Can the "Gorilla" Deliver? Assessing the Security of Google's New "Thread" Internet of Things (IoT) Protocol

ISE5400: Advanced Network Intrusion Detection GIAC (GCIA)

Author: Kenneth Strayer, kstrayer@gmail.com
Advisor: David Hoelzer

Accepted: September 2017

Abstract

Security incidents associated with Internet of Things (IoT) devices have recently gained high visibility, such as the Mirai botnet that exploited vulnerabilities in remote cameras and home routers. Currently, no industry standard exists to provide the right combination of security and ease-of-use in a low-power, low-bandwidth environment. In 2014, the Thread Group, Inc. released the new Thread networking protocol. Google's Nest Labs recently open-sourced their implementation of Thread in an attempt to become a market standard for the home automation environment. The Thread Group claims that Thread provides improved security for IoT devices. But in what way is this claim true, and how does Thread help address the most significant security risks associated with IoT devices? This paper assesses the new IEEE 802.15.4 "Thread" protocol for IoT devices to determine its potential contributions in mitigating the OWASP Top 10 IoT Security Concerns. It provides developers and security professionals a better understanding of what risks Thread addresses and what challenges remain.

1. Introduction

Internet of Things (IoT) devices have become ubiquitous. The Gartner Group estimates that as of 2017, 8.4 billion connected IoT devices will be in use, not including smartphones, tablets, or computers, representing an increase of 30 percent from 2016 (Gartner, 2015). Security incidents associated with IoT devices have recently gained high visibility, such as the Mirai botnet that exploited vulnerabilities in remote cameras and home routers. Undoubtedly, the number of IoT devices will continue to expand, rapidly creating an ever-growing security concern.

The broad range of available protocols serves to compound the security problem associated with IoT. Developers have a choice among many competing technologies to include Wi-Fi, Bluetooth, ZigBee, cellular, Near Field Communication, Z-Wave and others, all of which come with inherent security advantages and disadvantages. In 2014, a consortium released a new networking protocol vying to become a market standard for the home automation environment. The new "Thread" protocol implements an IEEE 802.15.4 mesh network which is similar to ZigBee, but utilizes IPv6 technology with 6LoWPAN as its foundation. The developers advertise the standard as having "security and low-power features that make it better for connecting household devices than other technologies…" (Randewich, 2014). Google's Nest Labs recently open-sourced their implementation of Thread in an attempt to gain industry adoption as the standard for IoT.

The Thread Group makes strong claims for their new protocol. In one of their press releases, they claim that "Thread closes identified security holes found in other wireless protocols and provides worry-free operation" (Thread Group, Inc., 2014). But in what way is this claim true, and how does Thread help address the most significant security risks associated with IoT devices? Manufacturers have released very few IoT consumer products implementing the Thread protocol, leaving many unknowns. Most of the available information on Thread and the Google Nest implementation has been issued by the Thread Group and Google themselves, or are summations of the original developer's marketing material. Very little independent analysis has been conducted to assess the potential contributions of Thread to IoT security concerns.

This research paper will assess the new IEEE 802.15.4 "Thread" protocol for IoT devices to determine its potential contributions in mitigating IoT security concerns. It will evaluate these potential contributions utilizing the Open Web Application Security Project (OWASP) Top 10 IoT security concerns as a reference benchmark. The results of this study will serve developers and security professionals in better understanding what risks Thread may address and what challenges remain. It will help security professionals better analyze how devices are implementing the protocol at the data link, network, and transport levels.

1.1. Overview of the Thread Protocol

The Thread Group released the latest Thread 1.1.1 Specification on February 13, 2017. The specification provides extensive detail on the Thread protocol and claims to provide everything necessary to implement a Thread networking stack (Thread Group, Inc., 2017). The Thread protocol is described in the specification as "an open standard for reliable, cost-effective, low power, wireless device-to-device communications" (Thread Group, Inc., 2017, p. 1.3). The Thread standard is best referred to as a "network stack" in that it combines existing standards and protocols with specific implementation guidance to define the desired networking architecture. Various protocols were selected to meet the goals of Thread, to include support for IP-based addressing, use of existing hardware technology, scalability, low latency and power requirements, and simplified security (Thread Group, Inc., 2015b). As shown in Figure 1, the Thread networking stack primarily addresses the transport and network layers of the interconnect model, utilizing existing IEEE 802.15.4 radio components at the physical layer. Thread provides flexibility at the application layer, allowing a variety of market applications. According to the Thread technical overview, "Thread defines how data is sent in the network but not how to interpret it" (Thread Group, Inc., 2015c, p. n.p.). The application level flexibility becomes a critical point in assessing Thread's contribution in addressing IoT security concerns.

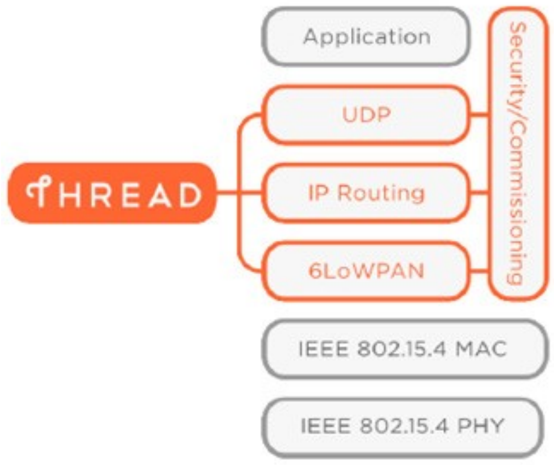


Figure 1. The Thread specification defines existing protocols and standards for various layers of the interconnect model, with implementation guidance primarily focused on the network and transport layers (Thread Group, Inc., 2015c).

In addition to the physical radio, the Thread "stack" also specifies the 802.15.4 Media Access Control (MAC) layer for basic message handling and link layer control "for reliable messaging between adjacent devices" (Thread Group, Inc., 2015b, p. 4). The MAC layer also provides the primary encryption and integrity protection for Thread. IPv6-based addressing is fundamental to the Thread stack at the network layer (Thread Group, Inc., 2015b). To make Thread efficient over low power in a low-bandwidth environment, Thread utilizes "IPv6 Over Low Power Wireless Personal Area Networks" (6LoWPAN) for header compression and end-to-end fragmentation of messages. 6LoWPAN is an adaption layer that encapsulates the 802.15.4 packets for use over the IP network, providing a low overhead mechanism for forwarding multi-hop packets (Thread Group, Inc., 2015). Thread transport communications occur primarily via User Datagram Protocol (UDP) utilizing Datagram Transport Layer Security (DTLS) (Thread Group, Inc., 2015a). As described in the Thread 1.1.1 Specification, "DTLS is a variant of TLS with additional fields in the records to make it suitable for use over an unreliable datagram-based transport" (Thread Group, Inc., 2017, p. 1.4). Since Thread allows use of standard Internet Protocol, any number of additional IP-based services may also be leveraged.

As shown in Figure 2, the Thread network includes end-devices, routers, and commissioners. End devices can serve as routers that provide routing services as well as joining and security services for the network. They can also act as a border router that provides connectivity from the 802.15.4 network to external Wi-Fi or Ethernet (Thread Group, Inc., 2015b). A commissioner is the authentication server for the network, providing credentials for joining the network. Commissioners can be routers on the Thread network or external devices connected to the border router, such as a smartphone connected via Wi-Fi (Thread Group, Inc., 2015a).

