CSE 484 / M 584 Spring 2020: Lab #3

Smart Homes Security Lab

Due: ~~Friday, June 5~~, Saturday, June 6, 11:59pm
Turn in: Canvas assignment
Individual or group: Groups of up to 3 (as in prior labs)

Before you start

For this lab, you do not need to fill out a Google form, but you need to create an account at https://cse484.cs.washington.edu/nidan/signup to sign up. **You need to use your @uw.edu email and your UW Net ID's. No other email addresses are allowed. One account per group -- just choose one of your UW email addresses.** While we have taken basic measures to secure the password you choose and we do not see it in plain text, we still recommend you avoid reusing any real password from your other web accounts.

Additionally, portions of this lab require you to proxy your web traffic through a UW server, such as `attu`. We recommend you dedicate Mozilla Firefox for this purpose, set up the proxy on it, and only use that browser for the entirety of the lab. If something isn't working, we recommend first checking whether this proxy has been set up correctly. See the section "Helpful Tools and Setup" below for instructions on how to set up the proxy. Other useful tips are below too; **we recommend reading the whole lab description before starting, and scanning back through it whenever you get stuck.**

Warning

During this lab, you will be allowed to search for and find real devices on the real Internet that may or may not control real cyber-physical systems. Some of those may have real exploitable vulnerabilities. **You are not to interact with real devices or use the search feature in a malicious way.**

The devices you are supposed to be interacting with for the lab will be identifiable as belonging to a UW organization and will have an ipv6 address. If in doubt, ask the TAs but do not attempt to exploit or circumvent the security of any "real" devices you find through nidan.

Finally, security vulnerabilities in the implementation of the lab are specifically out of scope for this lab. :)

Server Address

https://cse484.cs.washington.edu

What To Turn In

We will record your progress in our database when you have achieved each of the 5 checkpoints in the lab and you can see that progress reflected on your *Profile* page (*You have compromised X out of 5 devices.*). However, you are also required to submit a **pdf or txt** to the Canvas assignment that, for each device:

- Provides a one-sentence summary of what the device was and how it was vulnerable.
- Provides the command you ran or steps you took to compromise the device.
- Describes how that vulnerability can be mitigated.

Please note that there is an opportunity for partial credit here: if you can't compromise a device, please still submit what you tried and why you thought it might work.

Goal

The goal of this lab is to gain hands-on experience with penetration testing of Internet of Things (IoT) devices as they might be deployed in a smart home. You will need to compromise a sequence of such devices in order to achieve an adversarial goal. **In this case, your end goal is to cause a (simulated) fire in a smart home with a series of IoT devices by turning on a (simulated) smart microwave over the Internet and without physical access.**

The technical exploits you need to carry out are not as sophisticated as in previous labs. They do not require advanced knowledge beyond a simple understanding of how the web and computer networks work. Thus, the key to success in this lab is the ability to sift through technical specifications to look for vulnerabilities and combine knowledge from different areas of computer science in order to achieve your goals. These are skills that real adversaries use :)

