of its non-blocking approach, which supports a higher level of concurrency:



## Run my experiment

In the GENI Portal, create a new slice, and load the [RSpec](https://gist.github.com/ffund/b9b69d4a118d009761a7aca664f0324a) from the following URL:  <https://iastate.box.com/s/1srjqn5bn9mdqgzu1laynjqq3rygxixm>

This will load a topology with two nodes connected by a link, like this:



Click on "Site 1" and choose an InstaGENI aggregate, then reserve these resources (include screenshot after reserving the resources).

Graphical user interface, website

Description automatically generated

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

lynx [http://server](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](https://github.com/shekyan/slowhttptest/wiki) tool. This tool implements several Layer 7 DoS attacks, including slowloris.

Then, on the client, run

slowhttptest -c 1000 -H -g -o apache\_no\_mitigation -i 10 -r 200 -t GET -u [http://server](http://server/) -x 24 -p 3 -l 120

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

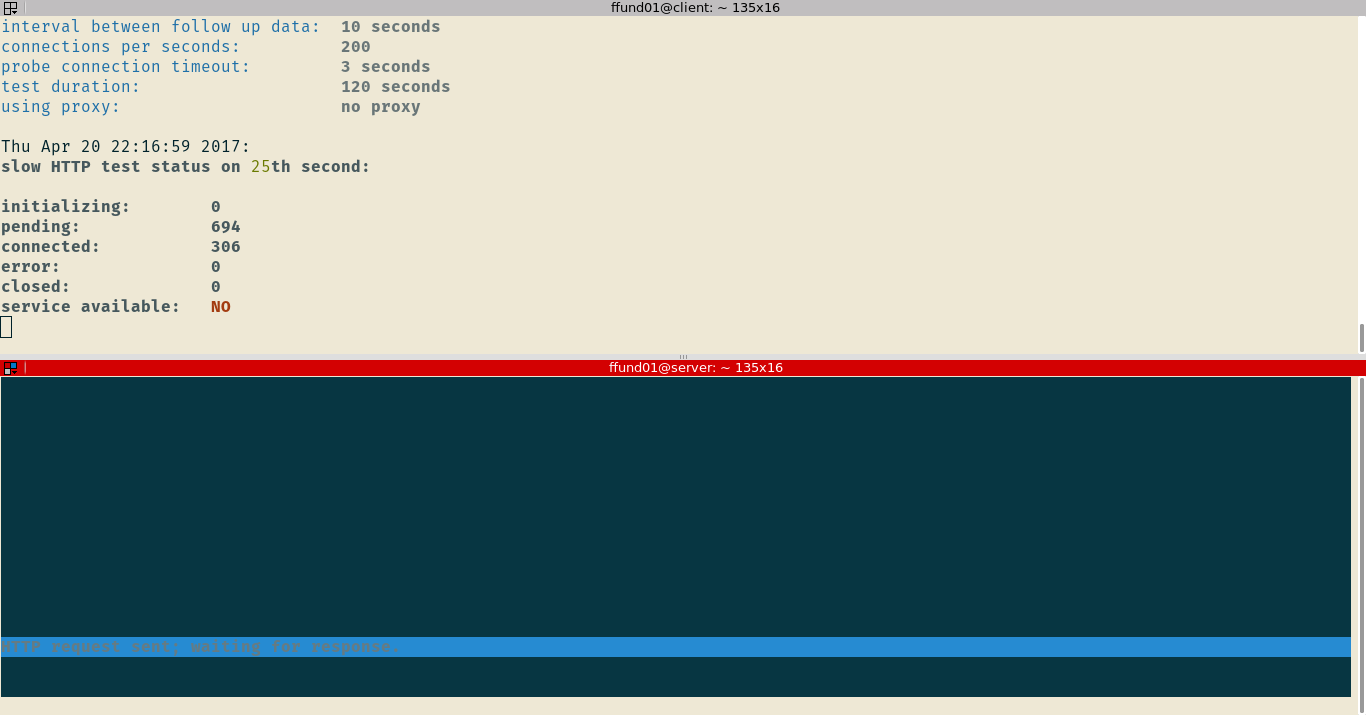
service available: NO

means that the DoS attack on the webserver was successful.

