Computer Networks | 2020

HTTP Packet Stealer

March 2020 | Deadline: 16th, April 11:55 , Submission Form here

Changelog (Please check this out whenever you open the document)

Changes are highlighted in yellow.

- Late policy changed to 80% [8/4]

First things first, Academic Honesty ⚖

(Don’t skip this section please, some points may not seem like cheating but in fact they are, so read on.)

NOTE We really would like to remove this section about cheating forever, it’s just for the very tiny percentage of you who want to take things easy and get free marks. This section is added to the document because when one of that tiny fraction cheats, we can tell them that you were informed by the consequences. It’s also helpful to illustrate what’s reasonable because we aren’t normally familiar with some of those items.

You should do the "coding" totally on your own, so you won't be at a level that's lower than your peers when you join a company or do whatever after you graduate.

Sometimes it's not clear what's reasonable and what's not, so here is some guidance based on CS50's Academic Honesty guidelines.

What you’re basically asked to do is to convert English sentences to code. Accordingly, the below guidelines must be followed.

Cheaters ***will* get negative coursework grades for the project;** for example, let’s say that student X initially has 15 coursework grades, they cheated in this project and their mark was 10/10, after applying the cheating punishment, student X will end up with 5 marks as their coursework grade.

Delivering an incomplete project will be much better than cheating, the lowest possible grade without cheating will be more than the highest possible grade with cheating.

Reasonable ✔️

- **Discussing** the project with others in order to understand it better.
- **Helping** a classmate identify a bug in his or her code even online, as by viewing, compiling, or running his or her code, even on your own computer.
- **Incorporating a few lines of code that you find online** or elsewhere into your own code, provided that those lines are not themselves solutions to assigned problems and that you cite the lines' origins.
- Looking up the documentation of the modules you’ll be using, e.g.
    - How to open a or read/write to socket.
    - How to make a new thread.
    - How to use the select function.
- Using stackoverflow to solve the problems you’re facing with the modules.
- Reading the RFC or its explanation online or whatever document that does NOT contain code about the protocol.

Not Reasonable 😑

- Accessing a solution to some problem prior to submitting your own.
  Asking a classmate to see his or her solution to a problem before submitting your own. Following a tutorial that solves the project you're doing is prohibited as it violates this point.
- Any google search including raw socket\ packet stealer(or whatever equivalent word) **and you expect to get code in the results**.
- Following an online tutorial on how to make the project is cheating, since you’re basically copying the code **even if you understand every single line in that code**. Understanding a book about networks has nothing to do with being able to write such book

Why do you want to announce cheater names? Why don’t you keep their privacy?!

That’s a valid question. 1) Assume that I want to cheat, if I know that everybody will know that I’m a cheater and will look down on me, (logically speaking), I won’t think about cheating.

2) Cheaters have no privacy whatsoever, that’s because the TAs are there to help and answer questions and they could’ve asked their classmates for clarifications also. Issuing even an “arbitrary punishment” for an “unjustified” action isn’t a "wrong” thing.

3) Cheaters cause a great deal of unfairness to people who work ethically, this itself is a great crime, let alone the action of cheating itself.

Note if you’re not cheating (which we assume is your default case), you shouldn’t be worried about what's written above. There’s no need to ask yourself *if I do X will I be cheating?* If X isn’t on the list (and doesn’t violate the essence of the above list), feel free to do it. ✌

Intro 🚪

We've been using sockets in the "nice" way for long enough. Let's do something evil this time 😈. Sockets give us shear amounts of power, since in some sense you can control the behaviour of the process on the other end of the socket. With great power comes great responsibility, We're learning some evil stuff to be aware of them and to find techniques to mitigate vulnerabilities.

One other aspect that you might have noticed that everything is strictly defined and network elements are dumb, manipulating the packets on the network can let you for example impersonate some PCs, fake some IPs or even disconnect others from WiFi.