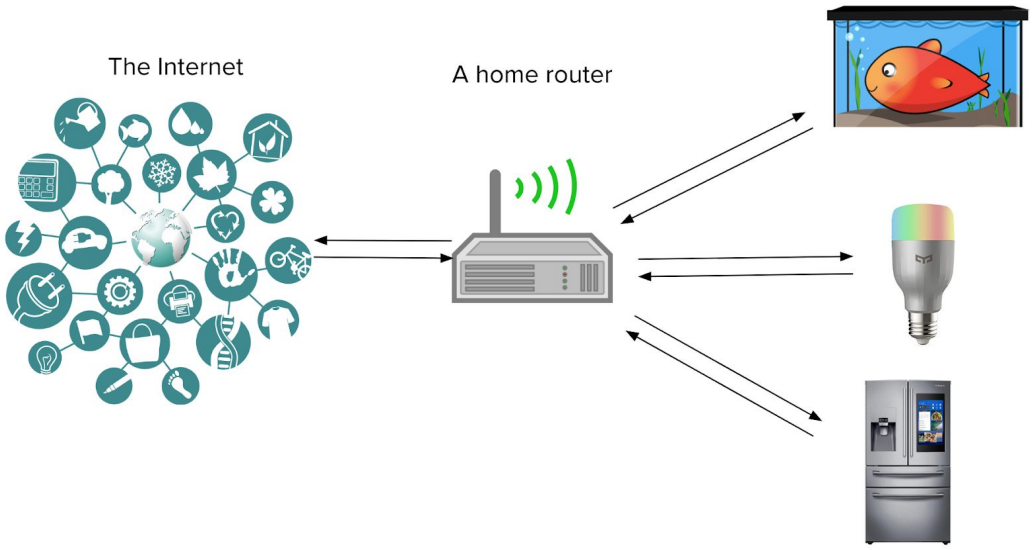
Points

You can receive a total of 4 points per device, for a possible total of 20 points. The points are distributed as follows:

- 1 point for compromising the device
- 1 point for providing a summary of the vulnerability (in the writeup)
- 1 point for providing the command you used to compromise the device (in the writeup)
- 1 point for a short description of how the vulnerability can be mitigated (in the writeup)

Some Useful Background

What does a smart home network look like?



The router “hides” all devices behind the NAT:

- Only the router has an external IP.
- Requests from devices to the Internet get translated to appear as if they are coming from a high port on the router.

Port forwarding:

- Devices (through UPnP) or the user (through the router command interface) can request that they be exposed permanently to the Internet.
- In this case, the router allocates a persistent port and forwards all connections on that port to the appropriate device (useful to allow external control).

Chaining vulnerabilities

- Even one device with a security vulnerability can serve as an entry point to compromising the entire home.
    - Target Hackers Broke In Via HVAC Company (Krebs on Security)
    - How a Fish Tank Helped Hack a Casino (The Washington Post)
- You do not even need buffer overflows, web exploits or other highly technical vulnerabilities. Some very common pitfalls (some may appear in this lab!):
    - Default credentials
    - Trusting that the device is only accessible on a LAN
    - Failure to authenticate physical-channel commands (e.g., audio)
    - Authorization tokens embedded in the open-sourced code of a controlling app
    - Fallacies in programming automation rules

Backstory

Canvas Notification:

Hey Class! Come one, come all: The Amazon Smart Life Roadshow is rolling into town! This fascinating, yet mildly terrifying display of latest and greatest Internet-of-Things (IoT) devices will totally ~blow your mind.~ Bring your friends and your wallets and join your course TAs in exploring how to retrofit your old-fashioned homes with a brand new Smart Home ecosystem!

News Alert:

With all the force of a great typhoon, Shodan has taken the world by storm. Shodan, or shodan.io, is the premier search engine for IoT devices. Just as Google lets people sift through endless streams of funny cat videos, Shodan returns pages of results containing internet-connected devices: construction equipment, traffic lights, power plants, and baby monitors halfway around the world… as well as their *command interfaces*. Stay tuned to see how some Shodan hackers took over Seattle's construction cranes and programmed them to do the macarena!

Canvas Notification – Revision

So… the TAs went to the Amazon Smart Life Roadshow… and due to our salary constraints, we have decided to build our own set of totally-legit, totally-secure Smart Devices! Why give your money to Amazon, when you can give your money to us and use our totally-private Husky-branded IoT devices! Just think: HuskyCam, HuskyRouter, HuskySpeaker, and so on. If our system gains enough popularity, maybe the University will agree to deploy it to livestream New Dubs frolicking in his playpen and stealing Officially Retired Dubs' HuskyTreats!

