Carlos Martinez
Feb 6, 2019
u0969416

Design

When it came to the design of the program, we started with an echo server and client. The server was designed to get connected once by 1 client, sending back the client's message then finishing. To start with the proxy, I simply combined the echo server and client into 1 program. Then multithreading was suppose to be apart of it, so the decision was made to put the client listener in the main thread listening for clients and when a new client connections a new thread was created to handle the client in the correct way. In my program "def **clientthread**(client):" is the method that handles the functionality for each method, if the client fails or throws an exception then the thread may crash or die, but the server can still continue with the rest of the clients.

The clientthread method starts with handling doing checks that the request of the client is valid, if it fails the test the proxy will get a bad request message. If the requests pass the tests then it then tries to figure out weather it is an absolute or relative request form. Once the proxy determines the type of request it then tries to get the port and address to connect to the proper server. If the connection fails it then gets a bad request otherwise it then moves into sending and receiving the message with the remote server. If minor tests determine that the response from the remote server is safe it moves on to making the proper modifications (Simple to Silly). If the response from remote server fail the test or can't be modified it then moves to collecting the proper information to connect to the virustotal server.

We will then try to connect the virustotal server and parse it's response to determine if the request contains malware or not. If the the request does not contain malware the modified response gets send back to the client otherwise the client gets a malware response. Once the client gets its response the thread ends.

Due to the design that each client is in there own thread, if anything goes wrong with the client request the client only affects their own thread but the proxy is still able to go on with the rest of the clients. Due to the design of the client thread method, the client is able to handle multiple types of requests and depending on the type of request the appropriate steps can be taken.

Testing

When it comes to testing multiple tests were done.
GET http://www.cs.utah.edu/~kobus/simple.html HTTP/1.1\r\n the request is not valid because the request does not end in \r\n\r\n, bad request received.

GET http://www.cs.utah.edu/~kobus/simple.html HTTP/1.1\r\n\r\n the request is valid get ok request and the Simple to Silly is also change correctly.
The same applies to GET http://www.cs.utah.edu/~germain/CS4480/ HTTP/1.1\r\n\r\n and GET http://www.google.com/ HTTP/1.1\r\n\r\n

GET http://www.cs.utah.edu:80/~germain/CS4480/index.html HTTP/1.1\r\n\r\n The request works appropriately testing the port number will work in a request and print statements determine that the request took the correct path in the method

GET http://www.cs.utah.edu/~germain/CS4480/ HTTP/1.1\r\nCookie: $Version=1; Skin=new;\r\n\r\n The request tests that the proxy can handle other headers in the request. It works as expected.

ST http://www.google.com/ HTTP/1.1\r\n\r\n The request tests that this gets handles as a bad request

DELETE http://www.google.com/ HTTP/1.1\r\n\r\n The request tests that this is a not implemented request

GET http://www.cs.utah.edu/~germain/CS4480/abc HTTP/1.1\r\n\r\n The request tests that this is malware

GET /~germain/CS4480/abc HTTP/1.1\r\nHost: www.cs.utah.edu\r\n\r\n The request tests that the relative path works

*GET /~germain/CS4480/abc HTTP/1.1\r\nHost: www.cs.utah.edu\r\n*Cookie: $Version=1; Skin=new;\r\n\r\n The request tests that relative request works with other headers

*GET /~germain/CS4480/abc HTTP/1.1\r\nHost: www.cs.utah.edu:80\r\n*Cookie: $Version=1; Skin=new;\r\n\r\n The request tests relative with port

All of the requests produced the appropriate and expected response

Output

The below are examples of some of the output my program produce then testing my proxy

```
message: GET http://www.cs.utah.edu/~kobus/simple.html HTTP/1.1\r\n\r\n
********** Message Received by Server **********
HTTP/1.1 200 OK
Date: Wed, 06 Feb 2019 17:30:56 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Sun, 12 Jan 2014 04:01:24 GMT
ETag: "46-4efbe03469098"
Accept-Ranges: bytes
Content-Length: 70
Vary: Accept-Encoding
X-Frame-Options: sameorigin
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html>
<body>
<h1>Real Silly</h1>
</body>
</html>

message: GET /~germain/CS4480/abc HTTP/1.1\r\nHost: www.cs.utah.edu:80\r\nCookie: $Version=1; Skin=new;\r\n\r\n
********** Message Received by Server **********
HTTP/1.1 200 OK
Date: Wed 06 Feb 2019 10:43:45 GMT
Server: CS4480-Proxy
Connection: close

<html>
 <body>
   <h1>The File you requested appears to contain Malware.</h2>
   <h2>Information:</h2>
  <ul>
    <li>MD5 Hash: 2a9d0d06d292a4cbbe4a95da4650ed54</li>
    <li>Positives: 63/69</li>
    <li>Scan Date: 2019-02-05 06:41:37</li>
    <li>First Scan ID: Bkav</li>
   </ul>
   <p>Thanks to VirusTotal for this information.</p>
   <p>For more information see <a href="https://www.virustotal.com/file/09a1c17ac55cde962b4f3bcd61140d752d86362296ee74736000a6a647c73d8c/analysis/1549348897/">Virus Total Permanent Link</a></p>
 </body>
</html>

DELETE http://www.google.com/ HTTP/1.1\r\n\r\n
********** Message Received by Server **********
HTTP/1.1 501 Not Implemented

message: GET http://www.cs.utah.edu/~kobus/simple.html HTTP/1.1\r\n
********** Message Received by Server **********
HTTP/1.1 400 Bad Request
Date: Wed, 06 Feb 2019 10:28:52 GMT
Server: CS4480-Proxy
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at www.cs.utah.edu Port 80</address>
</body></html>
```