In the previous lab we made an HTTP Proxy which forwarded clients' requests to their destination. As you all noticed all of us were able to read the HTTP request as it was plaintext, but what if that client was logging into their bank account or their facebook account? Would we be able to steal their passwords? The answer is YES, it was possible back then when HTTP was the mainstream internet protocol. Nowadays we have HTTPS where all of the request packets are encrypted. That doesn't mean that nobody can check out your data, but it narrows down that number so much.

In this lab. We'll steal some HTTP information by making a program that listens to TCP packets then steals the data if it's an HTTP request and displays that request.

Objective 🔍

We'll learn how to use the third and final type of sockets, which is `SOCK_RAW`, this type of sockets access to everything as the operating system, depending on its address family.

By the end of this lab, you'll have a much deeper understanding of how network packets look like and you'll learn how to parse IP protocol network packets that are in L3 in the TCP/IP model. (note that this is not the OSI model).

You should also have the feeling of the level of accuracy required in code that's related to networks. Since all the parties involved in a networked system are not known to each other, everything needs to be specified in a very detailed manner.

Estimated Effort 🏃

This lab is so simple when it comes to coding, I tried a scratch working solution which took less than *60 lines of code* (I didn't use anything advanced since it's just packet parsing).

However, **lots of** care needs to be taken when writing this code, since you're now dealing with things on the *bit* level. Being not cautious / sleep deprived will increase the required time to complete the project significantly. For example, when writing those 60 lines I got stuck for about 40 minutes in some part of the code as I didn't handle some variable correctly. Things will be sorted out in the skeleton to reduce the chance of this to happen anyways.

You can finish this project in 6-7 hours. If you follow the mentioned precautions, you should be spending most of your time

1) Learning about bitwise operations & array slicing in Python and reviewing the "struct" module
2) Reading the *header format* (only, you never need to read the full RFC) sections in the RFCs.

On the other hand, if you don't follow the precautions this project can needlessly take much more work, due to the large number of locations that can make bugs occur. If you ever get stuck, stop coding and return to it later; the amount of logic required to make the code work is **minimal**.

Team 👫

The project will have teams of **at most two,** both team members will get the same grade. Please make sure you read & code **together** to reduce the probability of having bugs.

ℹ Requirements

Operating System: Linux

You can find a **video** tutorial that relates to the project here (the sound is a bit loud, feel free to lower it from the player)

You're asked to implement a Wirehsark-ish packet stealer that will track HTTP requests/replies and print them to console using RAW AF_INET sockets.

Using the sockets as illustrated in the video will allow you to access IP packets.

1) You'll need to parse IP packets to extract their data.
2) After that if they hold a TCP packet
3) you'll check if that data can be decoded as utf-8 characters or not.
4) If so, you'll consider this as an HTTP packet (regardless of its validity)
5) then print it directly to the console.

Parsing the packets requires that you read the header sections in the relevant IP and TCP RFCs. Be informed that both headers are of variable length and make sure you understand how this happens in order to parse the packets correctly. We're using IPv4 only.

Notes

- IHL in IP header and Data Offset in TCP header fields need special care.
- Please don't parse all the packet fields; they're not needed. Only those in the skeleton are needed.
- Building code based on simple test cases isn't correct. Build code based on knowledge from the RFCs instead.
- This project requires a *fair amount* of knowledge with bitwise operators, slicing and binary. Kindly check the resources section.

Checklist

> Your code can parse a byte literal containing an IP [10%]

> Your code has sockets set up and can capture TCP packets **ONLY** [10%] (capturing other types of packets will get a penalty)

> Your code extracts the IHL field correctly from the IP header [20%]

> Your code extracts source and destination IPs from the IP header and extracts the payload [10%]

> Your code extracts TCP source and destination ports [15%]

> Your code extracts TCP data offset [20%]

> Your code extracts TCP payload and prints it [15%]

The `parse_application_layer_packet` will always return the parsed packet and never `None`

Testing your code

To start, you can use netcat + TCP (send normal text) and make sure everything is working. You can learn more about netcat in this video.

