How Secure is Secure? Realities of IoT

Peter Vondras, Tufts University

December 14, 2016

1 Abstract

Today shows IoT products in almost every facet of our lives and we will only see more every year. Currently, many devices do not communicate with each other unless they are of the same brand and family but the trend is moving toward greater integration and sharing.

I look to demonstrate that even a well designed and secure IoT device is by current design standards, quite literally a window into our private lives. This project introduces Prioritize, a network traffic analyzer that detects whether a Nest Indoor Security Camera is viewing movement or a still scene. While only a beginning of what is possible, Prioritize demonstrates that tools like encryption go only so far.

2 Introduction

Continuously streaming webcams are becoming ever more common and it is not always clear whether they are recording or not.[6] We have them on our phones and computers, in our homes, watching our children. The expectation is that they bring some layer of convenience or security to our lives. The truth is, one the data leaves the camera, it is susceptible to interpretation from anywhere that can capture the network packets. This interpretation is particularly damaging because it provides users outside of our circle of trust access to the Perception Layer, which is linked to the physical world.[5]

I set out to see what I could learn from a well designed IoT device, the Nest Indoor Security Camera. This camera offers fully encrypted communication from the camera to the cloud, where it can be stored, viewed etc. By observing the network traffic that the camera produced, I learned that the camera communicates consistently with the same server, on port 443. One could always confirm that this server was unchanged by connecting their own Nest Camera to wifi from a nearby location. From here, a malicious party need only gain access to your camera's outgoing network traffic. This could be done via a lan tap, monitoring an unencrypted wifi network or stealing a session key to join and decrypt an encrypted wifi network. I tested the first two of the three methods with success. Most alarmingly, using MAC address lookup it is possible, but not implemented in Prioritize, to infer all of the information that we need without joining the encrypted wifi network!

Through further network traffic analysis of the camera during inactivity, activity, and the transition between, it became clear that:

1. The throughput of the camera while inactive is measurably lower than when it is active.

2. There are predictable bursts of data when the camera is inactive which repeat over a 4000ms interval.

3. There is a measurable increase in bursts when the camera is active.

Leveraging this, Prioritize reliably recognizes a transition from inactive to active or vice versa in a network. Prioritize is also resistant to limited traffic interference as it was tested both on an empty network and in tandem with a heavy TCP upload stream.

3 To the Community

This topic was chosen due to my interest in the Internet of Things, Networks and current research into slack inference of network traffic. It is important because it is very unclear what amount of privacy is upheld when we opt in to one of the many new technological conveniences that continually appear. The first assumption is that when our privacy is undermined, it is because the software provider was un-moral or careless in design, but when they are not careless, another question is how much does the technologies mere existence expose us. The truth seems to be that in a world full of devices, even the most secure devices are compromised by the rest of the system. [1]

4 Experiment

4.1 General Setup

* In order to have repeatable results the camera was aimed at a macbook air 13in Mid 2011 laptop screen on high brightness. A rudimentary alignment jig was used to aid repeatable setup. See Figure 1. A thick shroud was draped over the setup which ensured that no shadows fell across the screen during testing. The image was then monitored via a streaming connection provided by nest.com. Using this setup, a fullscreen image or video encompasses the entire image.

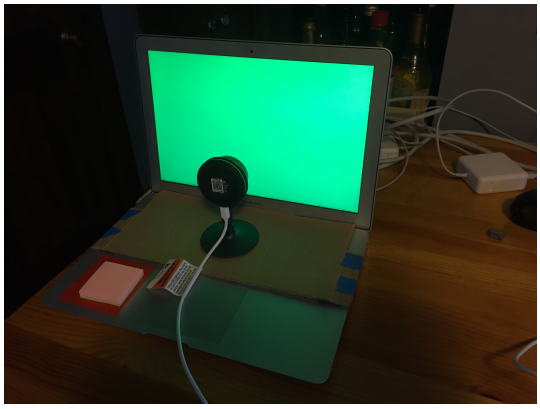

Figure 1: Camera Setup

There are several ways to capture the camera's traffic, one such way is to create a wireless hotspot with another computer and then monitor that computers ethernet traffic. Data was collected using tcpdump using the following command which will filter out all traffic that is not the camera uploading data to the cloud: "sudo tcpdump -n -i en7 dst 52.201.218.19 and dst port 443"