News Alert:

Mysterious as the dark side of the moon, Nidan enters the scene of IoT search engines… Nidan (cse484.cs.washington.edu) is a hand-crafted, artisanal, GMO-free platform devised by the Paul G. Allen School of Computer Science & Engineering Computer Security and Privacy Top Secret Developer Team (PGASCS&ECSPTSDT). Much like Shodan, it allows users to peruse real IoT devices but can also help to ethically facilitate a CSE484 homework assignment ;) After the commercial break, we ask experts why they think Nidan seems to be picking up on various ~wintery canine~ themed devices…

Specifications of the Husky Smart Home Technology

The Nidan Search Engine

The Nidan search engine -- accessible at https://cse484.cs.washington.edu/ -- provides a subset of the functionality of the shodan.io search engine. We recommend you familiarize yourself a bit with Shodan first. As students, you get some features for free if you sign up with your .edu email. In particular, you can access the shodan book for free (follow the link in the email you get after signing up for shodan with you .edu address to get the free version). You don't need to read this entire book, but you might find it helpful. If you have trouble accessing the book, please let us know. The following assumes you have a basic understanding of how the shodan search engine works.

The search functionality works as follows:

- A search query is made up of tokens separated by white spaces.
- Each token specifies an AND condition for the search (so the results are at the intersection of the sets matched by the queries).
- If a token does not contain a colon (the character `':'`), the term is interpreted as a keyword to look for in a non-case sensitive manner in the `data` field of a device.
- If a token does contain a colon (the character `':'`), there are two possibilities:
    - If the left-hand string does not begin with a minus (the character `'-'`), the token is interpreted as a filter. The left-hand side of the filter specifies the field to filter on and the right-hand side specifies the *exact* value that field should take on in any matched results.
    - If the left-hand string begins with a minus, the token is also interpreted as a filter but any devices matching the filter are excluded from the results.
- If a filter value contains white spaces, enclose it either in single or double quotes.
- The list of fields you can filter on are as follows (refer to the Shodan documentation on what they mean)
    - transport
    - timestamp
    - port
    - asn
    - ip
    - org
    - isp
    - os
    - city
    - region_code
    - area_code
    - longitude

    - country_code
    - latitude
    - postal_code
    - dma_code
    - country_name

For instance, to find all devices containing the word “apache” in their header response, listening at port 80, located in the United States but not in New York, you can use the query:

```
apache port:80 country_name:"United States" -city:"New York"
```

Hint: You will need to use Nidan as a starting point to find devices of interest that are publicly accessible. (Remember not to actually attack any non-lab devices you may find…)

Caution: The web app of Nidan itself and the database backing it are ***not*** exploit targets for this lab. We’d be happy to leave it up for you to play with and test your web security skills after the lab is over, but **do not** attack it during the lab.

The Husky Speaker

Due to budgetary constraints in the development of the Husky Speaker, it provides a deceptively simple, yet highly functional interface. It accepts `POST` requests with a single data field: `url`. The speaker then accesses the URL and plays the requested file.

The format specifications for the file are as follows:

- `flac` file format
- <500 KB in size
- single channel

Anything that does not match those specifications will be rejected by the speaker.

The Husky Voice Assistant

The Husky Voice Assistant is the most exciting new entrant in the world of Google Home, Cortana, Siri, and Alexa. It enables you to control a wide range of home devices simply with the power of your voice. Just say the word and the Husky Voice Assistant will get it done!

In order to enable full transparency, the assistant allows anybody on the home network to read what commands the speaker heard recently and how it reacted to them by accessing its web interface. Unlike with some *other* smart speakers, the Husky Voice Assistant will never creepily laugh at you without telling you why.

The HuskyCam Camera

This device amazes with its accessibility - you just need a simple username and password and you can watch your home from wherever you are in the world. Make sure that your cat is eating well while you are on vacation, check if the mail has come in, or just spy on other members of your household - now all a simple login away!

You will be able to choose one of nine easy-to-remember username/password combinations upon enrollment - and the best part is, you cannot change them and accidentally set a password you'll forget!