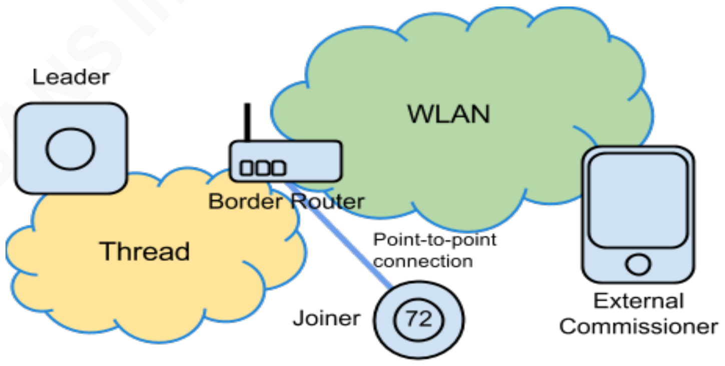


Figure 2. One of several configurations that can be used with Thread for commissioning, authorizing, and joining of new devices.

Thread communicates as a mesh network to provide reliability, allowing individual devices to forward messages for other devices towards their end-destination. Each router device in the network has connectivity and current paths for all other routers in the network, communicating with Mesh Link Establishment (MLE) messages. The MLE messages establish and configure secure links, detect neighboring devices, and maintain routing costs between devices as the network changes (Thread Group, Inc., 2015b), creating an adaptable and secure transport architecture that requires no user interaction to maintain.

A device that seeks to join a Thread network must go through three phases: discovery, commissioning and attaching. A joining device discovers new networks through a beacon request. A responding beacon contains a payload with the network Service Set Identifier (SSID). Commissioning is typically performed by utilizing a commissioning application that is external to the 802.15.4 network, such as a smartphone connected via Wi-Fi through the Thread border-router, designed to force user-initiation for joining. Once the joining device has the required commissioning approved credentials and security material, it completes attaching to the Thread network via a Thread router (Thread Group, Inc., 2015b).

1.2. Overview of the OWASP Top 10 IoT Security Concerns

The Open Web Application Security Project (OWASP) is most commonly known for its Top 10 list of common web application vulnerabilities. But in June 2014, OWASP released its first list of Top 10 IoT security concerns. The foundation describes the OWASP IoT effort as being “designed to help manufacturers, developers, and consumers better understand the security issues associated with the Internet of Things, and to enable users in any context to make better security decisions when building, deploying, or assessing IoT technologies” (OWASP Foundation, 2017). The OWASP approach is to take a holistic approach to IoT security to include hardware interfaces, software configurations, network communications, and applications. As stated by one group of researchers, “Security is not an add-on feature; it must be built into the foundation of any given device. The level of security held by a device is derived from both the architecture and coding choices made by developers.” (Sullivan & Sullivan, 2017, p. 14). While no single technology can be expected to resolve all the concerns across the various surfaces of an IoT device, the OWASP Top 10 serves as a useful framework to view the technology’s contributions systematically and holistically.

2. Research Approach and Test Bench

This study utilizes the OWASP IoT Testing Guide to develop an assessment and description of the Thread protocol's potential in mitigating each of the OWASP Top 10 IoT security concerns. It is done primarily through analysis of available documentation to include the Thread Specification 1.1.1, various published white papers, and sample demonstration implementations provided by third-party vendors.

A hardware/software Thread test bench was built to include a control board, multiple radios, and a border router implementing the Thread protocol. This test bench was used to assess implementation of the Thread protocol to include live packet captures of component communications and the commissioning/association process. Analysis of the networking protocol in action provided opportunities to visually observe strengths and potential weaknesses as part of an end-to-end implementation.

The SiliconLabs Mighty Gecko Wireless Starter Kit served as the basis for the test bench, along with the Thread software stack, sample code, and integrated debug adapter. As shown in Figure 3, multiple radio boards enable the creation of a demonstration mesh network utilizing a built-in IoT switch and light application to exercise the Thread software stack. The switch sends on/off and level control messages to the light in response to button pushes. A border router device allows management of the traffic between the Thread network and adjacent IP networks as well as the commission of the light and switch Thread nodes.

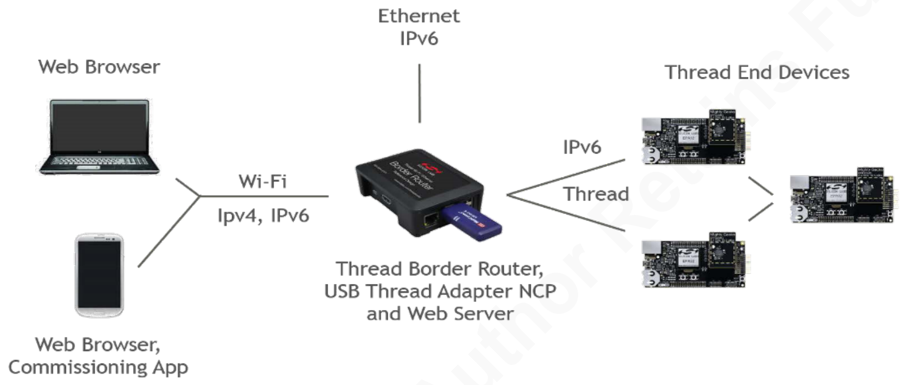


Figure 3. The Thread Border Router System Components with EFR32 Mighty Gecko Wireless SoC Starter Kit served as the basis of the analyzed test bed.

The SiliconLabs Simplicity Studio suite of tools provided sample applications, internal mesh network debugging, and packet tracing used to understand, analyze, and depict how the thread protocol implements security. Simplicity Studio's network analyzer is a graphical tool that displays network and node activity in real-time from either an untrusted perspective or with security keys to allow packet decryption and analysis. The network analyzer was used to validate the security assessment against the OWASP IoT concerns.

3. Assessment of OWASP Top 10 Security Concerns

3.1. Insecure Web Interface

The first of the OWASP Top 10 IoT concerns deals with insecure web interfaces. As observed by the Infosec Instittue, "The fact that your TV, toaster or baby monitor includes a web server is often a surprise" (Infosec Institute, 2014). Web interfaces are often poorly designed and insecure. Chances are, if a device has a web interface, it will also include default credentials. In 2012, the web application on TrendNet cameras was found to expose a full video feed to anyone who accessed it. While it included a secure sign-on capability, hackers quickly discovered that the authentication mechanism was just for show, and could easily be bypassed (Notopoulus, 2012). Besides default or weak passwords, one must assess the web interface for all the common vulnerabilities, to include cross-site scripting, SQL injection, lack of secure data transmission, and faulty account lockout mechanisms to prevent brute forcing (OWASP Foundation, 2016).

As shown previously, the Thread networking stack primarily addresses requirements at the transport and network layers of the interconnect model, providing broad flexibility at the application layer. The implication for Thread devices is that they are subject to all the same web interface vulnerabilities as any other IoT device. While Thread-based systems could have any number of web interfaces, the protocol defines two specific instances where a web interface is the standard implementation. First, the preferred method of commissioning new devices is through an external commissioner that allows a human administrator to manage joining to the Thread network. This device may be a smartphone or other device connected via a Wi-Fi network, or may be further extended to the cloud. Secondly, a Thread border router is typically employed to serve as a gateway between the 802.15.4 and the external Wi-Fi network. Both Thread features necessitate web interfaces for authentication and configuration, bringing all the common web vulnerabilities to the Thread IoT network. Before deployment, developers should test operational interfaces for account enumeration, weak or default credentials, account lockout, and fuzz test for SQL-Injection, cross-site scripting, or other flaws (OWASP Foundation, 2016).

The SiliconLabs test bench used in this study included sample applications and web interfaces for both the development border router (shown in Figure 4) and an Android-based commissioner. Considering the sample applications are for demonstration purposes only and not inherently controlled by the standard, this study did not conduct a complete vulnerability assessment. However, initial investigation of the Thread border router revealed a web interface with no credentialing interface, needing only a direct connection through its attached Wi-Fi access point. While the SSID of the access point was an arbitrary hex number, the passphrase was hard coded as "solutions" without any means to change the default setting. Additionally, as described in the quickstart documentation, when connecting to the border router, the commissioning application requires an admin password that is "set at compile time by the Border Router application and printed on stdout immediately after boot" (Silicon Laboratories, Inc., 2017b, p. 8). For the demonstration application, the developers set the admin password to "COMMPW1234."