4.2 Test Cases

The following experiments were executed:

1. Inactive Tests without local network congestion: Traffic was captured of the camera monitoring a green, blue and red image. Each run was one minute in duration and was preceded of at least a minute of the same image to ensure that no change was recorded. There were ten runs completed of each of the three image types.

2. Active Tests without local network congestion: A video of strobing single

*This experiment and the Prioritize source code was created for dual purpose, the exploration of inferring slack of network traffic and as an example of how even the use of well designed and secure IoT products exposes the user to risk and possible privacy invasion. Therefore, sections three and four are replicated for the use of both perspectives.

color screens was downloaded and played in front of the video camera for one minute.[4] This experiment was performed ten times.

3. Transition between Inactive and Active Tests without local network congestion: The camera is aimed at the strobe video which is paused on a single color image. Every minute, the video is resumed or if it is running, paused. This is done for a total of twenty minutes, yielding twenty transition from Inactive to Active and twenty from Active to Inactive.

4. Transition between Inactive and Active Tests with extended TCP upload congestion: The transition test above was repeated for four minutes with the addition of a TCP upload being run on the local network to a host out of the network. Iperf was used to simulate this. The measurement was started at least 10 seconds after the upload traffic was started to allow the router buffer to fill up.

5. Transition between Inactive and Active Tests with extended heavy UDP upload congestion: The transition test above was repeated for four minutes with the addition of 20Mbps UDP upload being run on the local network to a host out of the network. Iperf was used to simulate this. A speed test measured the upload bandwidth at 9.85Mbps. The measurement was started at least 10 seconds after the upload traffic was started to allow the router buffer to fill up.

4.3 Results

The average throughput (Mbps) of all runs of the Inactive and Active tests are summarized in Figure 2. It is clear from these tests that

Average Trhoughput over all tests

Run Num	blue	green	red	strobe
1	0.173	0.180	0.187	0.455
2	0.175	0.181	0.184	0.460
3	0.175	0.182	0.183	0.456
4	0.180	0.182	0.182	0.455
5	0.179	0.183	0.186	0.453
6	0.177	0.179	0.185	0.444
7	0.180	0.183	0.183	0.456
8	0.177	0.183	0.185	0.445
9	0.177	0.180	0.182	0.456
10	0.180	0.181	0.186	0.459
AVG	0.177	0.181	0.184	0.454
STD	0.002	0.001	0.002	0.005

STILL AVG	0.181
STILL STD	0.003

Figure 2: Average throughput of experiments

over a one minute period, there is a statistically significant difference between an Inactive and active camera in terms of throughput.

Additionally, looking at the packets with Wireshark, we can make some observations regarding delivery time and size. The first observation is that packets seem to be sent at two different speeds, either bursts where the interval between packets are less than 5ms or sporadically, where the interval is on the order of 20-60ms, which will be referred to as slow. For the bursts, every packet, except perhaps the last one, is the max size allowable (1514 bytes). This allows us to infer that each of these bursts represents some logically connected pieces of data. If you count every burst as a single packet, there are on average 27 chunks of data sent every second. As the camera reports to supply 30 fps, it is likely that each chunk is the requisite data to update the image.