- ("root", "xc3511"),
- ("root", "vizxv"),
- ("root", "admin"),
- ("root", "888888"),
- ("root", "default"),
- ("admin", "123456"),
- ("admin", "password"),
- ("admin", "pass"),
- ("admin", "1111111")

Helpful Tools and Setup

Proxy through ssh

You might not need to do this, if you are ***not*** on the UW network ***and*** you already have an IPv6 address assigned to you by your ISP. You can Google "What is my ip?" to check these things. If you are connected through the UW network ***or*** do not have an IPv6 address, follow along.

MacOS and Linux

To set up an ssh proxy through a UW server, run the following command on your local computer:

```
ssh -D <portnumber> <csenetid>@attu.cs.washington.edu
```

where `<portnumber>` is the port on your localhost that you want to use for your proxy and `<csenetid>` is your CSE id that is authorized to access attu. For example, to set up a proxy on port 1080 that tunnels traffic through johndoe's account, run:

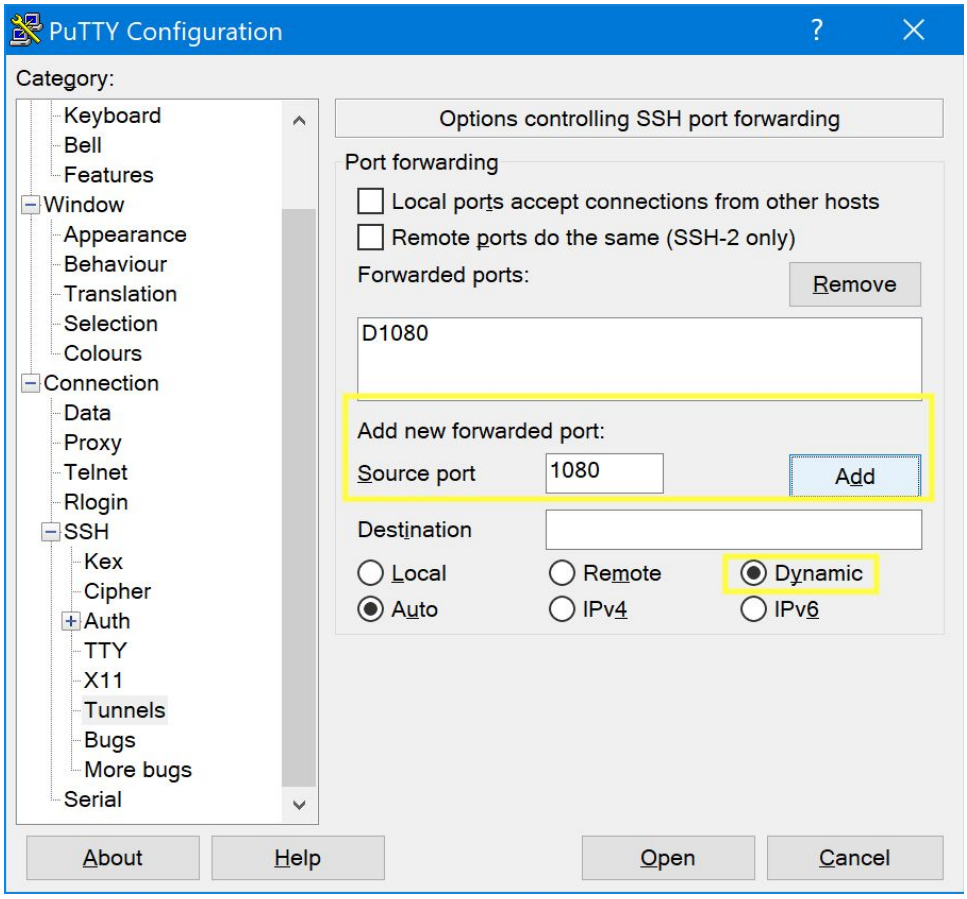
```
ssh -D 1080 johndoe@attu.cs.washington.edu
```

Note that this does ***not*** set up any program on your computer to ***use*** this proxy. It only makes it available; to make your browser and other tools use it, read on.

