HW5

Due Dec 8, 2020 by 11:59pm **Points** 100 **Submitting** a file upload
Available Nov 24, 2020 at 11:59pm - Dec 18, 2020 at 11:59pm 24 days

This assignment was locked Dec 18, 2020 at 11:59pm.

Smart Homes Security Lab

For this assignment you can form a group of up to 3 students.

For this lab, you need to create an account at **https://nidan.cs.wisc.edu/nidan/signup (https://nidan.cs.wisc.edu/nidan/signup)** to sign up. You need to use your **@wisc.edu** email and your **UW Net ID's**. No other email addresses are allowed. While we have taken basic measures to secure the password you choose --- like they are salted and hashed ---, please **do not use any real password from your other web accounts**.

Additionally, portions of this lab require you to proxy your web traffic through a UW-Madison machine such as *best-linux.cs.wisc.edu*. 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.

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-Madison 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 Nidan are specifically out of scope for this lab.

Please read through the following detailed sections carefully:

Assignment Structure

Server (Nidan):

The server you will be interacting with can be found at: **https://nidan.cs.wisc.edu/ (https://nidan.cs.wisc.edu/)**

Goal:

The goal of this lab is to gain hands-on experience with penetration testing of Internet of Things (IoT) devices deployed in a (virtual) 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 CS642 you have read until now in order to achieve your goals.

Deliverables:

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.)* as well. However, you are also required to **submit a writeup (i.e., solution.txt)** on the Canvas, which should have the following details for each smart device to gain full points.

For each device,

- Provide a brief summary of what the device was and how it was vulnerable.
- Provide the command you ran or steps you took to compromise the device.
- Describe 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.

At the top of solution.txt, please mention the team member's name and netid.

Points:

You can receive a total of 20 points per device, for a possible total of 100 points. The points are distributed as follows:

- 10 points for compromising the device
- 5 points for providing a summary of the vulnerability (in the writeup)
- 2.5 points for providing the command/steps you used to compromise the device (in the writeup)
- 2.5 points for a short description of how the vulnerability can be mitigated (in the writeup)

Side Note:

This lab has been inspired from **shodan.io (https://www.shodan.io/)**, which 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. We came up with our own IoT device search engine called *Nidan.* :)

Specifications of the Badger Smart Home Technology

The Nidan Search Engine

The Nidan search engine -- accessible at **https://nidan.cs.wisc.edu/ (https://nidan.cs.wisc.edu/)** -- provides a subset of the functionality of the shodan.io search engine. We recommend you familiarize yourself with Shodan first. As students, you get some features for free if you sign up with your .edu email. There are many tutorials on the web that explain shodan search syntax. The following assumes you have a basic understanding of how the Shodan search engine works (the following also applies to Nidan).

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: Similarly you can find smart devices which are of *interest* to us for the purpose of this assignment on Nidan.

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 BadgerCam 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!

The BadgerCam comes with a username/password combo preset for you. However, because the Badger company does not like to keep records, there is no way to be sure which of the following is actually your username/password combo for your device.

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

The BadgerSpeaker

Due to budgetary constraints in the development of the Badger 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 BadgerVoice Assistant

The Badger 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 Badger 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 Badger Voice Assistant will never creepily laugh at you without telling you why.

Helpful Tools and Setup

Please follow the following guidelines carefully:

How to Proxy through ssh

- **For MacOS and Linux**

To set up an ssh proxy through CSL lab computers, run the following command on your local computer:

```
ssh -D <portnumber> <csnetid>@best-linux.cs.wisc.edu
```

For example, to set up a proxy on port 1080 that tunnels traffic through johndoe's account, run:

```
ssh -D 1080 johndoe@best-linux.cs.wisc.edu
```