Another observation is that a large burst starts every 4000ms. Between this time and the next cycle, the throughput drops and al-

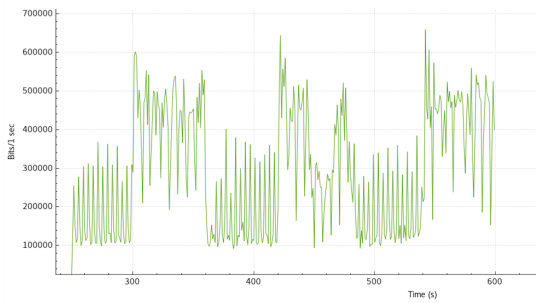

Figure 3: Average throughput of experiments

though their might be a burst, it is less likely and will inevitably be smaller.

Observing the active traffic does not yield much information other than the fact that the 4000ms cycle is replaced by frequent and often bursts or varying length. Even with low traffic on the local network, there were sporadic low traffic segments during an active period, lasting up to 750ms.

The transition experiment without local network congestion looked much like alternating between the inactive and active one minutes tests, but both the the TCP and UDP traffic tests obscured the patterns. The number of bursts during inactive segments was highly increased so as to mask its presence. That being said, the throughput was still an accurate measure of activity for the TCP test. Although the throughput is generally much higher during Activity, there are significant periods low throughput during a period of Activity. This is shown best by the 20 second period in the second active segment in Figure 3 in which the throughput looks much like an Inactive period.

The UDP test seemed to throttle the camera's throughput permanently, likely due to TCP back-off from heavy packet loss.

5 Description of Prioritize

5.1 Design

Prioritize has three modes, Startup, Inactive and Active. Figure 4 shows a state diagram with the general transition logic between the three states. To navigate between the three modes, Proactive uses six variables:

1. Throughput Window (TW): Proactive continually measures the throughput of the camera in Mbps over a period of time. That period of time is called the Throughput Window. It does this by adding every new packet to a Byte_Sum and keeping a Tail pointer to the oldest packet in the Byte_Sum.

2. Throughput Threshold (TT): This is a value in Mbps that is used in the logic to trigger a mode change. Proactive uses 0.23Mbps globally for the nest camera, which was determined through experimental measurements and empirical testing to be both stable and responsive.

3. Burst Count (BC): As observed in the testing, data comes in either slow intervals (20-60ms) or in bursts of max sized packets within 5ms of each other. Proactive keeps track of the number of these bursts over a specified window. Single max sized packets are also considered bursts as they are rare during inactive mode.

4. Burst Window (BW): The period of time back that the system keeps track of the Burst Count. The Burst Window is set globally to 1500ms using the same methods as the TW.