Then, you can use the simple HTTP server provided by Python to avoid getting huge responses, you can launch that server in a directory that preferably contains a simple "`index.html`" file like so.

```
python -m http.server
```

You can use curl to make HTTP requests without typing them everytime. It's also very useful when using APIs to test things out.

To send a GET request using curl

```
curl -X GET http://127.0.0.1:8000/   # make a GET request to the default python server URI
```

And for sure, you should always check that your parsing matches the one done by Wireshark.

You can select a packet -> right click -> copy -> as Hex stream. Then you'll need to remove the Link Layer bytes manually then you can use it normally. You can convert a hex stream to bytes using `binascii.unhexlify()` then compare the results.

Bonus 🥇

Early Submission

Submission in the **first** 36 hours gets 15%, the **first** 72 hours gets 10% of their original grade. (measured from 12:00PM, 7th April)

"Hacker Edition" Bonus

If you read things carefully, you might have noticed in the manual page that SOCK_RAW from AF_INET sockets can "send" packets too.

This bonus is about exploiting this feature but using UDP protocol. Note that this code will be in a separate file since it needs a from-scratch setup that's different from the main project.

Your objective will be to craft IP packets to be sent by the raw socket and will be received by netcat. I.e. You'll **manually** set the UDP packet header and send it to a netcat udp server listening on some port (as in the video) and observe that it arrives on netcat without even opening a UDP socket, but using a RAW socket. This is by no means a simple project and will require a fair amount of both reading and debugging.

This should serve as a very light introduction to how a network stack works in an operating system.

This project will have a **separate submission form**. #TODO

To launch the code use the following command line arguments.

python [id1]_[id2]_bonus TARGET_IP TARGET_PORT MESSAGE

Submission 🛵

> Deadline: 16th, April, 11:55PM 🍿 .

> Submission Form here

> Code skeleton here

You must submit a **single** file. Either **code or an archive** (depending on your language choice).

You must prefix the file name with the lowest team member then the higher ID (sorted in ascending order), example: `[id_low]_[id_high]_lab3` and can have `.py` or `.zip` or `.rar` depending on what you're submitting. For example; if your IDs are 4112 and 3297 the file name should be `3297_4112_lab3.py` and NOT `4112_3297_lab3.py`. This helps us with the grading process and makes it more organized.

Notes

- If you submitted early. You won't be able to request resubmission.
- If it's the final submission day, you won't be able to request resubmission.

If you're late, multiply by zero point eight (80%)

You can submit up to 3 days after the project deadline, but you'll get only 80% of the marks. Late submission form is #TODO
Please don't abuse that, the best engineers are always in time. But sometimes it's the case that things don't go well, we understand that.

Were Google and Stackoverflow an illusion?

Sorry, but Yes. You might have noticed that stackoverflow and Google aren't your friends anymore when you're doing this project. This is normal, the most help exists for newbies since they're way more in numbers and almost anybody can help them.

When you get to advanced levels docs become your new friends and if you go even further only the truth will be your friend, which is the source code of what you're using. This is just a heads up to get your expectations settled correctly. Accordingly, you'll be doing yourself a good favor by reading as much docs as possible to get yourself used to the amount of patience and reading skills needed to adapt with your new friends.

Resources ✨

How to fight coronavirus, for software engineers here. 😷 🤦‍♀️

Man pages

- socket(2)
- socket(7)
- packet(7)
- All man pages
- What do the numbers 2 & 7 mean?

Python

- Bitwise operators tutorial here
- Slicing tutorial here
- try/except here
- SimpleHTTPServer module here
- What are byte literals? Here [**important**]

RFCs

- IP RFC here
- TCP RFC here

Tools

- Netcat tutorial here
- cURL tutorial here

For explorers 🔍

- What difference would it make to use AF_PACKET instead of AF_INET?
- Are you able to send packets to localhost if you use AF_PACKET? Why?
- Python's C FFI (foreign function interface) here, btw this is a universal thing.

TODO

- Specify where to throw or ignore the packet
- Compute checksum for packets
- Add tests to compute checksum
- Don’t import any external modules
- Tell them no to mess up function names