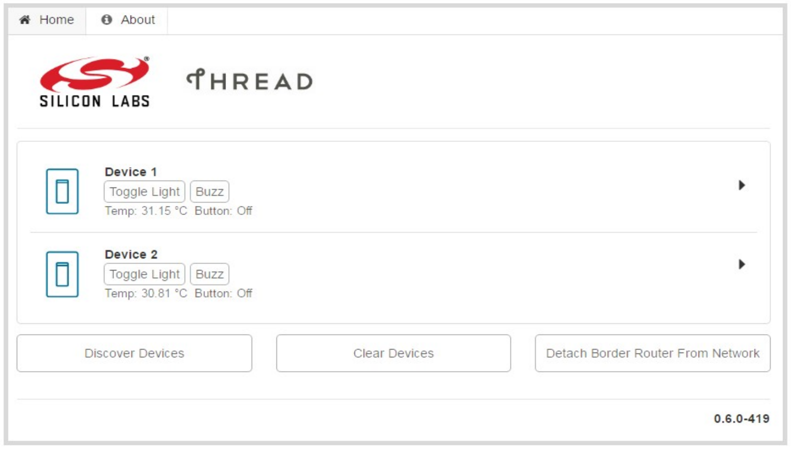


Figure 4. The SiliconLabs Thread Border Router Sample Web Interface

For further analysis, the web interface in the border router was assessed using OWASP Zed Attack Proxy (ZAP), a free security tool used to scan web interfaces and applications for security vulnerabilities (OWASP Foundation, 2017a). The scan revealed two low-risk and one medium-risk alerts which included an issue with the X-Frame-Options header that could leave the web interface vulnerable to "ClickJacking" attacks. These problems would be considered critical flaws in any deployed product and demonstrate that the Thread protocol provides no significant contribution in addressing this particular OWASP Top 10 concern. In fact, Thread is agnostic at the application layer and is not designed to provide any web interface security enhancements.

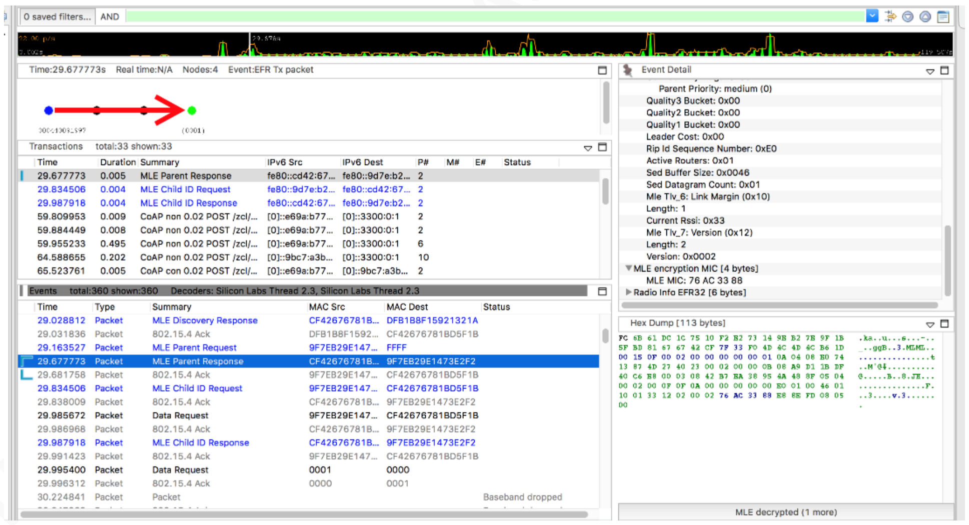
3.2. Insufficient Authentication/Authorization

For IoT, authentication and authorization primarily involve weak or insufficiently protected passwords or credentials, or faulty authentication schemes. Default passwords provide easy access, while lack of mandatory password complexity can result in quick brute-force attacks. The Mirai botnet performs extensive scans of IP addresses to locate under-secured IoT devices with easily- guessable or default login credentials (Herzberg, Bekerman, & Zeifman, 2016). Some protocols, such as HTTP and FTP are notorious for passing credentials "in the clear" and can be easily sniffed and captured. These issues are all common in the implementation of IoT because developers often assume that interfaces will only be exposed on internal networks with minimal threat access (OWASP Foundation, 2017b). The OWASP security concern goes beyond credentials for web interfaces and addresses key management and network service authorizations. With poor key management or authentication, loss of a single node can compromise the entire system, or break the confidentiality and integrity of messages from other nodes (Sastry & Wagner, 2004).

Several credentials come into play in the ordinary operation of Thread devices, as well as in the joining and credentialing process. As discussed above, the web interface on commissioning devices, border routers, or the edge devices are not controlled by the Thread standard and may often be lacking appropriate security controls. However, the Thread standard does provide specific guidance on the implementation of transport and media access layer authentication and encryption. The standard claims that "Devices do not join the Thread Network unless authorized and all communications are encrypted and secure" (Thread Group, Inc., 2015b, p. 3). In order to achieve this, Thread utilizes a network-wide key at the Media Access Layer (MAC) to implement standard IEEE 802.15.4 authentication and encryption. The Thread standard describes the MAC layer encryption key as being "an elementary form of security used to prevent casual eavesdropping and targeted disruption of the Thread Network from outsiders without knowledge of the network-wide key" (Thread Group, Inc., 2015a). However, the network-wide key is pre-shared and stored in non-volatile memory in the edge device. Any compromise of a Thread device could reveal the key and allow compromise of the network (Thread Group, Inc., 2015a). Also, distribution of the network-wide key to new devices on an IoT network is problematic. Asking consumers to enter authentication credentials into IoT devices that lack robust user interfaces adds complexity to the user experience, and the passing of credentials over unsecured connections would also be unacceptable. The Thread protocol commissioning process resolves these challenges.

During Thread network formation, the border router generates a random network master key. According to the Thread technical overview, the Thread software stack does not provide any mechanism for retrieving the key once created. If a Thread device is not yet a member of a Thread network and seeks to join, the thread protocol demands that the device first establish a secure Datagram Transport Layer Security (DTLS) connection with a Thread Border Router. Meanwhile, the commissioning device (an off network smart phone, for example) establishes a secure DTLS session with the border router using a pre-determined commissioning passphrase. This passphrase is used to derive an enhanced key using key stretching (Thread Group, Inc., 2015a). A human operator then authenticates and authorizes the new joining device through the commissioning device. Once authorized, the border router provides the device the necessary security material to attach to the network over the secure DTLS connection that attackers cannot intercept. At no point does the commissioning device ever receive or hold the network security credentials, protecting from off-network exploitation (Silicon Laboratories, Inc., 2017a). Once joiner and border router exchange the network-wide key, the nodes utilize MLE messages “to establish and configure secure links, detect neighboring devices, and maintain routing costs between devices as the network changes” (Thread Group, Inc., 2015b).

This paper’s research included observation of the Thread commissioning process to confirm secure implementation. The test bed study included both trusted network captures (with internal network keys to decode traffic) and untrusted sniffing. Figure 5 illustrates the authentication and key exchange process with MLE Parent and Child requests and responses. Thread commissioning provides a secure means for distribution of key materials and simplicity in authorizing new devices to the network. The Thread border router commissioning process allows an autonomous self-configuring mesh protocol to implement MAC link-level security (Silicon Laboratories, Inc., 2017) in a simplified, user-friendly manner and significantly contributes in addressing the OWASP IoT concern for authentication and authorization.