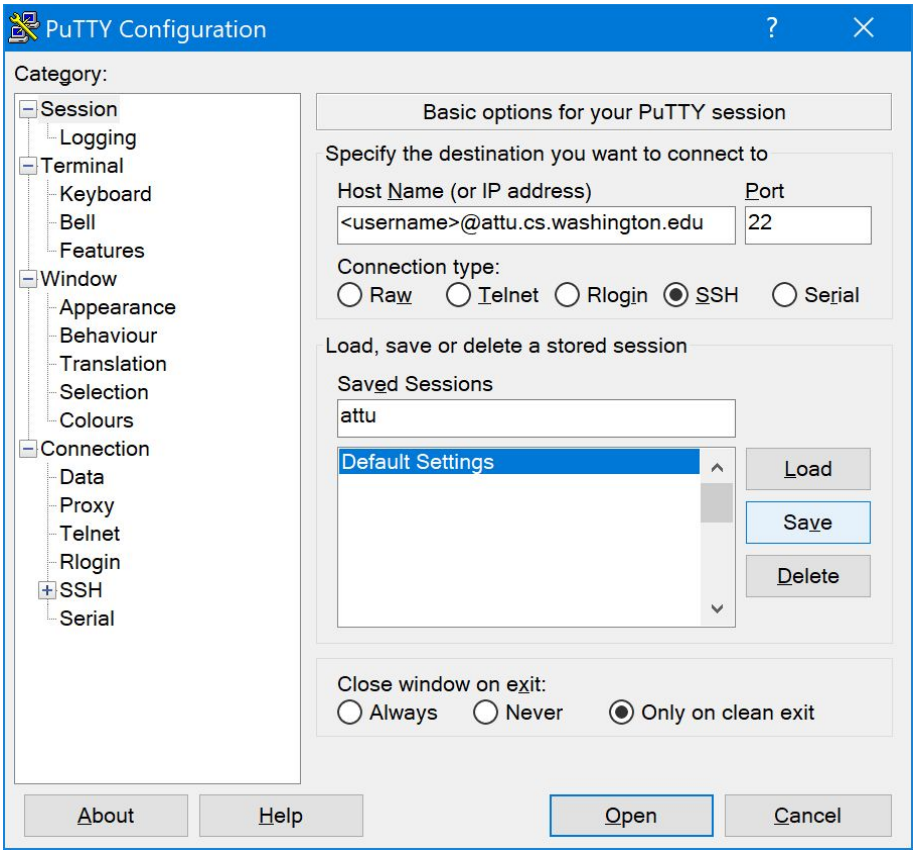
Windows

Follow these instructions to set up the proxy via PuTTY on Windows.

1. Go to **Connection > SSH > Tunnels**. Fill in **1080** for the **Source port** field. Then select **Dynamic** and click **Add**. You should see a line in the text box that reads **D1080** (or whatever number you chose).



