PiOT  a small form factor defense for indefensible devices

James Leyte-Vidal

Copyright SANS Institute 2020. Author Retains Full Rights.

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.
PiOT – a small form factor defense for indefensible devices

GIAC (GCIA) Gold Certification

Author: James Leyte-Vidal, james.leyte.vidal@gmail.com
Advisor: Mohammed Haron
Accepted: July 31st, 2018

Template Version September 2014

Abstract

For several years, trending observed has shown the ever-increasing growth of network-connected ‘things’ – items like appliances, lighting, controllers and, others that have not typically been network connected in the past. This has resulted in a significant increase in attack surface in networks that connect these devices, as many of these ‘things’ have not been designed or implemented with security in mind. While the industry continues to work with these manufacturers to offer better, more secure alternatives, there are many devices out there today that present a risk. To combat this issue, and to help mitigate this risk, we present PioT. PioT is a RaspberryPi-based device intended to be placed in front of vulnerable IoT devices. In conjunction with traffic monitoring and logging tools, PioT is intended to be a robust, expandable platform for monitoring and responding to attempted access to vulnerable IoT devices. In this paper, we will outline the PioT build process and show the capability to observe access to an IoT device. The total cost for this build is less than $100.
1. Introduction

In the past five to ten years, the Information Technology industry has begun to face a new challenge - the rapid proliferation of new classes of network connected devices. These devices have become known collectively as the Internet of Things (IoT). While network connectivity for these devices is not in and of itself novel, these are devices that may have been in home or businesses for many years, and new models or versions are being shipped with network capabilities. Examples of these include refrigerators ("Overview," 2018), coffeemakers ("Mr. Coffee Smart," 2018), and Lighting control ("Hue Products," 2018). The intent is to enhance the functionality of the product or at least make that functionality available without standing directly in front of it. An unfortunate side effect of this additional availability is that these products are now subject to a more prolific range of cyber-attacks. Further complicating this issue is the fact that in many cases, network connectivity has been bolted onto existing or lightly modified products, often with little to no consideration for security.

Within the past several years, a significant uptick has been seen like attacks specifically targeting these IoT devices for exploitation. One significant incident involved the Mirai botnet, which executed several attacks in late 2016, including distributed denial of service attacks against a major DNS provider ("Mirai," 2018). After the Mirai source code was released, it became apparent that Mirai created bots for its network by logging in to IoT devices that had default, well-known credentials still enabled. A partial list of these devices was reverse engineered by taking the credentials in the source code and identifying devices that used these credentials as default settings (Krebs, 2016). While some mitigation has occurred against the initial Mirai strain, the release of the source code continues to invite new variants that target additional IoT devices via default credentials (Goodin, 2017).

To mitigate some of these risks, we are introducing PioT. PioT is a Raspberry Pi-based platform, designed to be placed in-line in front of an IoT asset that cannot be secured properly by other means. Some essential components are included in this build

James Levte-Vidal. james.levte.vidal@gmail.com
like Bro for Intrusion Detection, Squid for proxying traffic, iptables for firewalling, and syslog-ng for log shipping. The concept behind PioT is to create a platform that is flexible and expandable to many business needs. Not every IoT device is the same, nor has the same vulnerabilities - but PioT should be able with additional software and configuration to protect these devices from many common threats. When mapped against the OWASP Top 10 for IoT ("Top 10 IoT Vulnerabilities," 2015), we believe a small, configurable, inline tool can partially or completely mitigate several of the vulnerabilities outlined, including:

- I1 - Insecure Web Interface and I2 - Insufficient Authentication/Authorization - by proxying the traffic and redirecting to a webpage hosted on PioT, we can ensure proper bespoke authentication is completed before accessing the device;

- I3 - Insecure Network Services, I8 - Insufficient Security Configurability, and I9 - Insecure Software/Firmware - Utilizing firewalling on PioT, in conjunction with proxying of permitted traffic we can restrict access to the device to those with an authorized need to access it. Additionally, Bro intrusion detection will allow us to alert on events of interest such as attempted access from unauthorized parties.

At this time, the base product is being discussed, but we hope if this is a useful project to many, that we can create a community of contributions to the project to enhance and allow PioT to mitigate many types of threats. As a side benefit of the project, many ICS devices face some of the same challenges as they are being network connected more frequently than in the past ("USA has most Internet-connected," 2016). It is possible that PioT may be able to be used to protect ICS devices as well in some circumstances.