Transactions total:33 shown:33

Time	Duration	Summary	IPv6 Src	IPv6 Dest	P#	M#	E#	Status
29.677773	0.005	MLE Parent Response	fe80::cd42:67...	fe80::9d7e:b2...	2			
29.834506	0.004	MLE Child ID Request	fe80::9d7e:b2...	fe80::cd42:67...	2			
29.987918	0.004	MLE Child ID Response	fe80::cd42:67...	fe80::9d7e:b2...	2			
59.809953	0.009	CoAP non 0.02 POST /zcl/...	[0]::e69a:b77...	[0]::3300:0:1	2			
59.884449	0.008	CoAP non 0.02 POST /zcl/...	[0]::e69a:b77...	[0]::3300:0:1	2			
59.955233	0.495	CoAP non 0.02 POST /zcl/...	[0]::e69a:b77...	[0]::3300:0:1	6			
64.588655	0.202	CoAP non 0.02 POST /zcl/...	[0]::9bc7:a3b...	[0]::3300:0:1	10			
65.523761	0.005	CoAP con 0.02 POST /zcl/...	[0]::e69a:b77...	[0]::9bc7:a3b...	2			

Events total:360 shown:360 Decoders: Silicon Labs Thread 2.3, Silicon Labs Thread 2.3

Time	Type	Summary	MAC Src	MAC Dest	Status
29.028812	Packet	MLE Discovery Response	CF42676781B...	DFB1B8F15921321A	
29.031836	Packet	802.15.4 Ack	DFB1B8F1592...	CF42676781BD5F1B	
29.163527	Packet	MLE Parent Request	9F7EB29E147...	FFFF	
29.677773	Packet	MLE Parent Response	CF42676781B...	9F7EB29E1473E2F2	
29.681758	Packet	802.15.4 Ack	9F7EB29E147...	CF42676781BD5F1B	
29.834506	Packet	MLE Child ID Request	9F7EB29E147...	CF42676781BD5F1B	
29.838009	Packet	802.15.4 Ack	CF42676781B...	9F7EB29E1473E2F2	
29.985672	Packet	Data Request	9F7EB29E147...	CF42676781BD5F1B	
29.986968	Packet	802.15.4 Ack	CF42676781B...	9F7EB29E1473E2F2	
29.987918	Packet	MLE Child ID Response	CF42676781B...	9F7EB29E1473E2F2	
29.991423	Packet	802.15.4 Ack	9F7EB29E147...	CF42676781BD5F1B	
29.995400	Packet	Data Request	0001	0000	
29.996312	Packet	802.15.4 Ack	0000	0001	
30.224841	Packet	Packet			Baseband dropped

```
      Parent Priority: medium (0)
    Quality3 Bucket: 0x00
    Quality2 Bucket: 0x00
    Quality1 Bucket: 0x00
    Leader Cost: 0x00
    Rip Id Sequence Number: 0xE0
    Active Routers: 0x01
    Sed Buffer Size: 0x0046
    Sed Datagram Count: 0x01
    Mle Tlv_6: Link Margin (0x10)
    Length: 1
    Current Rssi: 0x33
    Mle Tlv_7: Version (0x12)
    Length: 2
    Version: 0x0002
▼MLE encryption MIC [4 bytes]
    MLE MIC: 76 AC 33 88
▶Radio Info EFR32 [6 bytes]
```

Hex Dump [113 bytes]

```
FC 6B 61 DC 1C 75 10 F2 B2 73 14 9B B2 7B 9F 1B   .ka..u...s...-..
5F BD 81 67 67 42 CF 7F 33 F0 4D 4C 4D 4C B6 1D   _..ggB..3.MLML..
00 15 0F 00 02 00 00 00 00 00 01 0A 04 08 B0 74   ...............t
13 87 4D 27 40 23 00 02 00 00 0B 08 A9 D1 1B DF   ..M'@#..........
40 C6 B8 00 03 08 42 B7 BA 38 95 4A 48 8F 05 04   @.....B..8.JH...
00 02 00 0F 0F 0A 00 00 00 00 00 E0 01 00 46 01   ..............F.
10 01 33 12 02 00 02 76 AC 33 88 E8 8E FD 08 05   ..3....v.3......
00                                                .
```

MLE decrypted (1 more)

Figure 5. The Thread commissioning process and network key exchange were observed in the study's test bench from a trusted perspective (with network keys to decode traffic) to confirm secure implementation.

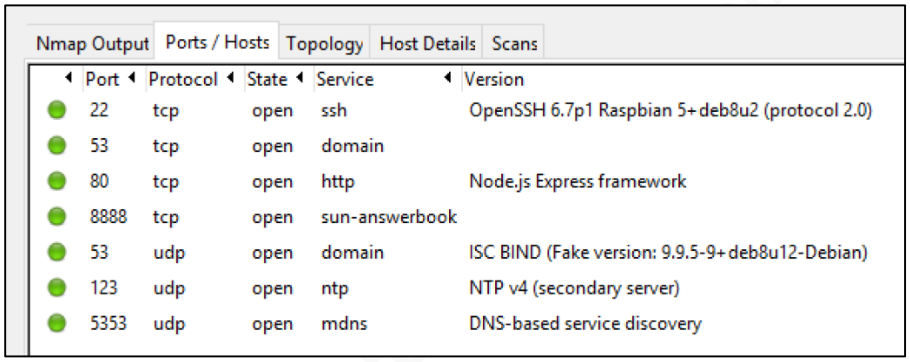
3.3. Insecure Network Services

Weak network services in IoT devices can result in denial-of-service, or facilitate attacks on other devices. Devices may contain open ports that are unnecessary for their intended functionality. Developers often overlook these ports on IoT devices, assuming the network interfaces will not be exposed to external networks. Besides providing an access vector with weak credentials, these services can also often be exploited via buffer overflow or fuzzing (OWASP Foundation, 2017b). Attackers are very familiar with the vulnerabilities posed by insecure device ports and services. For example, the Mirai botnet went so far as to scan the infected device after initial infection and close off any open

SSH, Telnet, or HTTP port services to prevent further infection from competing malware (Herzberg, Bekerman, & Zeifman, 2016). This is but one example of why insecure network services are a concern for IoT.

The Thread 1.1.1 Specification provides flexibility for implementation of various communication and commissioning topologies that may include border routers and off-network commissioning devices (Thread Group, Inc., 2017). Thread does not mandate specific hardware, software, or operating systems for such componentry, allowing configuration and deployment to support vendor-specific features while mandating consistency for the Thread specific functions (Thread Group, Inc., 2017). This flexibility poses immense challenges in securing network services and communication ports. The specification calls for various inter-device message exchanges utilizing UDP ports but has no limitation regarding the use of other UDP or TCP ports for functionality that is outside the constraints of Thread. The specification includes a series of SHOULD statements regarding firewalls and control of border router traffic but is focused on implementation of specific Thread processes broader security concerns.

The Thread test bed devices utilized in this analysis included a border router running Linux Raspbian Jessie Lite operating system as part of a standard Raspberry Pi computing device. A simple port scan of the device with ZenMap (Figure 6), revealed open ports without apparent functionality for the Thread network. While the border router must be able to assign IPv6 addresses to join edge-devices, it is not clear as to why a DNS server (TCP 53) is exposed on the interface facing the public internet or LAN router. TCP port 8888 and 5353 have no documented functionality for the test bed device. And while a Network Time Protocol service could be beneficial to an IoT network, exposure on the public side of the router only opens the system to potential additional exploits. For the test bench Linux configuration, IPTABLES was enabled providing a firewall service. However, the firewall was configured to allow all UDP traffic by default. As observed in this demonstration implementation, the Thread protocol does not provide significant contributions to address the OWASP IoT concern regarding insecure network services.