This test will run for 120 seconds. After about half a minute, while the test is still running, try to access the web page again on the server node by running

lynx [http://server](http://server/)

and verify that it is not responsive:



Also, 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.

After the test finishes running, transfer the "apache\_no\_mitigation.html" to your laptop with scp.

Please refer to the following video for how to SCP: <https://iastate.box.com/s/0d5ohi6awzcracupthtn9c2qeihs1jr8>

Open this file with a web browser. You should see an image similar to the first one in the [Results](https://witestlab.poly.edu/blog/slowloris/#results) section, indicating that a large number of established connections have made the service unavailable.

Let us explore several ways to mitigate this kind of attack.

First, let's see if this attack is still feasible when the client has very limited bandwidth. On the client node, run

ifconfig

and find the name of the network interface that is connected to the server. Then, run

sudo tc qdisc replace dev eth1 root netem rate 100kbit

substituting the name of the interface you have found in the previous step for eth1. This will limit the rate of outgoing traffic on this interface to 100 kbps.

Now we'll run the slowloris attack again. On the client, run

slowhttptest -c 1000 -H -g -o apache\_lowrate\_client -i 10 -r 200 -t GET -u [http://server](http://server/) -x 24 -p 3 -l 120

Wait about 30 seconds and then check the service availability by running lynx again on the server. When the test finishes (after 120 seconds), transfer the "apache\_lowrate\_client.html" file to your laptop with SCP and open it in a browser. Open this file with a web browser. You should see an image similar to the second one in the [Results](https://witestlab.poly.edu/blog/slowloris/#results) section, indicating that even when the attacker has very little available bandwidth, the attack can still be successful.

Remove the rate-limiting traffic shaper on the client with

sudo tc qdisc delete dev eth1 root

substituting the correct interface name in the command above.

Next, we will try using firewall rules to mitigate this attack. Specifically, we will create a rule that says that any single host is limited to 20 connections to port 80 on the server.

On the server, run

sudo iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 20 --connlimit-mask 40 -j DROP

to set up this rule.

On the client, run

slowhttptest -c 1000 -H -g -o apache\_iptables -i 10 -r 200 -t GET -u [http://server](http://server/) -x 24 -p 3 -l 120

and then, after half a minute, run

lynx [http://server](http://server/)

on the server to check the availability of the service. Even when slowhttptest reports

service available: NO

we can still load the page in lynx on the server:



This is because the service is only unavailable to the malicious user. The firewall does not affect a non-malicious user.

Transfer the file "apache\_iptables.html" to your laptop with SCP and open it in a browser. Compare it to the third figure in the [Results](https://witestlab.poly.edu/blog/slowloris/#results) section.

While this mitigation prevents a slowloris attack that is launched from only one host, it still would not protect against a distributed slowloris attack, with many participants each consuming a smaller number of connections. Also, if the number of allowed connections per host is set too low, it might limit connections from clients behind a NAT or a proxy, which share the same IP address. Use:

sudo iptables --flush

to remove the firewall rule on the server.

====================================================================

All steps below this line are optional

====================================================================

We are going to try one more way to mitigate this attack: changing the application design.

The Apache web server allocates a worker thread for each connection, allowing a slow or idle connection to block an entire thread. When the total number of worker threads is exhausted, then no new connection is accepted.

In contrast, the nginx web server has a non-blocking design, in which worker threads are not assigned to connections on a one-to-one basis. Instead, a thread will dynamically serve a connection only when there is data to send or receive for that connection. This makes it more resistant to the slowloris attack at Layer 7 (although it may still be possible to launch a low-rate attack that exhausts the total number of connections possible at a lower level, such as the total number of file descriptors available to the operating system.)