2. Creation of the Solution

2.1. Hardware

To create the PioT testing and development environment, several items of hardware are required. For the purposes of demonstration, we will use another Raspberry Pi to act
as an IoT style vulnerable web server. The overall architecture of the test platform looks similar to this:

![Diagram of PioT test platform]

### 2.1.1. Raspberry Pi 3 B (Quantity: 2)

The Raspberry Pi 3 B is a small form factor, low-cost device. This was chosen in the hopes that should PioT be used in an actual defense capacity, the solution could be implemented and maintained for a relatively low cost. In this example, two units are required to act as the PioT device and as the IoT ‘victim’. Raspberry Pis can be obtained at many electronics and hobbyist locations for less than $50. Some additional hardware will also be needed such as storage for the devices (MicroSD card) and power adapters. The overall cost is still very minimal.

James Levte-Vidal. james.levte.vidal@gmail.com
2.1.2. USB to Ethernet Adapter

A USB (2 or 3) to Ethernet Adapter is required for the PioT device, as it is multihomed to ensure all wired traffic passes through the PioT. Known good USB to Ethernet device listings that work with the Raspberry Pi is maintained (“RPi USB Ethernet Adapters,” 2016). In this case, the TRENDnet TU3-ETG was utilized which was readily found online for less than $15.

2.1.3. Connectivity and Network Services

While it may seem apparent, this test environment would not work properly without the appropriate connectivity and network services. In this case a simple network setup with DHCP and DNS available as well as a default gateway that was routable to the Internet was sufficient to install, configure, and test this setup. For this example, a standard SOHO router was used configured for DHCP and providing DNS from an Internet Service Provider (ISP).

2.2. Software

2.2.1. Raspbian Jessie

Initially, PioT was built using the most current version of Raspbian, Raspbian Stretch. However, Raspbian Stretch includes OpenSSL 1.1. When attempting to install Bro, the make process failed on the Bro X509 plugin. After some searching, it appears Bro did not (and at the time of this writing, appear to still not) support OpenSSL 1.1. Though some progress has been made, apparently issues with specific OpenSSL feature sets are still preventing Bro support for OpenSSL 1.1 as of the most recent issue update (“[BIT-1775],” 2018). Rather than try to install a lower version of OpenSSL, the decision was made to install Raspbian Jessie instead, which uses OpenSSL 1.0.1. After this, Bro was able to be successfully compiled and run on the platform.

2.2.2. OS Setup

Imaging the Raspberry Pis with the appropriate Raspbian Jessie OS is very straightforward. Various types of software can be used to image Raspbian Jessie onto the MicroSD card (“Installing Operating System Images,” 2018). After that, insert the card into the Raspberry Pi and power up the unit.

James Levte-Vidal. james.levte.vidal@gmail.com
2.2.3. Install Squid Proxy Server

Squid is a commonly used proxy server with support for many Operating Systems. While we will not actively use Squid in this proof of concept, it should be installed as it can readily be used should a deployer of PioT wish to put additional controls around an insecure web interface. In these cases, Squid can easily be used to intercept incoming traffic to the IoT device and redirect it to a different website (likely hosted on the PioT) for authentication as appropriate (“Reverse Proxy using Squid,” 2011). To install squid, run the commands ‘sudo apt-get update’ followed by ‘sudo apt-get install squid3’.

2.2.4. Install Bro

Bro installation was not well documented, but a resource on Github did manage to list a set of steps that were RaspberryPi-focused and worked well to get things running (“raspberry pi bro ids,” 2016). However, the Bro packages in that resource were outdated and were revised to reflect a newer version, specifically 2.5.2. These changes were published in a fork and the original author notified of the changes (“Install instructions for Bro IDS,” 2018).

2.2.5. Install syslog-ng

Syslog-ng is a newer version of syslog that supports various types of output methods and is quite flexible in terms of being able to support many types of input log files (“Github – syslog-ng,” 2018). Syslog-ng was not integrated into this relatively straightforward proof of concept because every business operates differently, and many consumers of PioT will elect to use syslog-ng in different configurations and ways, and some maybe not at all. Regardless, the package was included in the build so that it can be used in various configurations if needed. Syslog-ng was installed from the terminal by issuing the command ‘sudo apt-get install syslog-ng.’