	Port	Protocol	State	Service	Version
●	22	tcp	open	ssh	OpenSSH 6.7p1 Raspbian 5+deb8u2 (protocol 2.0)
●	53	tcp	open	domain	
●	80	tcp	open	http	Node.js Express framework
●	8888	tcp	open	sun-answerbook	
●	53	udp	open	domain	ISC BIND (Fake version: 9.9.5-9+deb8u12-Debian)
●	123	udp	open	ntp	NTP v4 (secondary server)
●	5353	udp	open	mdns	DNS-based service discovery

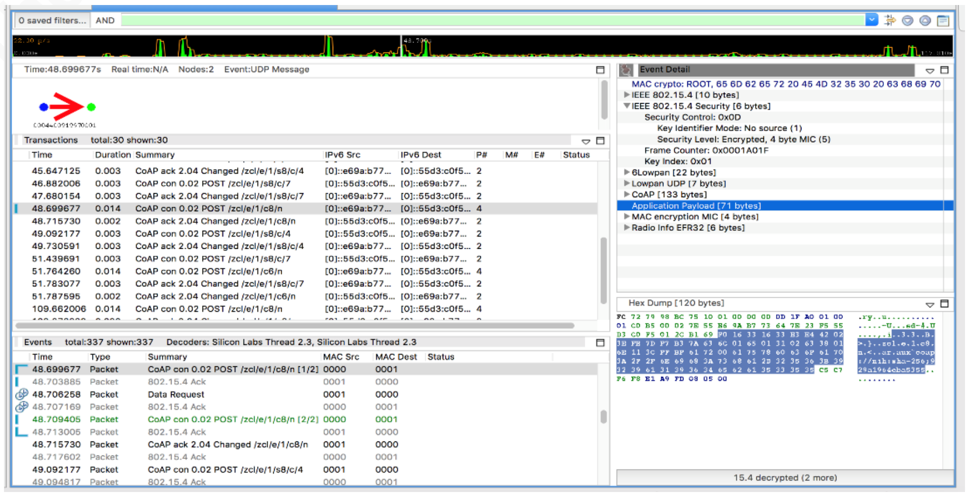
Figure 6. A port scan of the test bed border router revealed several open ports without apparent functional requirements associated with the operation of the Thread system and may represent vulnerabilities.

3.4. Lack of Transport Encryption

Transport encryption prevents data from being viewed as it travels across networks. Local networks are usually unencrypted and visible to anyone on the network. Wireless networks can often be misconfigured resulting in unauthorized access. IoT devices may utilize proprietary or weak encryption protocols. Lack of encryption can lead to exposure of data, but more importantly, it can provide critical information necessary to further compromise an IoT device or network (OWASP Foundation, 2017b). The use of encryption on IoT devices has been a constant challenge given the significant power drain associated with advanced features. To significantly contribute to resolving this concern, an IoT standard must mandate accepted protocols that can operate in low-power environments.

Thread is advertised as a secure, power efficient standard for IoT. According to a Thread overview briefing, "Host devices can typically operate for several years on AA type batteries using suitable duty cycles" (Thread Fundamentals, 2017). To extend operations, Thread allows devices to sleep with adjacent nodes monitoring activities. The protocol mandates neighbor information exchange to include information on sleepy end devices and their sleep cycles (Thread Group, Inc., 2015b). These power management features allow the implementation of AES-128 link-layer security provided by the 802.15.4 MLE protocol. Additionally, since Thread utilizes 6LowPAN to encapsulate the 802.15.4 messages in IPv6, Thread allows the application to use any additional internet security protocol for end-to-end communication.

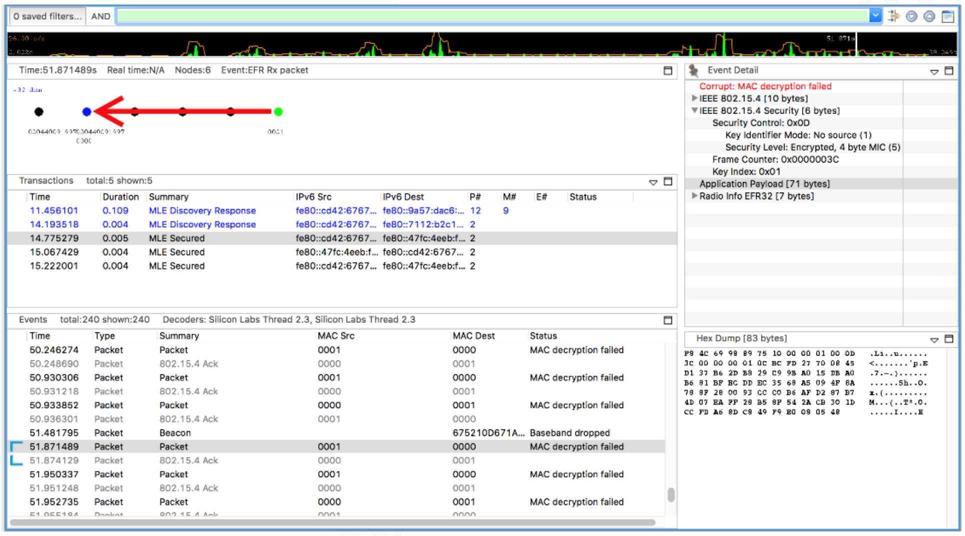
The study captured network traffic from the test bench and sample switch-light application to validate the operation of the Thread MLE message encryption. The initial data stream shown in Figure 7 was obtained by entering the pre-shared network key in the Simplicity Studio network analyzer module, allowing decryption of all message traffic to include the 71-byte application payload containing the 1/c6/n/ message to turn the light on or off. These captures demonstrate that if an attacker has access to the pre-shared keys (possibly through physically compromising and edge-node device), then the network confidentiality and integrity cannot be assured. When the same data stream is captured without access to the preshared keys, as shown in Figure 8, confidentiality and integrity is assured through MLE based encryption.



```
Time:48.699677s   Real time:N/A   Nodes:2   Event:UDP Message

Transactions   total:30 shown:30
Time        Duration Summary                              IPv6 Src          IPv6 Dest         P#  M#  E#  Status
45.647125   0.003    CoAP ack 2.04 Changed /zcl/e/1/s8/c/4  [0]::e69a:b77...  [0]::55d3:c0f5...  2
46.882006   0.003    CoAP con 0.02 POST /zcl/e/1/s8/c/7     [0]::55d3:c0f5...  [0]::e69a:b77...  2
47.680154   0.003    CoAP ack 2.04 Changed /zcl/e/1/s8/c/7  [0]::e69a:b77...  [0]::55d3:c0f5...  2
48.699677   0.014    CoAP con 0.02 POST /zcl/e/1/c8/n       [0]::e69a:b77...  [0]::55d3:c0f5...  4
48.715730   0.002    CoAP ack 2.04 Changed /zcl/e/1/c8/n    [0]::55d3:c0f5...  [0]::e69a:b77...  2
49.092177   0.003    CoAP con 0.02 POST /zcl/e/1/s8/c/4     [0]::55d3:c0f5...  [0]::e69a:b77...  2
49.730591   0.003    CoAP ack 2.04 Changed /zcl/e/1/s8/c/4  [0]::e69a:b77...  [0]::55d3:c0f5...  2
51.439691   0.003    CoAP con 0.02 POST /zcl/e/1/s8/c/7     [0]::55d3:c0f5...  [0]::e69a:b77...  2
51.764260   0.014    CoAP con 0.02 POST /zcl/e/1/c6/n       [0]::e69a:b77...  [0]::55d3:c0f5...  4
51.783077   0.003    CoAP ack 2.04 Changed /zcl/e/1/s8/c/7  [0]::e69a:b77...  [0]::55d3:c0f5...  2
51.787595   0.002    CoAP ack 2.04 Changed /zcl/e/1/c6/n    [0]::55d3:c0f5...  [0]::e69a:b77...  2
109.662006  0.014    CoAP con 0.02 POST /zcl/e/1/c8/n       [0]::e69a:b77...  [0]::55d3:c0f5...  4

Events   total:337 shown:337   Decoders: Silicon Labs Thread 2.3, Silicon Labs Thread 2.3
Time       Type    Summary                               MAC Src  MAC Dest  Status
48.699677  Packet  CoAP con 0.02 POST /zcl/e/1/c8/n [1/2]  0000     0001
48.703885  Packet  802.15.4 Ack                          0001     0000
48.706258  Packet  Data Request                          0001     0000
48.707169  Packet  802.15.4 Ack                          0000     0001
48.709405  Packet  CoAP con 0.02 POST /zcl/e/1/c8/n [2/2]  0000     0001
48.713005  Packet  802.15.4 Ack                          0001     0000
48.715730  Packet  CoAP ack 2.04 Changed /zcl/e/1/c8/n   0001     0000
48.717602  Packet  802.15.4 Ack                          0000     0001
49.092177  Packet  CoAP con 0.02 POST /zcl/e/1/s8/c/4    0001     0000
49.094817  Packet  802.15.4 Ack                          0000     0001

Event Detail
  MAC crypto: ROOT, 65 6D 62 65 72 20 45 4D 32 35 30 20 63 68 69 70
▶ IEEE 802.15.4 [10 bytes]
▼ IEEE 802.15.4 Security [6 bytes]
    Security Control: 0x0D
      Key Identifier Mode: No source (1)
      Security Level: Encrypted, 4 byte MIC (5)
    Frame Counter: 0x0001A01F
    Key Index: 0x01
▶ 6Lowpan [22 bytes]
▶ Lowpan UDP [7 bytes]
▶ CoAP [133 bytes]
  Application Payload [71 bytes]
▶ MAC encryption MIC [4 bytes]
▶ Radio Info EFR32 [6 bytes]

Hex Dump [120 bytes]
FC 72 79 98 BC 75 10 01 0D D0 0D DD 1F A0 01 0D   .ry..u..........
01 CD B5 0D D2 7E 55 B6 9A B7 73 64 7E 23 F5 55   ......~U...sd~#.U
D3 CD F5 01 2C B1 69 F0 16 33 16 33 B3 E4 42 02   ....,.i..3.3..B.
3E FE 7D F7 B3 7A 63 6C 01 65 01 31 02 63 38 01   >.}..zcl.e.1.c8.
6E 11 3C FF BF 61 72 00 61 75 78 60 63 6F 61 70   n.<..ar.aux`coap
3A 2F 2F 6E 69 68 3A 73 68 61 2D 32 35 36 3B 39   ://nih:sha-256;9
32 39 61 31 39 36 34 65 62 61 35 33 35 35 C5 C7   29a1964eba5355..
F6 F8 E1 A9 FD 08 05 00                           ........
```