5. Inactive Time: The amount of time that camera has been below the TT.

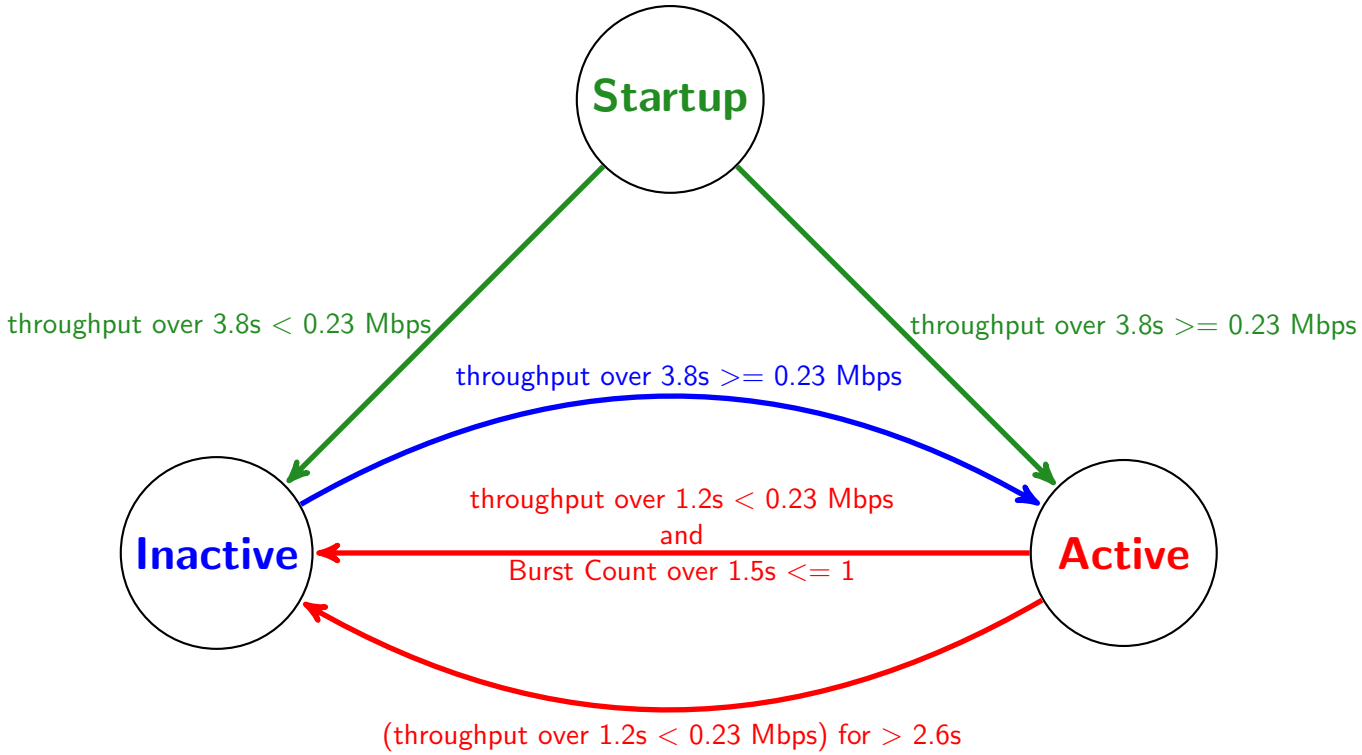

Figure 4: State Diagram

6. Active Time: The amount of time that the camera have been active.

Proactive starts in Startup mode and averages the throughput over the TW. 3800ms is a likely time for the TW because it is the longest window that you are guaranteed not to have two large bursts during an Inactive cycle. The throughput deviation over this period is higher than over the entire minute because it is over a much smaller time but also because, in order to ensure that there is never two large bursts in the TW, there are short periods when there is no large burst at all which pulls the throughput down dramatically. If after the initial TW, the throughput is less than the TT, Proactive moves to Inactive mode, otherwise, it will categorize the camera as Active.

When Inactive mode is triggered, if it is not already, the TW is set to 3800ms. Additionally, Proactive will attempt to backfill the packets into the throughput measurement if the TW was increased. It will stop backfilling if it gets to a packet of max size. This backfill feature helps dramatically with flickers back and forth at the tail end of Action mode. Active mode will be triggered if the throughput crosses the TT.

Upon entering Active mode, the system will change the TW to 1200ms after waiting at least that amount of time by monitoring the Active Time. Reducing the TW lowers the response time for recognizing the end of an Active period. There are two paths for Proactive to reclassify the camera as Inactive:

1. The camera crosses below the TT while the Burst Count is also less than or equal to one. This will allow the single large burst in an inactive mode to not interfere with the end of an Active period. The Burst Count maximum also provides smoothing during an Active period such that random drops in throughput do not inadvertently trip a false transi-

tion change. These false changes were present even when the TW was not decreased.

2. The Inactive Time is greater than 2600ms.

5.2 Performance

Proactive was able to detect activity on all data collected in this experiment with low local network congestion. On average, Action was detected 958ms (See Figures 5) after it actually occurred. Inactivity was also detected reliably at an average of 1548ms after the camera returned to normal. The algorithm used the Burst Count and TT method to return to Inactive mode every time. No jitter between modes or false classifications were observed.