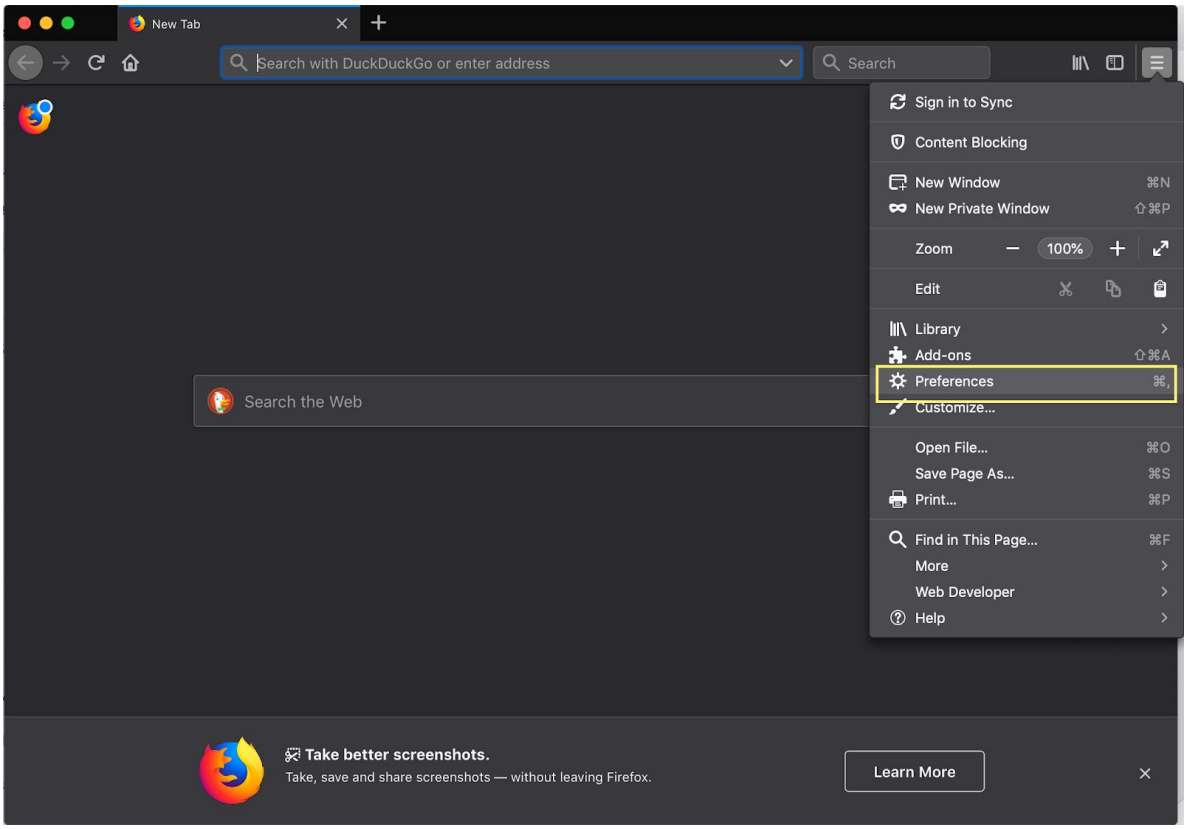
2. Go back to the **Session** tab, enter **<username>@attu.cs.washington.edu**, and optionally name the session as attu and click **Save** so you can load it the next time without having to explicitly re-enable the SOCKS proxy.



