NETWORK SECURITY HW1

GUR TELEM 206631848

1. Attacks on networks and malicious software

1.1. **ARP spoofing.**

1.1.1. *What is ARP spoofing.* An ARP spoofing attack is a type of attack where an attacker repeatedly identifies as a different computer on the network and thus causes a false linking between the victim computer's IP and the attacker's MAC address on other devices on the network. Then, whenever a device on the network wants to send a message to the victim's computer, it will reach the attacker instead (because communication in the LAN is done by MAC addresses).

1.1.2. *What can an attacker get from such an attack.* By using such an attack, an attacker can receive sensitive information. For example, if an attacker pretends to be an authentication server on the LAN, it will receive all of the authentication messages sent by clients and by that will be able to collect login details. The attacker can even forward the requests from the clients to the real destination endpoint and such they will effectively be doing a mitm attack.

1.1.3. *How can an attacker answer first.* Assuming an attacker knows which computers are going to send an ARP request, they can repeatedly start sending the response (identifying as the victim) before they even receive the request and by that, they will most likely be able to answer before even the legitimate computer (victim) will even receive the request to identify.

1.1.4. *Suggest a way to protect against an ARP attack.* **Good suggestion:**
The simplest way to block such an attack is to only allow automatic IP assignment by the DHCP server and then to set all the routing devices on the network to block any ARP responses with mismatching MAC/IP combination. If an endpoint requires a static IP address, it should be set in the DHCP server to always reserve that IP for the endpoint's MAC address.
A not very good suggestion:
If we want to allow self assigned static IP for endpoints we can still partially protect from ARP attacks by caching on routing devices (routers, switches, etc...) the ARP requests and if they get an ARP response message that doesn't match any request in the cache, they will block the packet.
This will not prevent an attacker from impersonating a victim located far away from the endpoint that sent the ARP msg (which will make it more likely the attacker will answer first as they are closer).

1.2. **Why should users install security updates ASAP.** Users should install security patches and updates ASAP because when a patch is issued, then it's a bonfire for attackers to exploit the issue and start developing malicious attacks based on the breach. Announced vulnerabilities are called a one-days (zero-days are exploits not publicly announced or known, and most times unpatched too).
An example for this is the Blaster worm (Chinese researchers RE the security patch and created the worm).

1.3. **Student in MIT in 1988 (Nov 3rd) had his computer crashing every few minutes.**

1.3.1. *Why did the computer crash that day.* The computer crashed because that day there was a worm running on the network (written by Robert Tappan Morris) that had a problem that caused the computers on the network to exponentially fill up with processes of itself and crash them.

1.3.2. *Why did it take a few minutes after restart to crash.* The worm didn't save any files on the disk (or rather it deleted them after loading everything to memory). So it had no persistency after a reboot of the machine. But the exploit still existed on the network so the student's computer got infected again. The bug that caused exponentially increasing number of instances of the worm to run on the machine, until the process table filled and the computer crashed (there was a 1 to 7 chance that the worm would not kill itself even if there's a duplicate of itself). The process took a few minutes.

1.3.3. *Suggest a simple way for the student to prevent his computer from crashing until he finishes the assignment.* Since the worm relied on re-infections after restart, the user could disconnect the internet, restart (to make sure it wasn't infected already), and then not re-connect the internet until the assignment is done.

1.4. **Buffer Overflow questions.**

1.4.1. *What is BOF and how can a worm use it.* BOF (buffer overflow) is an vulnerability that allows more data to be written to a buffer than the buffer can hold and by that, overwriting the memory adjacent to the buffer (that contains other information, such as the return address from the function).

The worm can use the BOF vulnerability by replacing the return address of the function by the address to a function that the worm write (to the buffer for example). The worm's code can be written to the buffer and the return address is replaced with the address of the buffer. Then, when the function reaches the `ret` instruction, it will "return" (jump) to the address of the buffer and start reading it as instruction to execute. This kind of exploit is called arbitrary code execution.