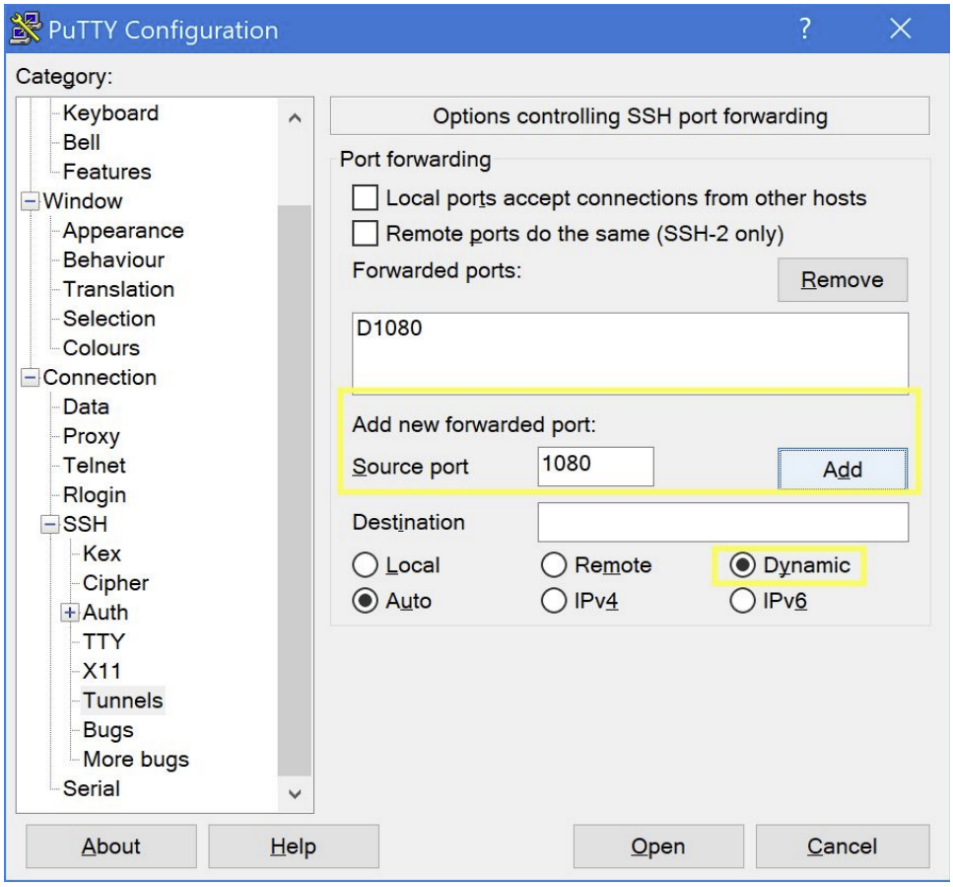
You can use any port number, but recommended is 1080.

We will use this ssh tunnel using our browser or other tools (**cURL (https://curl.haxx.se/docs/manpage.html)**), to connect with Nidan. Please read on.

- **For Windows**

Follow these instructions to set up the proxy via **PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)** 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**



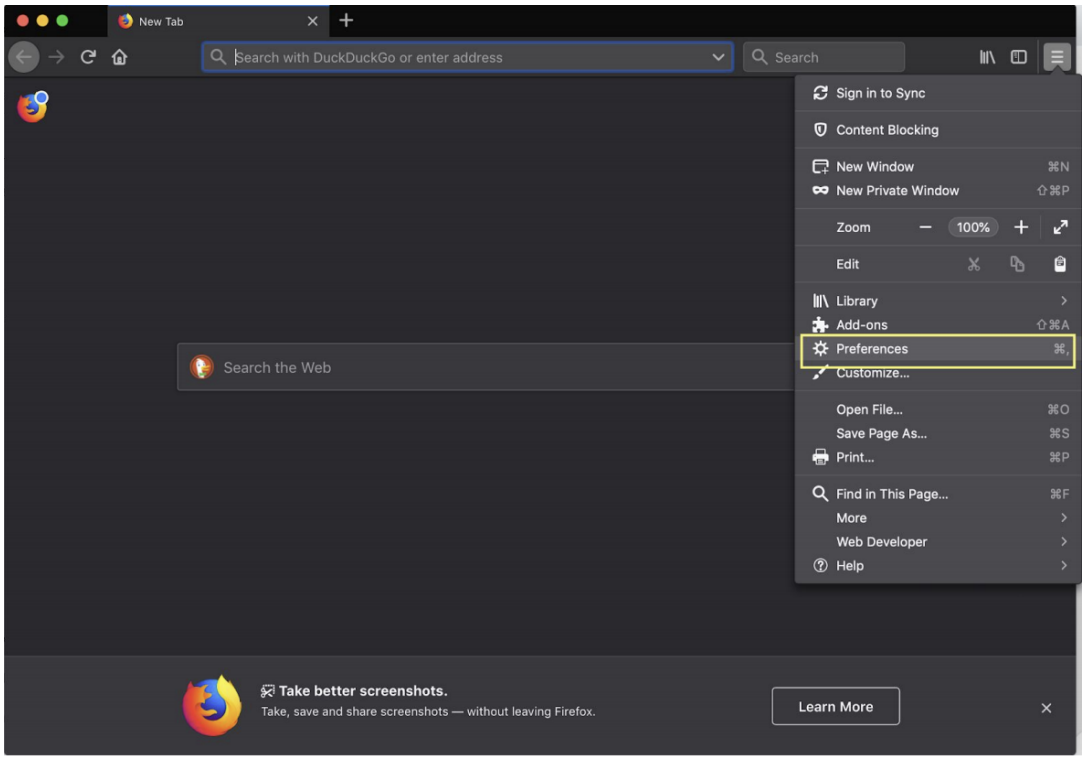
2. Go back to the **Session** tab, enter **<csnetid>@best-linux.cs.wisc.edu**, and optionally name the session as nidan and click **Save** so you can load it the next time without having to explicitly re-enable the SOCKS proxy.