Under heavy TCP traffic, Proactive also was able to reliably classify the camera and detect mode changes. It was not possible to figure out the exact packet that the camera actually changed modes as it was in the uncongested tests. In fact, Proactive was able to downgrade the mode to Inactive about three seconds before I could by inspecting packets. In all cases, the system used the Inactive Time to transition back to Inactive mode. No jitter between modes or false classifications were observed.

The heavy UDP traffic forced the nest camera to drop its throughput below the TT regardless of what the camera was seeing. This is a limitation of the current system.

	Time to classify traffic w/o Interference (s)	
Run Num	Recognise Active	Recognise Inactive
1	0.523	1.501
2	0.896	2.117
3	0.437	1.270
4	0.430	1.535
5	1.418	1.528
6	1.036	1.438
7	1.554	1.517
8	1.300	1.467
9	0.676	1.554
10	1.314	
AVG	0.958	1.548
STD	0.427	0.230

Figure 5: Time elapsed for Proactive to change modes compared to when the camera actually does change traffic modes

6 Defenses and Discussion

It possible to modify Prioritize only slightly, to be able to make all of the same inferences about behavior by monitoring an encrypted wifi channel without gaining access at all. The reason for this is that the nest itself identifies itself via the 802.11 protocol by its MAC address which is can be searched by MAC address databases. One such database is offered by Wireshark. [7] As nest has a limited amount of devices, it would not be hard to pin this one down as the camera based on its network usage. The only other pieces of data that we need are timestamps and packet lengths which are all present in the encrypted packet.

There are several methods that might be used to stop a malicious agent from using this tool on a network. Most of these involve somehow masking your network activity. For example, one could encrypt all traffic and send it through to a proxy that we trusted, where it could be unencrypted and forwarded on. To much similar effect, a VPN could be utilized. Note that this would only work if the attacker was monitoring our network from a lan tap outside of our network.

A method that the camera designers might employ would be to use many, circulating IP addresses and ports which change randomly and are not similar for cameras in the same vicinity. This method however would not

combat the MAC looking method as this does not change for any device. The camera could also send random decoy bursts which would be wasteful, but could help obscure patterns.

Unfortunately if we are not to be intentionally wasteful with bandwidth, there is a limit to what can be done beyond making the communication harder to find. It is worth mentioning though that this type of issue is beyond the knowledge of the average web cam user and therefore the responsibility must be shouldered by another party.

7 Conclusion

This paper introduced Prioritize, a simple command line utility to analyze whether a Nest Indoor Security Camera was monitoring movement or a still scene. The intent of this was to understand how much could be learned from a completely secure IoT device. It was demonstrated that if the network traffic was able to be monitored or intercepted, a malicious party could learn when there was activity in a home or room, giving them detailed information about users without their knowledge. This discovery by itself is not as significant as the understanding that logical inference can be applied to network traffic patterns in addition to specifically what is sent as data. This type of inference is particularly damaging when it is combined with IoT devices that couple digital real time data with the physical world as a security camera does.

8 Acknowledgements

Thanks to Ming Chow for the countless hours demonstrations regarding what is possible at the base level in security today along with the mindset of how to think like an attacker. Fahad Dogar is also appreciated for his guidance on experiment regarding wireless camera monitoring.

References

[1] Tianlong Yu et al, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things,", 2015.

[2] Nermin Hajdarbegovic, "Are We Creating An Insecure Internet of Things (IoT)? Security Challenges and Concerns", https://www.toptal.com/it/are-we-creating-an-insecure-internet-of-things

[3] "Securing the Internet of Things: A Proposed Framework", http://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html

[4] 12 Color Strobe Light Effect *https://www.youtube.com/watch?v=FTYY9pwlxjE*

[5] M.U. Farooq, Muhammad Waseem et al. *A Critical Analysis on the Security Concerns of hte Internet of Things (IoT)*, 2015

[6] Mlot, Stephanie, *Is Your MacBook Webcam Watching You?*, PCmag.com, 2013

[7] Wireshark MAC lookup tool, *https://www.wireshark.org/tools/oui-lookup.html*