1.4.2. *Give two examples for worms that used BOF.*

(1) The worm from section 3 at MIT in 1988 that exploited a BOF in the `finderd` daemon.
(2) A second example would be the worm Blaster which used BOF in the RPC functionality which was patched in MS03-026

1.4.3. *Choose one example and explain how it used BOF (Blaster).* The RPC service was used port 445 which was open by default on every computer. In combination with a BOF vulnerability and the lack of mitigation on which code can be executed (today, code on the stack is set to RW but not E), it was able to write the execution code to the buffer and using the BOF to overwrite the ret address such that the it would point to the written code in the buffer.

1.5. **What could be "fixed" in the worm s.t. only one instance (on average) will be running on each infected PC?.** The worm could remove the "feature" that even if another instance is detected, one in 7 times, none of them will terminate.

If we want to protect against falsified signals that another instance exists the worm can hold a symmetric encryption key (will be identical in all copies) or even simply a salt (for hashing). The worm will send a random number and ask the other potential instance to encrypt/hash that number, and if the result is correct, it will start negotiating which one will terminate. To prevent someone extracting the key/salt, we can use a custom packet to make reverse engineering of the process nearly impossible (at least impractical).

1.6. **What is a syn attack, how is it done, what resource does it fill, and what is the purpose of that attack?** A syn attack is an attack where an attacker sends a lot of syn requests which will require the victim to allocate resources to initiate the connection.

The sender will usually change the src IP (IP spoofing) to prevent detection as well as prevent identification of a syn request as malicious.

After sending the initial syn request, the attacker would not send any ack messages and thus force the victim to hold resources until it times out.

The syn attack fill the connections queue and does prevents the victim from creating connections with other endpoints (DoS attack).

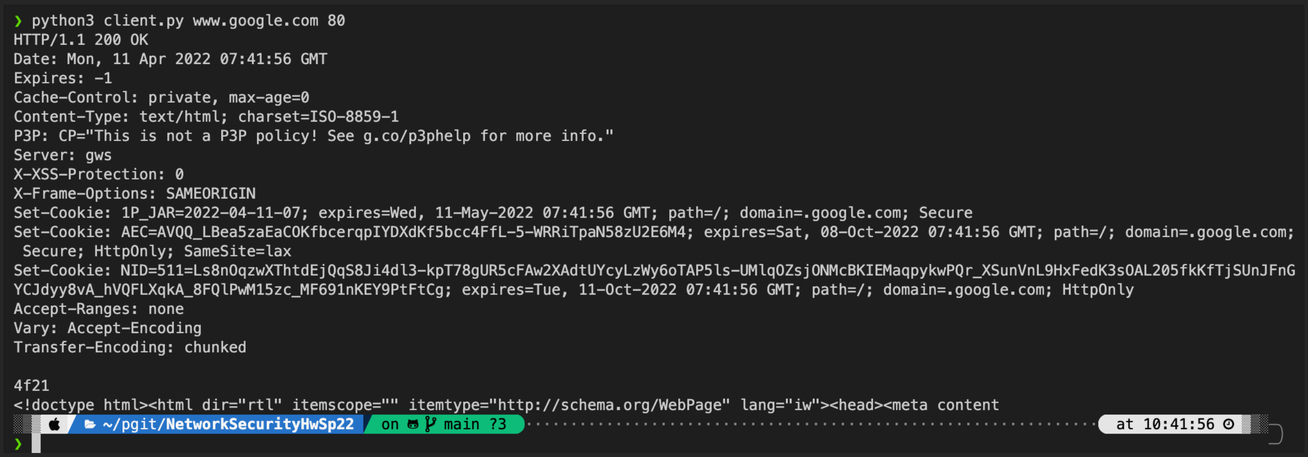
With a combination of a botnet we can DDoS a server with a syn attack and make it inaccessible for almost all users (statistically inaccessible).