Figure 7. When capturing data utilizing the pre-shared keys, you can decrypt the payload and read the 71-byte application payload containing the zcl 1/c6/n/ message to turn the light on or off.

Kenneth Strayer, kstrayer@gmail.com



Time:51.871489s Real time:N/A Nodes:6 Event:EFR Rx packet

Transactions total:5 shown:5

```
Time       Duration Summary                  IPv6 Src             IPv6 Dest            P#  M#  E#  Status
11.456101  0.109    MLE Discovery Response   fe80::cd42:6767...   fe80::9a57:dac6:...  12  9
14.193518  0.004    MLE Discovery Response   fe80::cd42:6767...   fe80::7112:b2c1...   2
14.775279  0.005    MLE Secured              fe80::cd42:6767...   fe80::47fc:4eeb:f... 2
15.067429  0.004    MLE Secured              fe80::47fc:4eeb:f... fe80::cd42:6767...   2
15.222001  0.004    MLE Secured              fe80::cd42:6767...   fe80::47fc:4eeb:f... 2
```

Events total:240 shown:240 Decoders: Silicon Labs Thread 2.3, Silicon Labs Thread 2.3

```
Time       Type    Summary        MAC Src   MAC Dest      Status
50.246274  Packet  Packet         0001      0000          MAC decryption failed
50.248690  Packet  802.15.4 Ack   0000      0001
50.930306  Packet  Packet         0001      0000          MAC decryption failed
50.931218  Packet  802.15.4 Ack   0000      0001
50.933852  Packet  Packet         0000      0001          MAC decryption failed
50.936301  Packet  802.15.4 Ack   0001      0000
51.481795  Packet  Beacon                   675210D671A... Baseband dropped
51.871489  Packet  Packet         0001      0000          MAC decryption failed
51.874129  Packet  802.15.4 Ack   0000      0001
51.950337  Packet  Packet         0001      0000          MAC decryption failed
51.951248  Packet  802.15.4 Ack   0000      0001
51.952735  Packet  Packet         0000      0001          MAC decryption failed
```

Event Detail

```
Corrupt: MAC decryption failed
▶IEEE 802.15.4 [10 bytes]
▼IEEE 802.15.4 Security [6 bytes]
    Security Control: 0x0D
        Key Identifier Mode: No source (1)
        Security Level: Encrypted, 4 byte MIC (5)
    Frame Counter: 0x0000003C
    Key Index: 0x01
  Application Payload [71 bytes]
▶Radio Info EFR32 [7 bytes]
```

Hex Dump [83 bytes]

```
F9 4C 69 98 89 75 10 00 00 01 00 0D   .Li..u......
3C 00 00 00 01 0C BC FD 27 70 08 45   <.......'p.E
D1 37 B6 2D B8 29 C9 9B A0 15 DB A0   .7.-.)......
B6 81 BF BC DD EC 35 68 A5 09 4F 8A   ......5h..O.
78 8F 28 00 93 CC C0 B6 AF D2 87 B7   x.(.........
4D 07 EA FF 28 B5 8F 54 2A CB 30 1D   M...(..T*.0.
CC FD A6 8D C9 49 F9 E0 09 05 48      .....I....H
```

Figure 8. Without access to the pre-shared key, the network sniffer cannot read the message payload protected by MLE message encryption.

The Thread protocol implements both the 802.15.4 link-layer encryption as well as IP-based transport layer security enabled by 6LoWPAN and power management features to work in constrained environments. This combination provides a practical means to achieve a high-level of confidentiality and contributes significantly to addressing the OWASP security concern.

3.5. Privacy Concerns

Privacy concerns for IoT devices include both the collection and protection of personal data (OWASP Foundation, 2017b). Given the emerging, ubiquitous nature of IoT devices, personal data can go beyond financial and health records. IoT devices can provide insight into personal activities, preferences, and patterns allowing exploitation for nefarious purposes. Although the collection of personal data is an operational or functional concern, IoT privacy concerns magnify if a device has insufficient authentication, lack of transport encryption, or insecure storage of information (OWASP Foundation, 2017b).

In assessing privacy concerns for a Thread protocol device, security professionals must determine the amount of personal information collected, investigate the use of encryption at rest and in transit, and query end-user choices for data collection (OWASP Foundation, 2016). Apart from data transport, these items are application-level concerns that are not addressed by Thread. The simple switch and light application did not collect or store any private data. However, the included border router running a default version of Linux could be configured to log and store an unlimited collection of data. Except for Thread's default use of AES-128 encryption and its ability to leverage other secure transport protocols, Thread offers little in the way of contribution to the IoT privacy concern.

3.6. Insecure Cloud Interface

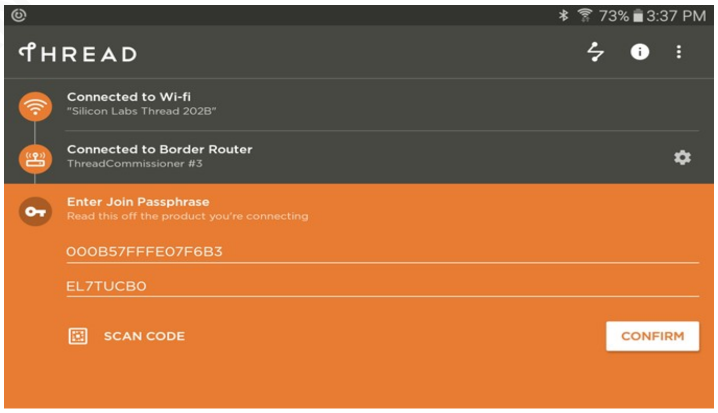
For most IoT devices, cloud-based data storage and access are integral to the required functionality. Off-premise storage of data leads to significant concerns for data protection. Insecure cloud interfaces often have weak credentials or allow account enumeration and manipulation of password reset mechanisms. The specific vulnerabilities are the same as the previous web interface concern which include default or weak passwords, lack of failed login lockouts, faulty password recovery mechanisms, or standard web-based vulnerabilities (OWASP Foundation, 2017b).