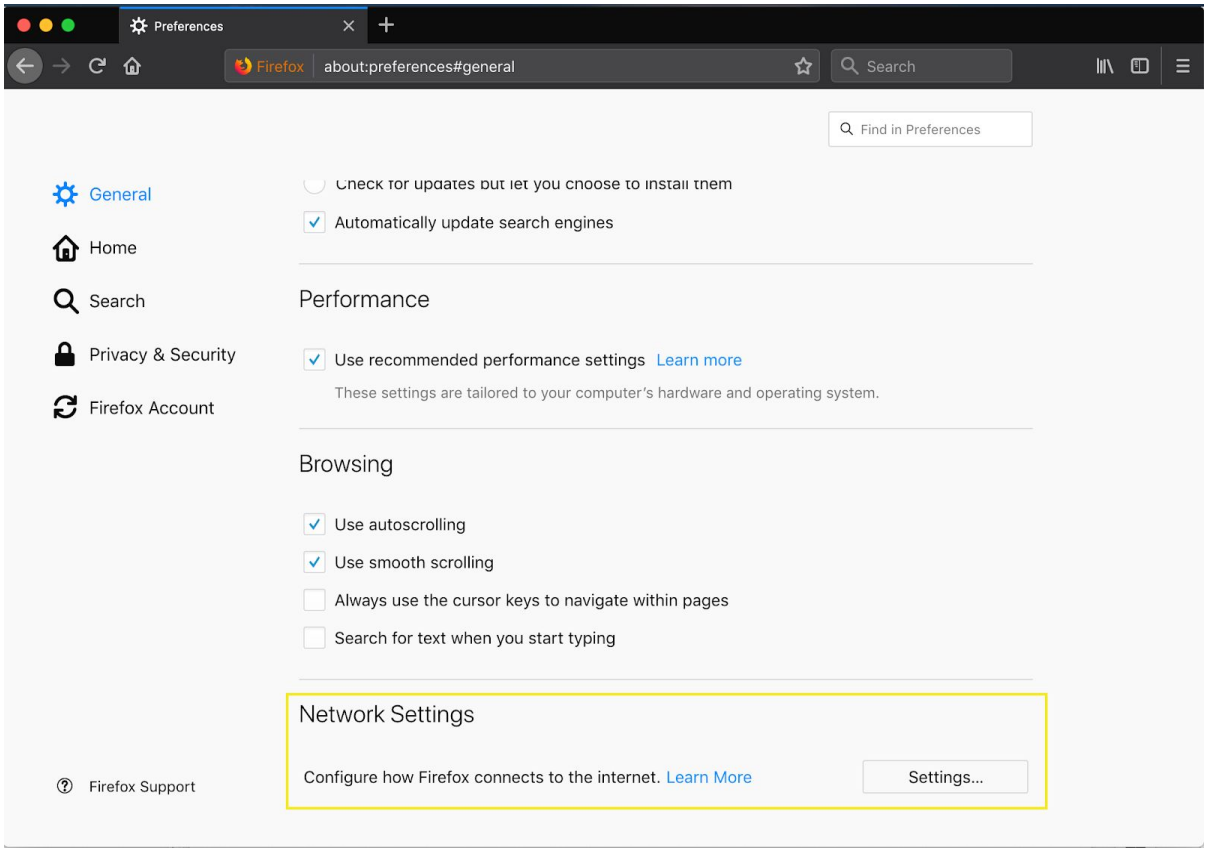
Browser

During the course of this lab, we recommend that you use Firefox. To set up Firefox to use the proxy you deployed above, follow these steps:

1. Click on the sandwich menu in the top right corner to expand the browser's menu. Then, select "Preferences."



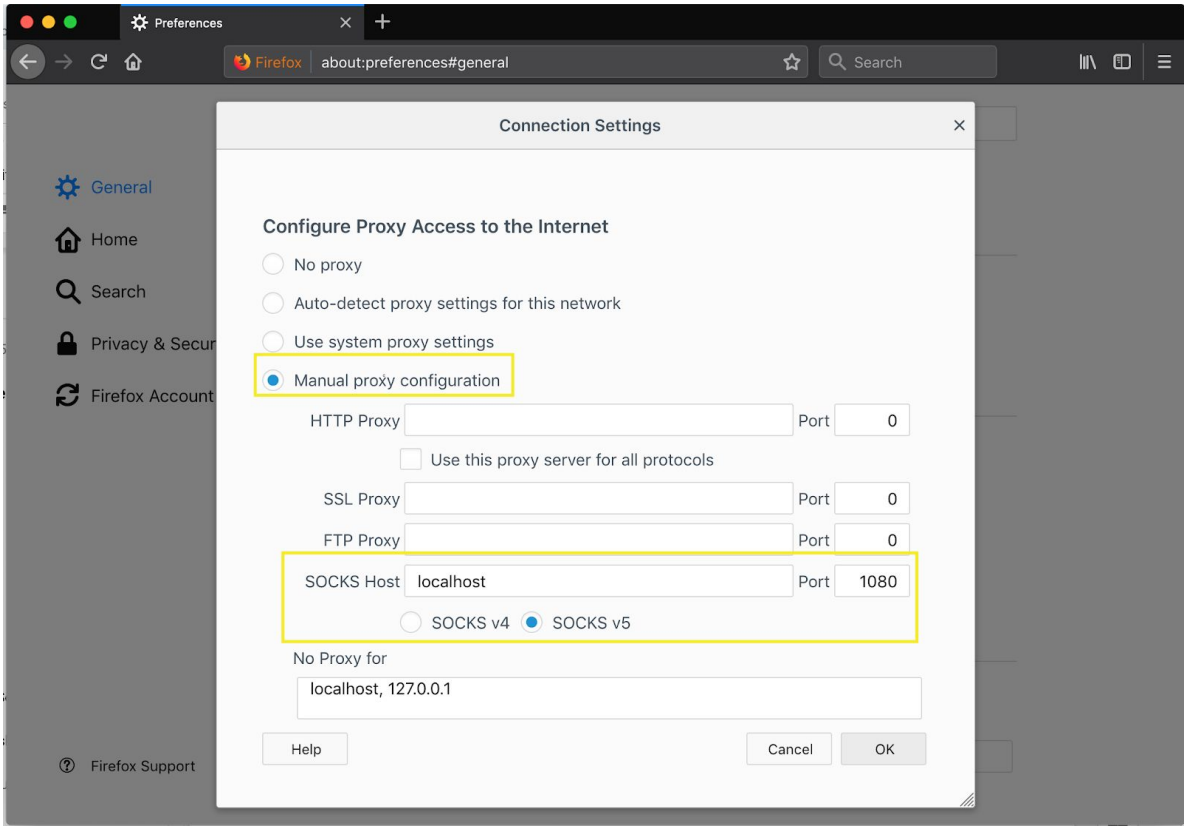
2. Under “General,” scroll down to “Network Settings” or use the search bar to search for “proxy”