2. TCP/IP

2.1. **What is the layers model described in the lecture.** There are 5 layer, each with its own role (some protocols don't use all the layers and may use only the lower level layers):

(1) Application layer - The layer that contains the actual data the application needs (protocols in this layer: FTP, HTTP, etc)
(2) Transport layer - Will split (and merge at the destination) packets. When all the fragments are received, it will pass it back to the application layer (protocols in this layer TCP, UDP, etc).
(3) Network layer - Responsible of routing packets to the correct destination.
(4) Link layer - Responsible of sending packets between devices physically connected with a cable (with no other devices between them).
(5) Physical layer - The actual physical signals being sent (optical signal, electronics signals, radio, etc....). Handles the signals and converts them into bits and bytes.

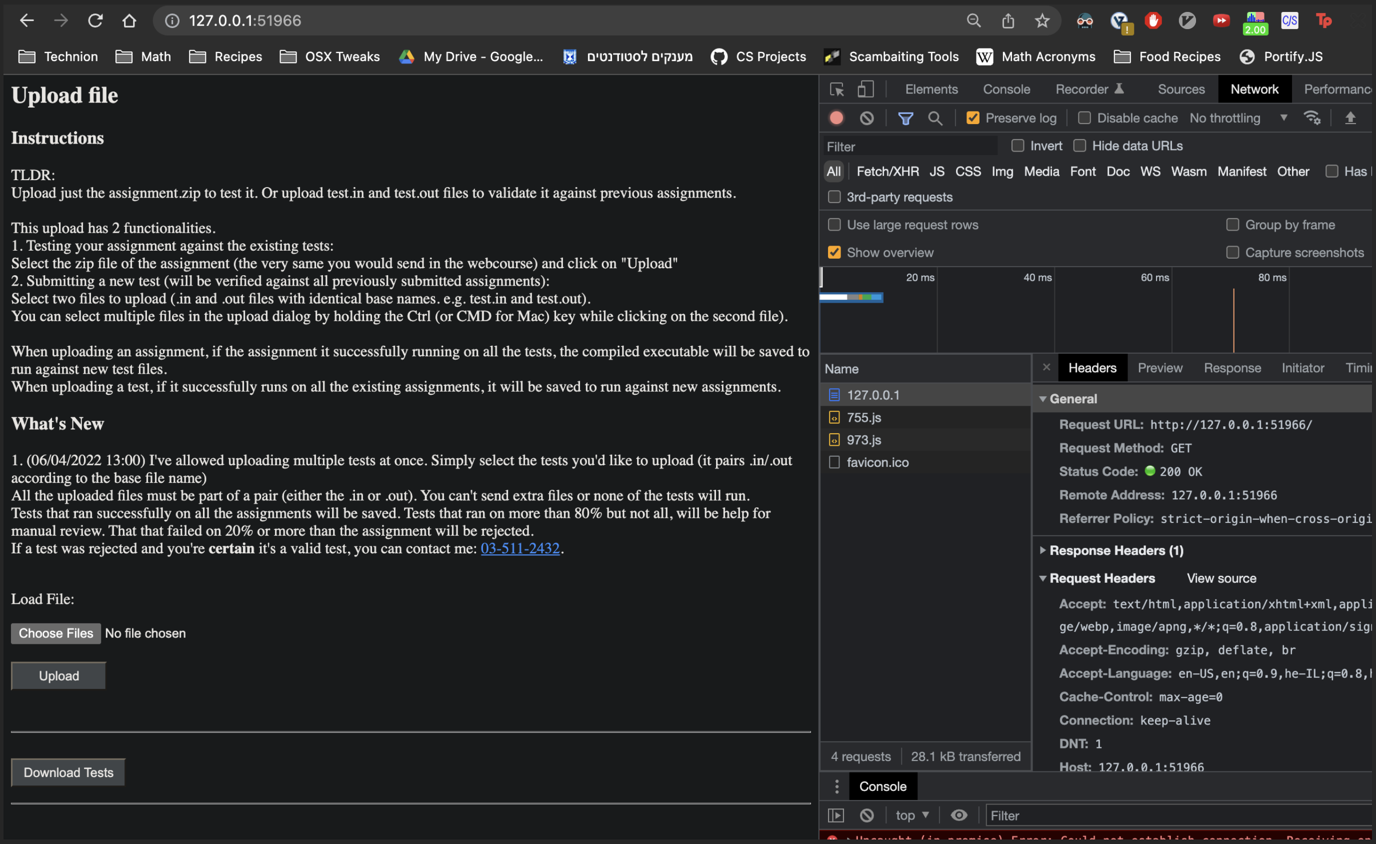
2.2. **Write Python** `client.py`**.** Screenshot of execution result:



```
❯ python3 client.py www.google.com 80
HTTP/1.1 200 OK
Date: Mon, 11 Apr 2022 07:41:56 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2022-04-11-07; expires=Wed, 11-May-2022 07:41:56 GMT; path=/; domain=.google.com; Secure
Set-Cookie: AEC=AVQQ_LBea5zaEaCOKfbcerqpIYDXdKf5bcc4FfL-5-WRRiTpaN58zU2E6M4; expires=Sat, 08-Oct-2022 07:41:56 GMT; path=/; domain=.google.com;
 Secure; HttpOnly; SameSite=lax
Set-Cookie: NID=511=Ls8nOqzwXThtdEjQqS8Ji4dl3-kpT78gUR5cFAw2XAdtUYcyLzWy6oTAP5ls-UMlqOZsjONMcBKIEMaqpykwPQr_XSunVnL9HxFedK3sOAL205fkKfTjSUnJFnG
YCJdyy8vA_hVQFLXqkA_8FQlPwM15zc_MF691nKEY9PtFtCg; expires=Tue, 11-Oct-2022 07:41:56 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

4f21
<!doctype html><html dir="rtl" itemscope="" itemtype="http://schema.org/WebPage" lang="iw"><head><meta content
  ~/pgit/NetworkSecurityHwSp22  on  main ?3 ·········································· at 10:41:56
❯
```