The Thread standard does not include any specific provisions for the implementation of cloud interfaces other than the ability to establish a commissioning device in the cloud and the inherent flexibility to implement other IP-based security applications and transport encryption. The Thread standard allows cloud-based data storage given the designed flexibility at the application-level, but the SiliconLabs test bench did not provide a specific cloud implementation. Giving credit for the Thread ease-of-use encryption and authentication mechanisms available for cloud interfaces, Thread only partially addresses the OWASP IoT security concern.

3.7. Insecure Mobile Interface

The presence of an IoT mobile interface implies remote access and potentially the control of the device over insecure public wireless networks. Developers of mobile devices are often pressured to simplify user access to the mobile interface given the constrained nature of the input mechanisms and screen size. Insecure mobile interfaces often have easily guessable credentials, lack two-factor authentication, and fail to encrypt passwords or other data during transport over public networks (OWASP Foundation, 2017b).

Most Thread systems would likely include a mobile interface for commissioning, and the potential for control and monitoring edge devices. The test bench for this study included an Android-based mobile application shown in Figure 9. The mobile interface included advanced features representative of several different consumer applications.



Connected to Wi-fi
"Silicon Labs Thread 202B"
Connected to Border Router
ThreadCommissioner #3
Enter Join Passphrase
Read this off the product you're connecting
000B57FFFE07F6B3
EL7TUCBO

Figure 9. The Android-based Thread commissioning app provided with the test bench system provided functionality for various consumer applications.

Once connected to the border router Wi-Fi network, the Thread mobile application searches for available Thread networks and requests the associated Thread administration password. As discussed in a previous section, the developers hard coded the Thread admin password for the test bench border router as "COMMPW1234." The sample application does not include two-factor authentication, or a means to change the admin password. However, data transport does take advantage of the previously described DTLS secure sessions mandated by the Thread specification providing full encryption during operation. While the Thread specification leaves much of the mobile interface design and security up to the developer, the communications mechanisms established for Thread commissioning simplifies security and partially addresses the OWASP IoT concern.

3.8. Insufficient Security Configurability

The ability to configure security options is essential in providing granular permissions for the access of data or controls for IoT devices. Broad access to certain data or functions on the IoT device may be a desirable feature for some applications, with the necessity of limiting access to administrative features such as the connection to new devices and password setting. To maintain high levels of security and priviledged access, IoT devices require the ability to separate administrative users from ordinary users, and a means for monitoring and logging various security events (OWASP Foundation, 2017b).

The Thread specification provides little guidance on security configurations or separation of administrative and standard user features nor does it discuss monitoring or logging features at the border router. According the SANS Institute Internet of Things Survey, "system monitoring was cited as the second most common security control (65%) currently in use to secure Internet Things" (Pescatore, 2014, p. 19). However, system monitoring relies on the collection of central logs or host-based agents on edge devices. Thread does not control either of these capabilities.

The border router in our test bench is running a version of Linux operating system and is beyond the control of the Thread standard. Root access, storage encryption, communication ports, software updates, and logging are all independent variables that are addressed by the consumer application developer. Neither the web interface for the border router or the Android commissioning app had administrative controls, or a means to enable alerts or notifications. The OWASP IoT guidelines state the need for an active "security audit trail of mobile application interactions with the ecosystem" to include "robust logging and appropriate credentials to track interactions from mobile components" (OWASP Foundation, 2016a). Neither the assessed test bench or the Thread specification provide any capabilities to address this OWASP IoT security concern.

3.9. Insecure Software/Firmware

OWASP includes software and firmware security as a major IoT concern. According to OWASP, "the lack of ability for a device to be updated presents a security weakness on its own" (OWASP Foundation, 2017b, p. 31). First and foremost, devices must have mechanisms to allow easy updates as vulnerabilities are discovered and resolved. Additionally, software and firmware can be insecure if they contain hard-coded sensitive data or credentials. Depending on how systems distribute software and firmware updates, it is possible to intercept and compromise updates, unless mechanisms are in place to deny malicious software configurations, such as signing and verification of code (OWASP Foundation, 2017b).

The Thread specification includes standard message formats for reporting software versions for edge devices and border routers. However, it does not specify any means to manage or distribute software updates to these devices. The border router in the test bench was running a version of Linux installed on an SD card. As part of the test bench analysis, the operating system was updated and patched. However, this process was problematic, requiring advanced knowledge including the ability to manually edit configuration files. Additionally, the analysis included the update of edge device firmware but required the use of a bootloader and flash program. More problematic, the test bench border router required a specific version of Linux, Raspbian Jessie Lite version 2016-05-31. Newer versions of the Raspbian operating system caused conflicts with the Thread border router services. The lack of a consumer-friendly means to update the software or firmware on these devices is indicative of a critical gap in the Thread protocol.

Additional investigation revealed hard-coded credentials in the edge-device application. The SiliconLabs documentation indicated that this was for ease of demonstration only stating, “While Thread applications deployed into the field are expected to use a randomly generated Master Key when starting the network, these Switch and Light examples applications use a hardcoded Master Key that can be found in the switch-implementation.c or light-implementation.c files…” (Silicon Laboratories, Inc., 2017c). While these samples are for demonstration purposes only, they indicate that IoT developers using the Thread specification are subject to the same problematic software or firmware security concerns.

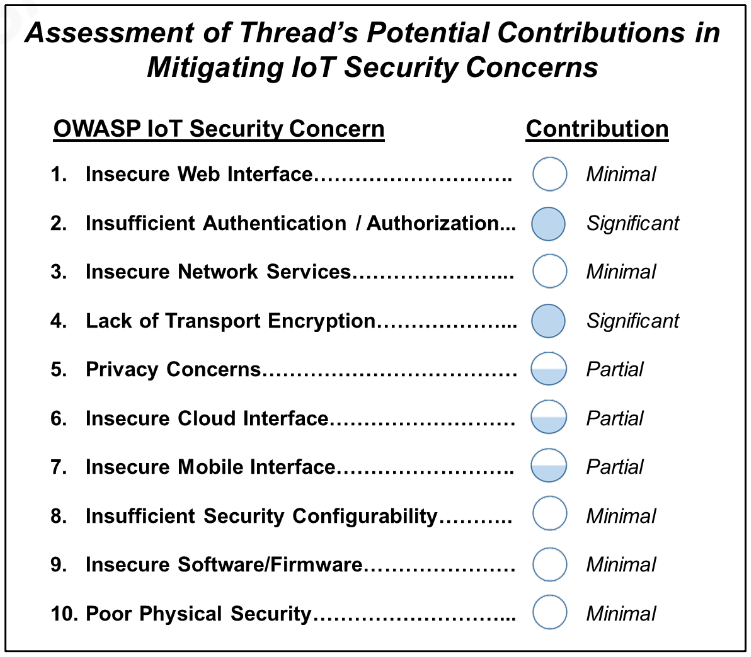
3.10. Poor Physical Security

The last of the OWASP IoT Top 10 security concerns addresses poor physical security. If an attacker can easily disassemble a device or otherwise exploit the provided external ports, the installed operating system, and stored data become exposed. Attackers can modify devices for use in other purposes than those originally intended. One must review how easily device software can be accessed if any ports are present that are not necessary for normal operation, or if any administrative functions are limited or protected from physical tampering. Encryption of data at rest can further protect data on physically compromised IoT devices. (OWASP Foundation, 2017b).