2.2.6. Install Apache

Apache HTTP Server is a commonly used cross-platform web server (“About the Apache HTTP Server,” 2018). In this case, Apache was installed on the ‘victim’ Raspberry Pi for the purposes of ensuring that HTTP requests to its open web interface were logged appropriately by Bro. However, it would be reasonable in the future to include Apache HTTP Server as a component or add-on in the PioT build as well.
intercepting and rerouting calls to the victim IoT device could be managed on the PioT itself.

2.3. Installation and Configuration

2.3.1. Connections

The PioT is connected to the local network via the Raspberry Pi built-in ethernet adapter. The Trendnet USB to Ethernet adapter is installed into a USB port, and connected to the built-in Ethernet adapter on the victim Raspberry Pi. See the previous diagram in 2.1 for visual detail.

2.3.2. Setup bridging

While there are several possible network configurations for the PioT device and the victim IoT device, this proof of concept was performed by setting up a bridge between the two PioT interfaces. The reason for this was to ensure that the victim IoT device had a DHCP IP address from the router and other devices on the local network could communicate with it without NAT being in place. In some deployments, however, other approaches like NAT’ing the traffic from the victim IoT device might be desired and is an option. To configure the bridge, initially run ‘ifconfig’ from a terminal window on PioT to get the names of the two Ethernet interfaces (likely eth0 and eth1).
The PioT network configuration in /etc/network/interfaces on the file system was set up in this manner:

```
pi@raspberrypi:~ $ cat /etc/network/interfaces
# Interface(s) file used by ifup(8) and ifdown(8)
# Please note that this file is written to be used with dhcpcd
# For static IP consult /etc/dhcpcd.conf and "man dhcpcd.conf"

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

auto eth1
iface eth1 inet manual

auto br0
iface br0 inet dhcp
    bridge_ports eth0 eth1
allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

(note in this particular screenshot the bridge was already created)
Here, the settings for eth0 and eth1 can be commented out, as the bridge interface br0 will bring up both and set the externally facing interface for DHCP. In the case of the victim IoT device, since DHCP requests can cross the PioT network bridge, it was configured as a DHCP client, which is very common for many IoT devices:

```
pi@raspberrypi ~$ cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
#
# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
allow-hotplug wlan1
iface wlan1 inet manual
  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
pi@raspberrypi ~$
```

3. Testing

3.1. **IP connectivity and Internet access**

IP connectivity and Internet access were tested by opening a web browser in PioT as well as the victim IoT device and pinging from both devices to the default gateway of the network. All tests were successful.

3.2. **Setup Apache**

Apache was set up and configured to use restricted directories, to place the page behind basic authentication ("How to configure private directories with Apache", 2015). While not completely necessary for this test, if later revisions of PioT choose to authenticate users before connecting to the device, this will be useful for continued testing, as many IoT devices are configured this way.

James Lepte-Vidal. james.levte.vidal@gmail.com
3.3. Setup and run Bro

Now that Bro has been installed, we can start up Bro via broctl. In this installation, broctl is found in /opt/bro/bin and can be started, bringing up the command interface. From there ‘start’ will initialize Bro.

Once Bro is running, logs will be started up in /opt/bro/spool/bro for that session. Once broctl is stopped, they will be archived to another location. However, you can access these files while being written using sudo.
3.4. Connect to Victim IoT Device and access website

Now that all the pieces are in place, it is finally time to connect to our bridged IoT device and access its web interface. The expectation is that the site can be accessed successfully, and information about that connection is logged by Bro. Using a web browser, we connect to the IoT web interface that we set up with Apache, on the standard port (80). For this example, the traffic will be sent by IP address 192.168.15.242 and, the Victim IoT device is at 192.168.15.195. The HTTP connection is completed, and our browser at 192.168.15.242 sees the IoT web interface:

![Image of browser window showing the IoT web interface]

Oh no, I am a very vulnerable IoT device. How did you find me?

In case you don't use google, my credentials are:

Username: admin
Password: admin

These cannot be changed.

Good luck!

However, Bro saw this as well. Looking at the Bro logs again in /opt/bro/spool/bro, it can be observed that they have continued to update, and Bro recognized the HTTP traffic and added a new file specifically for that protocol.
Looking at the http.log file, we see this is the only HTTP connection Bro has observed. However, this file has very large amounts of data and is hard to read because it scrolls back over itself in the output, even when the window is made quite wide. Let’s take this same output and pipe it to bro-cut, which will give us just the fields we specify. For now, we can use the fields for source IP and port, and destination IP and port.

James Levte-Vidal. james.levte.vidal@gmail.com
Here the HTTP connection from our browser at 192.168.15.242 to our victim IoT device at 192.168.15.195 is much clearer. However, even if the connection had been to a different interface and a different protocol, Bro would have identified that also. In the conn.log we see the same connections:

![HTTP connection screenshot]

As this display is less information rich, taking the output of the log and piping it to grep for one of the IP addresses of interest shows us the same HTTP connection (third entry from the bottom) without having to utilize bro-cut.

4. Conclusion

As has been demonstrated in the preceding pages, PioT, in its most base form, is easy to setup and configure a device that can be positioned in front of a vulnerable asset. As such, the most basic objectives of this project have been met. Part of the reason for very straightforward initial goals was to ensure time was not spent on initial capabilities that might not be used by some consumers of the solution. Log shipping was not configured because different users have different logging solutions, with different intake processes and requirements. Proxying of data inbound to the IoT device can be performed; however, it was not set up because one business may wish to proxy web communications with port 80, another protecting telnet or SSH connectivity, and so forth. However, it is apparent that PioT can continue to be enhanced to augment PioTs protection capabilities. As such, the PioT project, configurations, and a to-do list for additional features will be moved to Github. It is my hope that there, others can contribute and ensure the features that are most important to them are included in PioT.

James Levte-Vidal. james.levte.vidal@gmail.com
Some possible enhancements that will be added to the to-do list, in an attempt to make this solution more fully featured include:

- Wireless monitor mode support. In my opinion, the inability to monitor traffic sent over WiFi to IoT devices may be the greatest shortcoming of the PioT project at this time. With the Nexmon drivers, monitor mode capability can be added to the stock PioT without additional hardware. Configuration options should be available after that to respond, alert, or possibly de-authenticate devices attempting to connect to the IoT device (“seemoo-lab/nexmon,” 2017).
- Preconfigured proxy interception settings for commonly open ports (SSH, Telnet, Web)
- Support for placing the IoT device behind Network Address Translation (NAT)
- Scripting to support selecting which options you want PioT to perform (Network: Bridge/NAT, Proxy: configure, wireless monitoring: enabled, etc.) and configuring PioT to match.

The repository for PioT is now available at https://github.com/jamesleytevidal/PioT. I welcome contributions from the community to further this project and make it more usable for anyone who needs it.
References


Retrieved from https://bro-tracker.atlassian.net/browse/BIT-1775


Retrieved from https://httpd.apache.org/ABOUT_APACHE.html


https://github.com/balabit/syslog-ng


https://www2.meethue.com/en-

us/products#filters=STARTER_KITS_SU%2CBULBS_SU%2CLIGHTSTRIPS_SU%2CLAMPS_SU%2CCONTROLS_SU&sliders=&support=&price=&priceBoxes=&page=&layout=12.subcategory.p-grid-icon

Install instructions for Bro IDS on a Raspberry Pi - tested on Model 3B - Bro package versions updated (2018, June 29). Retrieved from

https://gist.github.com/jamesleytevidal/d6547804de47605a0f8ea42b077fe3ed

James Leyte-Vidal. iames.levte.vidal@gmail.com


James Leyte-Vidal. james.levte.vidal@gmail.com
Appendix

The PioT project repository has been created on Github at https://github.com/jamesleytevidal/PioT. The purpose of this repository is to collect the information gleaned while putting together this project, as well as offer a location for others to contribute to the project. Current enhancements needed to further the solution are documented as issues in the project Github page.

It is my hope that anyone who sees value in this solution can contribute to making it better, and I will do the same.
## Upcoming SANS Training

Click here to view a list of all SANS Courses

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Las Vegas Summer 2020</td>
<td>Las Vegas, NV</td>
<td>Jun 08, 2020 - Jun 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Silicon Valley - Cupertino 2020</td>
<td>Cupertino, CA</td>
<td>Jun 22, 2020 - Jun 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Chicago Spring 2020</td>
<td>OnlineIL</td>
<td>Jun 01, 2020 - Jun 06, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s Only, US</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>