2.3. **Write Python** `server.py`**.** Screenshot of execution result:



Upload file

Instructions

TLDR:
Upload just the assignment.zip to test it. Or upload test.in and test.out files to validate it against previous assignments.

This upload has 2 functionalities.
1. Testing your assignment against the existing tests:
Select the zip file of the assignment (the very same you would send in the webcourse) and click on "Upload"
2. Submitting a new test (will be verified against all previously submitted assignments):
Select two files to upload (.in and .out files with identical base names. e.g. test.in and test.out).
You can select multiple files in the upload dialog by holding the Ctrl (or CMD for Mac) key while clicking on the second file).

When uploading an assignment, if the assignment it successfully running on all the tests, the compiled executable will be saved to run against new test files.
When uploading a test, if it successfully runs on all the existing assignments, it will be saved to run against new assignments.

What's New

1. (06/04/2022 13:00) I've allowed uploading multiple tests at once. Simply select the tests you'd like to upload (it pairs .in/.out according to the base file name)
All the uploaded files must be part of a pair (either the .in or .out). You can't send extra files or none of the tests will run.
Tests that ran successfully on all the assignments will be saved. Tests that ran on more than 80% but not all, will be help for manual review. That that failed on 20% or more than the assignment will be rejected.
If a test was rejected and you're **certain** it's a valid test, you can contact me: 03-511-2432.

Load File:

Choose Files No file chosen

Upload

Download Tests

General
Request URL: http://127.0.0.1:51966/
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:51966
Referrer Policy: strict-origin-when-cross-origi