3. Select the “Manual proxy configuration option” and fill out ***only*** the SOCKS Host settings as follows:

- type in `localhost` in the text field
- provide the port you used with the `ssh -D` command above, e.g. 1080

Make sure that all other fields (HTTP Proxy, SSL Proxy, FTP Proxy) are left blank.



Please, note: All of your Firefox traffic will now be tunneled through the UW server you picked in the `ssh -D` command.

If you have not run the `ssh -D` command or if you have stopped it, your browser won’t be able to access any web pages.

Additionally, with these settings, ***all*** of your web traffic on this browser, including for any non-course related websites, will be proxied through UW’s server. Please, follow the CSE computing use policy when browsing in this way or disable the proxy for any personal use.

Make sure you access the nidan site with **https://**.

Note for Windows Users:

If you have your Firefox and putty settings entered correctly, you can access the device HTML through the ssh proxy and `curl`, and you still have issues connecting to devices through Firefox, it may be a putty issue. We suggest:

1. Have one of your teammates who has a Unix installation already do the ssh tunnel and work from their machine
2. Install a VM that runs a Unix system
3. Install Windows Subsystem for Linux

The curl Command

For some portions of the lab, you may need to use the `curl` command and proxy its traffic as well. To run `curl` through a proxy, use the `--socks5` option, e.g.

```
curl example.com --socks5 localhost:1080
```

Other useful options for curl for this lab are `-X` (specify the HTTP request type) and `-d` (specify the data payload sent with the request). You can find out more in the man pages.

Dealing with IPv6 Addresses

IPv6 addresses, when typed into a browser or given to `curl`, need to be enclosed in square brackets (`[`, `]`). When you do that with curl, you also need to give it the `-g` option. For example, to access ip `2607:4000:200:15:1234:5670:abcd:ef12` at port `8000`, you need to type in:

- in your browser: `http://[2607:4000:200:15:1234:5670:abcd:ef12]:8000`
- in curl: `curl -g http://[2607:4000:200:15:1234:5670:abcd:ef12]:8000`

Audio Files

For a portion of this lab, you may need to record your own voice. You can do that with any software you are comfortable with. For Mac, the QuickTimePlayer is easy to use and readily available but it does not produce the needed format. For a cross-platform piece of software, you can use VLC (it also lets you do conversions). You may also find https://ffmpeg.org/ useful for conversions, if you prefer (but VLC should suffice).

You may host these audio files on any publicly accessible URL and you should already have experience from the previous lab with hosting files on your CSE home page. Treat the audio file as any other static file you'd like to serve from your home directory. If you are getting errors around the type of file you are submitting, you are probably on the right track, but read the specifications for the HuskySpeaker carefully.

FAQs and Errata

Please, monitor the EdStem discussion board. We will update this with frequently asked questions and lab bugs that you should be aware of.