According to the Thread 1.1.1 specification, “A Thread device MAY include multiple physical and media access control interfaces available for radio frequency or wired connectivity” (Thread Group, Inc., 2017, p. 3.13). The test bench border router included multiple ports to include Ethernet, USB, and a removable SD card for the operating system. The sample edge devices included USB and ethernet connections. A simple port scan of the edge devices revealed a TCP 4900 listening port, typically utilized for SQL client/services. In this case, the Ethernet port provides a means for debugging the radio application, but its presence represents an unknown security risk. Additionally, as detailed the Thread documentation, information is stored in non-volatile memory on the edge devices to facilitate rejoining to a network after a power loss or reset. The stored information includes the personal network identification, security material (each key used), and “addressing information from the network to form the devices IPv6 addresses” (Thread Group, Inc., 2015b, p. 19). Based on this analysis, the Thread standard is shown to have little contribution to addressing this OWASP IoT security concern, leaving secure design and testing in the hands of the consumer developer.

4. Conclusion

This study assessed the new IEEE 802.15.4 "Thread" protocol for IoT devices to determine its potential contributions in mitigating IoT security concerns. Figure 10 provides the summary analysis of the protocol. Subjectively analyzed, the Thread protocol provides significant contributions for authentication/authorization, as well as transport encryption.



Assessment of Thread’s Potential Contributions in Mitigating IoT Security Concerns

OWASP IoT Security Concern		Contribution
1. Insecure Web Interface	○	*Minimal*
2. Insufficient Authentication / Authorization	●	*Significant*
3. Insecure Network Services	○	*Minimal*
4. Lack of Transport Encryption	●	*Significant*
5. Privacy Concerns	◒	*Partial*
6. Insecure Cloud Interface	◒	*Partial*
7. Insecure Mobile Interface	◒	*Partial*
8. Insufficient Security Configurability	○	*Minimal*
9. Insecure Software/Firmware	○	*Minimal*
10. Poor Physical Security	○	*Minimal*

Figure 10. The Summary table depicts Thread’s contribution in addressing the OWASP IoT Top 10 security concerns.

Kenneth Strayer, kstrayer@gmail.com

The Thread standard was defined to provide an uncomplicated way to authorize and connect new devices while providing secure transport in a low-power environment. Implementation of 6LoWPAN provides options for implementation of additional IP-based security features. However, even though Thread represents an excellent standard to implement authorization and encryption on consumer IoT devices, its contribution to addressing the wider range of OWASP security concerns is limited. It only partially addresses privacy concerns, cloud, or mobile interfaces. The standard provides a minimal contribution to the remainder of the OWASP concerns, primarily due to Thread being a networking protocol abstracted from the application layer and the physical implementation/configuration of the IoT device. As shown in the study, device security is a factor of both the architecture and the implementation. This is analogous to the larger web application ecosystem that often leverages Transport Layer Security (TLS) using HTTPS. While TLS provides authentication and confidentiality, definitive security depends on the application itself. Thread has a role in providing a secure foundation for IoT systems, but it must be combined with well-conceived designs, thorough testing, and ongoing monitoring and patching.

References

Bush, S. (2015). Security is All in the Mesh Protocol. *Electronics Weekly*, 15.

Digi International. (2016, August 11). *What Makes Thread a Secure Wireless Protocol?* Retrieved from Digi: https://www.digi.com/blog/iot/what-makes-thread-a-secure-wireless-protocol/

Gartner. (2015, November 10). *Newsroom Press Release*. Retrieved August 05, 2017, from Gartner: http://www.gartner.com/newsroom/id/3165317

Herzberg, B., Bekerman, D., & Zeifman, E. (2016, October 26). *Breaking Down Mirai: An IoT DDoS Botnet Analysis*. Retrieved from Imperva Incapsula: https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html

Infosec Institute. (2014, November 10). *How to Test the Security of IoT Smart Devices*. Retrieved from Infosec Resources: http://resources.infosecinstitute.com/test-security-iot-smart-devices/

Krentz, K. F., Rafiee, H., & Meinel, C. (2013). 6LoWPAN Security: Adding Compromise Resilience ot the 802.15.4 Security Sublayer. *ASPI'13*.

Notopoulus, K. (2012, February 3). *Somebody's Watching: How a Simple Exploit Lets Strangers Tap into Private Security Cameras*. Retrieved from The Verge: https://www.theverge.com/2012/2/3/2767453/trendnet-ip-camera-exploit-4chan

Olsson, J. (2014). *6LoWPAN Demystified.* Texas Instruments, Inc.

OWASP Foundation. (2016, May 14). *IoT Testing Guides*. Retrieved from OWASP: https://www.owasp.org/index.php/IoT_Testing_Guides

OWASP Foundation. (2016a, May 14). *IoT Framework Assessment*. Retrieved from OWASP: https://www.owasp.org/index.php/IoT_Framework_Assessment

OWASP Foundation. (2017, August 12). *OWASP Internet of Things Project*. Retrieved from Open Web Application Security Project: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

OWASP Foundation. (2017a, September 3). *OWASP Zed Attack Proxy Project*. Retrieved from OWASP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP Foundation. (2017b, 08 21). *OWASP IoT Top 10 PDF.* Retrieved from OWASP: https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf

Palenchar, J. (2015). How Will Thread Stitch Together a Home Network? *Twice.com*.

Pescatore, J. (2014). *Securing the "Internet of Things" Survey.* SANS Institute.

Randewich, N. (2014, July 15). *Google's Nest Launches Network Technology for Connected Home*. Retrieved August 5, 2017, from Reuters Technology News: http://www.reuters.com/article/us-google-nest-idUSKBN0FK0JX20140715

Sastry, N., & Wagner, D. (2004). Security Considerations for IEEE 802.15.4 Networks. *WiSE'04*.

Silicon Laboratories, Inc. (2017, August 21). UG103.11: Application Development Fundamentals: Thread. Austin, TX.

Silicon Laboratories, Inc. (2017a, August 13). *UG116: Developing Custom Border Router.* Retrieved from SiLabs: https://www.silabs.com/documents/public/user-guides/ug116-border-router-ug.pdf

Silicon Laboratories, Inc. (2017b, August 13). *QSG102: Thread Border Router Add-On Kit Quick-Start Guide.* Retrieved from SiLabs: https://www.silabs.com/documents/public/quick-start-guides/qsg102-thread-border-router-kit.pdf

Silicon Laboratories, Inc. (2017c, August 21). QSG113: Getting Started with Silicon Labs Thread. Austin, TX.

Silicon Laboratories, Inc. (2017d, September 4). *Thread Networking Solutions for the Connected Home*. Retrieved from https://www.silabs.com/products/wireless/mesh-networking/thread

Sullivan, D., & Sullivan, J. (2017). The Internet of Everything is Everywhere; Testing Must Be Too. *Information Security Magazine Insider Edition*, 13-16.

Symantec. (2016, October 27). *Mirai: what you need to know about the botnet behind recent major DDoS attacks*. Retrieved from Symantec Official Blog: https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks

Thread Group, Inc. (2014, July 15). *Introducing Thread: A New Wireless Networking Protocol for the Home.* Retrieved from Thread Group: http://threadgroup.org/news-events/press-releases/ID/20/Introducing-Thread-A-New-Wireless-Networking-Protocol-for-the-Home

Thread Group, Inc. (2015, July 13). Thread Usage of 6LoWPAN. San Ramon, CA.

Thread Group, Inc. (2015a, July 13). Thread Commissioning. San Ramon, CA.

Thread Group, Inc. (2015b, July). Thread Stack Fundamentals. San Ramon, CA.

Thread Group, Inc. (2015c, October 4). Thread Technical Overview. Berlin.

Thread Group, Inc. (2017, February 13). Thread 1.1.1 Specification. San Ramon, CA.