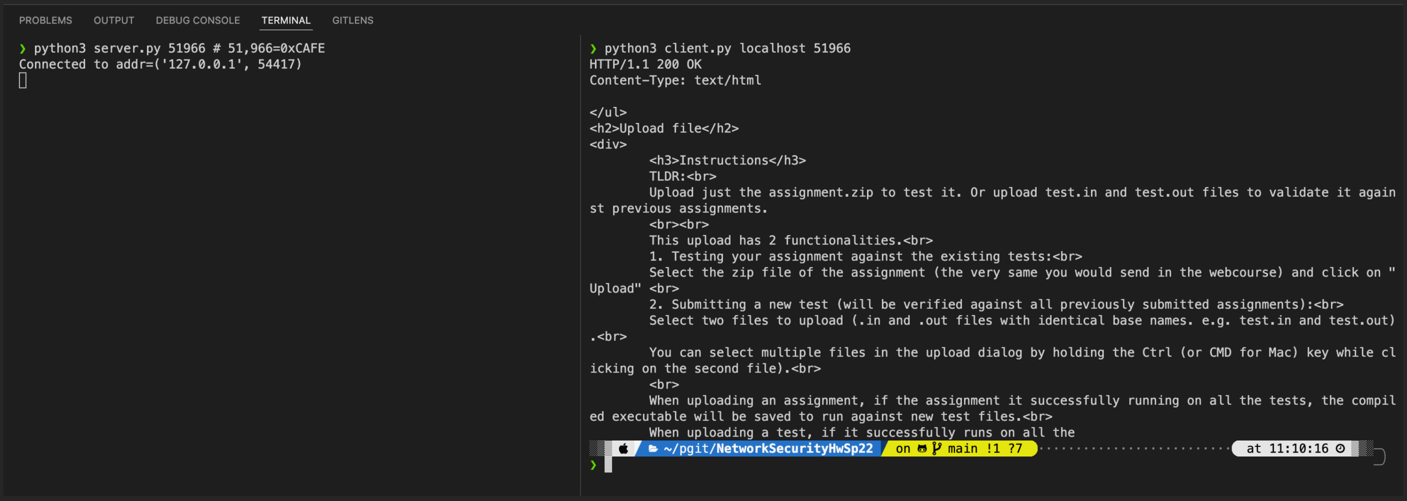
Response Headers (1)

Request Headers View source
Accept: text/html,application/xhtml+xml,appli
ge/webp,image/apng,*/*;q=0.8,application/sig
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,he-IL;q=0.8,
Cache-Control: max-age=0
Connection: keep-alive
DNT: 1
Host: 127.0.0.1:51966

2.4. **Run** `client.py` **on a running** `server.py`**.** Lines I ran:

```
python3 server.py 51966 # 51,966=0xCAFE
python3 client.py localhost 51966
```

Screenshot of the execution:



```
❯ python3 server.py 51966 # 51,966=0xCAFE
Connected to addr=('127.0.0.1', 54417)
```

```
❯ python3 client.py localhost 51966
HTTP/1.1 200 OK
Content-Type: text/html

</ul>
<h2>Upload file</h2>
<div>
        <h3>Instructions</h3>
        TLDR:<br>
        Upload just the assignment.zip to test it. Or upload test.in and test.out files to validate it again
st previous assignments.
        <br><br>
        This upload has 2 functionalities.<br>
        1. Testing your assignment against the existing tests:<br>
        Select the zip file of the assignment (the very same you would send in the webcourse) and click on "
Upload" <br>
        2. Submitting a new test (will be verified against all previously submitted assignments):<br>
        Select two files to upload (.in and .out files with identical base names. e.g. test.in and test.out)
.<br>
        You can select multiple files in the upload dialog by holding the Ctrl (or CMD for Mac) key while cl
icking on the second file).<br>
        <br>
        When uploading an assignment, if the assignment it successfully running on all the tests, the compil
ed executable will be saved to run against new test files.<br>
        When uploading a test, if it successfully runs on all the
 ~/pgit/NetworkSecurityHwSp22  on  main !1 ?7 ························· at 11:10:16
❯
```

2.5. **Socket APIs.**

(1) bind - Binds a socket to a an address and port. I used it after creating the socket before I started listening in `server.py`.
(2) listen - Starts listening to the bound port. The OS will do handshakes and complete init of connections after this line. I used it after binding the port. I used it after I used the bind in `server.py`.
(3) connect - Sends a syn request to a server to the given hostname and port. I used it to initiate a connection between my `client.py`
(4) accept - Getting a connection from the OS. I used it to accept a connection from a specific user (in a `while` loop) in my `server.py`.
(5) send - Sends data over an open connection. I used it (actually it was `sendall`) to send data to the other side of the connection. Both in the `server.py` and `client.py`
(6) recv - Receives data over an open connection. I used it in the `client.py` to read the response from the server. In the `server.py` I didn't use it because I don't care that my users have to say.

3. Social Engineering

3.1.