CSci 4271W (011 and 012 Sections) Lab Instructions

Lab 10 **April 7th, 2025**

Ground Rules. You may choose to complete this lab in a group of up to three students. Before you leave the lab, **make sure you have submitted to Gradescope, you included all group members on the submission, and the autograder found all required files!**

1 mitmproxy

In today's lab, we'll see a useful tool for inspecting and modifying TLS-encrypted network connections, mitmproxy. This tool can be used for debugging TLS-enabled applications, injecting or modifying traffic in TLS-encrypted flows, or reverse-engineering encrypted network applications. Before we install and see a simple demo of using mitmproxy, read this short explanation of how mitmproxy works.

2 Installing `mitmproxy`

For today's lab, you'll want to have multiple terminals open to your VM. We'll be doing some editing of text files on the VM, so unless you have a terminal-based text editor you know and like (e.g., `emacs -nw`, `vim` or `nano`) you might want to use x-forwarding when you ssh to your VM, e.g. `ssh -X student@csel-xsme-s25-csci4271-NNN`. Once you've logged in to your VM, we're going to have to follow a series of steps to get mitmproxy installed.

2.1 Install PyPI, pipx and virtualenv

mitmproxy is written in Python, so a convenient way to install it is using the Python package manager (PyPI, or `pip` on the command line), which we install and set up first:

```
$ sudo apt install python3-pip python3-venv pipx
$ pipx ensurepath
$ source ~/.bashrc
```

(The final command `source ~/.bashrc` gets the updated PATH environment variable into your current shell. This can also be accomplished by logging out and logging back in or opening a new terminal, as the message suggests.)

2.2 *Finally*, we can install mitmproxy

Use pipx to install mitmproxy:

```
$ pipx install mitmproxy
```

When I did this I got fireworks emojis, fun!

3 Running mitmproxy

In your VM terminal, you can now run mitmproxy. Since our VMs *also* need proxy access to reach the external internet, we'll run in "upstream" mode, as follows:

```
$ mitmproxy --mode upstream:http://proxy.oit.umn.edu:3128/
```

You should see a text window interface that looks like it could have been at home in the 1980s. Leaving that be for now, let's switch over to another VM terminal and start generating some traffic. In a second terminal window, let's start by downloading the Wikipedia front page using wget:

```
$ wget https://wikipedia.org/
```

This should work without any issues; if you look back at the window running mitmproxy, you won't see any activity.

3.1 Intercepting a flow

Next let's see what happens if we redirect wget to mitmproxy, but don't install the mitmproxy CA:

```
$ https_proxy=http://localhost:8080/ wget https://wikipedia.org/
```

Running this command you should see a complaint from wget that it cannot verify "wikipedia.org's" certificate (issued by mitmproxy). If you switch over to the mitmproxy window quickly enough, you might see a Warning message at the bottom that the handshake failed (because wget bailed out when it couldn't verify the certificate.) So we need to either convince wget not to check the certificate or install the mitmproxy certificate on our VM. First let's do it the easy way, telling wget not to check the certificate:

```
$ https_proxy=http://localhost:8080/ wget --no-check-certificate https://wikipedia.org/
```

Success! wget downloaded the Wikipedia front page. If you switch over to the terminal running mitmproxy, you'll see two "Flows" in the main window pane. You can navigate the cursor between them using the up and down arrow keys, and choose a flow by hitting enter. Let's first try this with the earlier flow. If you press the key, you'll see three "Tabs" labeled "Request", "Response", and "Detail". You can navigate between the tabs using the left and right arrow keys. If you open the "Response" tab you'll see that when we requested "wikipedia.org", we were told to instead go grab "www.wikipedia.org." Quit out of this flow by hitting the "q" key, and navigate to the next flow to see what happened when wget downloaded "https://www.wikipedia.org/". Under the "Request" tab you'll see that Wikipedia set a "cookie" on the first request, which `wget` returned on the second, telling itself where we were originating from (`GeoIP=...`), and under the "Response" tab you can see the full details of the HTTP response that wikipedia sent, nicely decrypted for us!

3.2 Installing the mitmproxy CA cert

That's great, but surely only negligent developers (BCI employees?) would turn OFF a default-ON security control, so if we were trying to analyze a system that DID check certificates, we would need to convince the OS that mitmproxy was a valid CA. In order to do that, you'll need to copy the CA certificate that was generated by mitmproxy (on your VM) into the system database of root CAs. This can be accomplished by executing the following two commands in your VM terminal:

```
$ sudo openssl x509 -in ~/.mitmproxy/mitmproxy-ca-cert.pem -inform PEM -out
↪  /usr/local/share/ca-certificates/mitmproxy-ca-cert.crt
$ sudo update-ca-certificates
```

Once this is done, try using wget to download `wikipedia.org` again on your VM, this time with certificate checking enabled (the default):

```
$ https_proxy=http://localhost:8080/ wget https://wikipedia.org/
```

With mitmproxy installed as a root CA on your VM, `wget` won't complain and the download will succeed. Back in the mitmproxy terminal window, you should see two additional flows. Use the arrow keys to select the second flow (with the actual page). We can save the request and response from this flow using the `save.file` command; in the window showing the flow type `:save.file @focus wiki.log`. This will save the current flow (`@focus`) to a file named `wiki.log` (you could put any filename you like here). Once this is done, you can exit the flow by typing `q` and exit mitmproxy by typing `q` again. The `wiki.log` file will be your submission for this lab, so use `scp` to copy it off of your VM.

3.3 Other mitmproxy actions

Of course mitmproxy is useful because it can do a lot more than just intercept and decrypt TLS traffic; it can also be used to modify traffic according to regular expressions, target specific flows using hostname patterns and regular expressions, and support arbitrary python-scripted add-ons to perform more complicated manipulation of the data in a TLS-encrypted flow. You might find it interesting to browse the "Features", "Modes of operation", and "Options" tabs in the mitmproxy documentation.

3.4 All done!

Once you've saved the `wiki.log` file, you're done with Lab 10! Use `scp` to copy the `wiki.log` file off of your VM, so you can submit it to the Lab 10 assignment on Gradescope. Make sure you include all of the members of your group!

Once you've submitted the file, the autograder will test to make sure the proper file was submitted, check that it includes the right information, and notify you if anything went wrong, within a few minutes.

Congratulations, you've finished Lab 10!