On the server node, stop the Apache server with

sudo service apache2 stop

Then install and start nginx:

sudo apt-get update

sudo apt-get -y install nginx

sudo service nginx restart

If you still have issues stopping apache2 try to delete the resources and reserve them again (don’t forget to install slowhttptest again on the client side) then install and start nginx as follows:

sudo apt-get -y install lynx-cur nginx

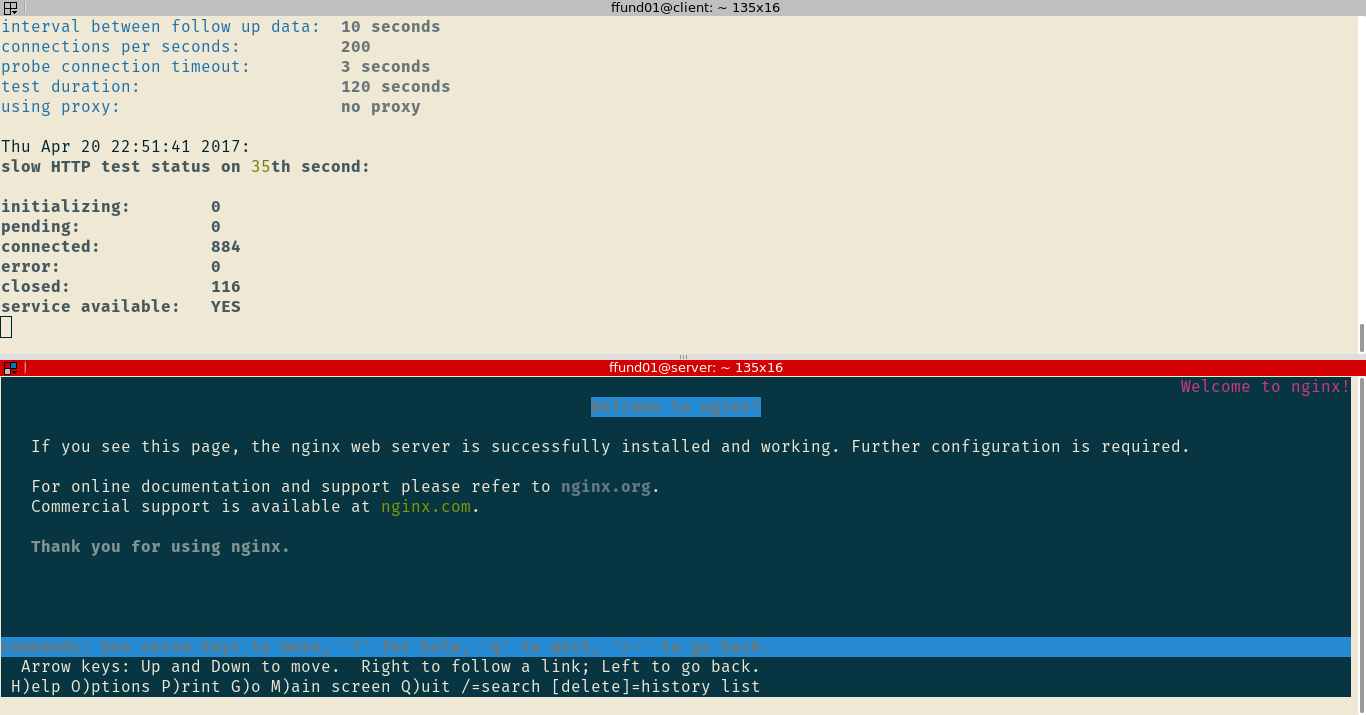
Run the attack from the client again, with

slowhttptest -c 1000 -H -g -o nginx\_no\_mitigation -i 10 -r 200 -t GET -u [http://server](http://server/) -x 24 -p 3 -l 120

and after 30 seconds, run

lynx [http://server](http://server/)

on the server. Here, you should see a "Welcome to nginx" page:



When the attack finishes, transfer the "nginx\_no\_mitigation.html" file to your laptop with SCP and open it in a browser. While you may see some brief outage, you should find that the service generally remains available (even to the malicious attacker) despite a large number of established connections (as in the fourth figure in the [Results](https://witestlab.poly.edu/blog/slowloris/#results) section.) Due to the difference in application design, this web server is less vulnerable to the slowloris attack.

Please delete your resources in the GENI Portal when you're done, to free them up for other experimenters!