How to access Nidan

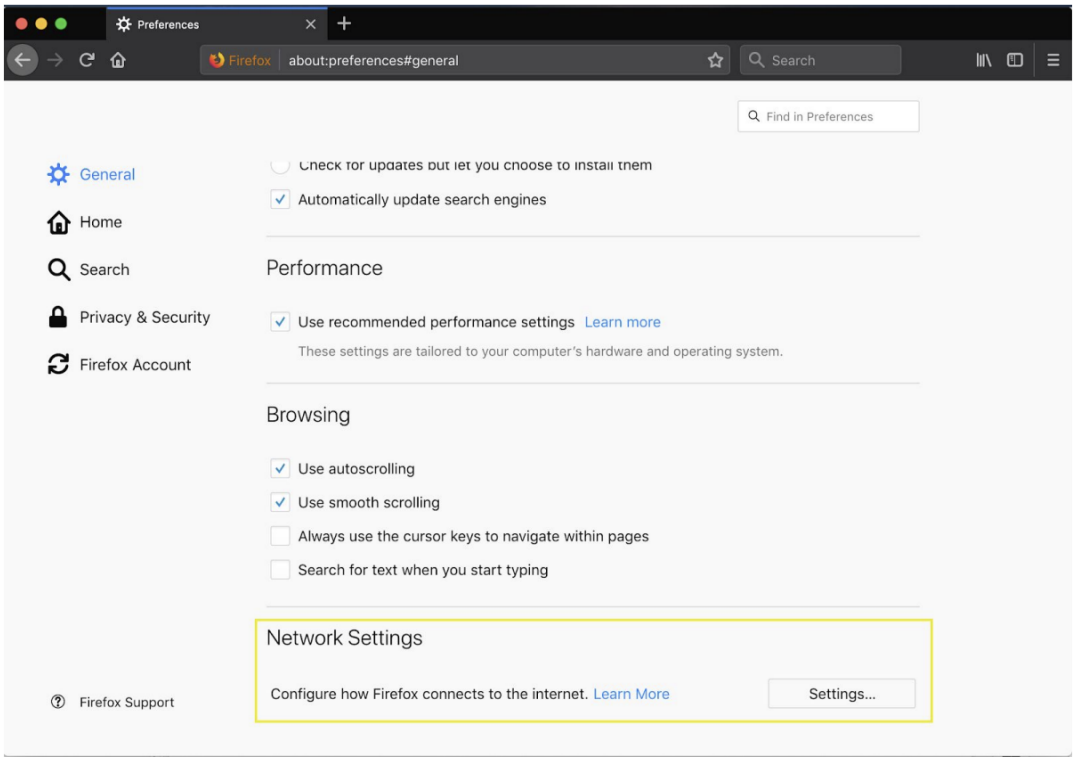
- **Using Browser:**

During the course of this lab, we recommend that you use **Firefox** (https://www.mozilla.org/en-US/firefox/new/) browser. To set up Firefox **to use the ssh 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"**

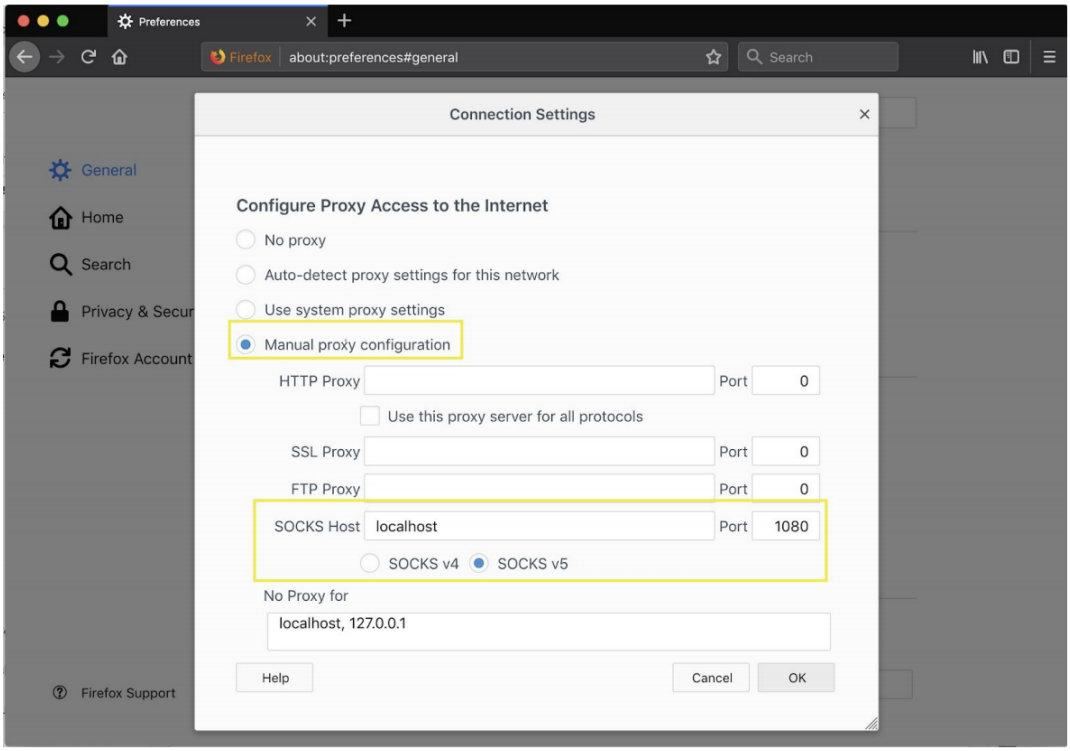


2. Under **“General”**, scroll down to **“Network Settings”** or use the search bar to search for **“proxy”**



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.



NOTE: All of your Firefox traffic will now be tunneled through the CSL machines, even your normal web traffic. 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. **Please, disable the proxy when you intend to use Firefox for personal use.**

- **Using cURL:**

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 (https://curl.haxx.se/docs/manpage.html)** .

Miscellaneous

- **Dealing with IPv6 Addresses**:

IPv6 addresses, when typed into a browser or given to curl, need to be enclosed in square brackets ([<ipv6addr>]). 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 (http://%5B2607:4000:200:15:1234:5670:abcd:ef12%5D:8000)***
  - in curl: *curl -g* ***http://[2607:4000:200:15:1234:5670:abcd:ef12]:8000 (http://%5B2607:4000:200:15:1234:5670:abcd:ef12%5D:8000)***

- **Audio Files:**

For a portion of this lab, you may need to record your own voice (and convert the file into *flac* format). You can do that with any software you are comfortable with. For a cross-platform piece of software, you can use **VLC** **(https://www.videolan.org/vlc/index.html)** (it also lets you do conversions). You may host these audio files on any publicly accessible URL (e.g **https://www.file.io/** **(https://www.file.io/)**). Treat the audio file as any other static file you'd like to serve from your home directory. Read the specifications for the BadgerSpeaker carefully.

Bugs and Questions

Please use Piazza to contact the Instruction staff if you encounter any bug when using Nidan and also for any follow up questions that you might have. Good luck!

HW5 Rubric		
Criteria	**Ratings**	**Pts**
Device 1 Compromise		20 pts
Device 2 Compromise		20 pts
Device 3 Compromise		20 pts
Device 4 Compromise		20 pts
Device 5 Compromise		20 pts
		Total Points: 100