Interested in learning more about security?

SANS Institute InfoSec Reading Room

Detecting Attacks Against The 'Internet of Things'

The need to detect attacks against our networks has exploded with the rapid adoption of connected devices affectionately dubbed the "Internet of Things" (or IoT). Manufacturers are rapidly producing devices to meet consumer and market demand which creates a shortened time-to-market in manufacturing. The level of security in the product development lifecycle becomes questionable, as well as production standards. Vulnerabilities have been showing up targeting the physical interfaces of IoT devices, wireless pro...

Detecting Attacks Against The 'Internet of Things'

GIAC (GCIA) Gold Certification

Author: Adam Kliarsky, adam.kliarsky@gmail.com
Advisor: Kees Leune
Accepted: March 19th, 2017

Template Version September 2014

Abstract

The need to detect attacks against our networks has exploded with the rapid adoption of connected devices affectionately dubbed the "Internet of Things" (or IoT). Manufacturers are rapidly producing devices to meet consumer and market demand which creates a shortened time-to-market in manufacturing. The level of security in the product development lifecycle becomes questionable, as well as production standards. Vulnerabilities have been showing up targeting the physical interfaces of IoT devices, wireless protocols, and user interfaces. It is imperative that intrusion analysts understand how to assess the attack surface, analyze threats, and develop the capability to detect attacks in IoT environments. This paper will review threats, vulnerabilities, attacks, and intrusion detection as it applies to the IoT.

1. Introduction

The Internet of Things (IoT) and Machine to Machine (M2M) technology will increase productivity in ways not seen since the Industrial and Digital Revolutions, but at what cost?

— (Trustwave, n.d.)

The proliferation of connected 'smart' things has created a new gap in the traditional security of endpoints. These devices, known commonly as the "Internet of Things" (or IoT for short) include an extensive list of devices, from wireless sensors, smart meters, security cameras and refrigerators to an extensive variety of machine to machine (M2M) devices. The explosive growth of the IoT has inspired new technologies and protocols to support devices and communication, and has manufacturers pushing product to market faster than security can keep up. The resulting attack surface areas suffer weaknesses to on multiple levels and will continue to do so as the IoT continues to grow.

The drivers of the IoT growth include regulatory compliance across multiple vertical industries (Verizon, 2016) and consumer demand, with anticipated revenue growth forecasted to rise from $130.33 billion in 2015 to $883.55 billion in 2022 (marketsandmarkets.com, 2016). As manufacturers rush to meet consumer demand, the time to market becomes an area of concern. "There has been little to no regulation of manufacturers, which means companies often produce insecure products simply because it is cheaper to do so" (Hayes, 2016). Of course, the implications this has on security, on top of all the other existing threats and vulnerabilities, is yet another issue security teams need to address.

The threats and vulnerabilities that face IoT devices are proportionate to the extensive range of IoT and its attack surfaces. These things suffer attacks against physical interfaces (Rosenfeld & Karri, 2009), attacks against wireless communication and routing protocols, and traditional attacks seen on IP networks. These include attacks against user interfaces, accounts, authentication, and communication (OWASP Internet of Things Project, 2017).

Adam Kliarsky, adam.kliarsky@gmail.com

The OWASP Internet of Things Project published a list of what it considers the top IoT vulnerabilities and lists username enumeration and weak passwords as its top vulnerabilities (OWASP Internet of Things Project, 2017). The Mirai botnet that took out Domain Name Service (DNS) provider Dyn (Hilton, 2016), as well as security blog site KrebsOnSecurity (Krebs, 2016) leverages these weaknesses. The Bashlite botnet from 2014 had the same modus operandi as Mirai. It would scan telnet ports to identify vulnerable devices, and brute force username and password to gain access (Ashok, 2016).

The proliferation of IoT devices is broadening the attack surface. "Attackers will likely invest more resources into taking over the hordes of IoT devices added to the Internet every day" (Laliberte, 2016). Leaked botnet source code only exacerbates the situation as more attackers use and modify the code to expand attack capability. The release of the Bashlite source code spawned over a dozen variations (Level 3 Threat Research Labs, 2016). Mirai infections grew to twice as many devices after its source was released, with threat actors developing new variations (Kan, 2016). Disparity of existing protocols, as well as new ones developed for the IoT, lack standardization.

The growing IoT market and associated vulnerabilities illustrate the significance of the threat and risk it poses. As pointed out by Covisnt, "many of these devices will never receive a single update from the manufacturer as time goes on, so patches aren't an option to help address emerging threats" (Knudson, 2015). Intrusion analysts need to be able to identify new threats and respond accordingly. This paper will discuss common threats and vulnerabilities facing the growing IoT landscape and how intrusion analysts can detect threats against the Internet of Things.

2. Understanding the Threats to the IoT

To develop and implement any type of intrusion detection system, a solid understanding of the ecosystem and associated threats are essential. This means understanding the architecture, technology involved, relevant communication, and of course impact if compromised. This helps an intrusion analyst understand what threats to be looking for, whether already existing or potentially undiscovered. This also helps with

the placement of the intrusion detection sensors. Ingress attacks from the internet may be detected with sensors between in internet point of presence and IoT gateway. Attacks against wireless protocols will need sensors between IoT nodes and gateway. Figure 1 below illustrates a basic IoT communication environment.

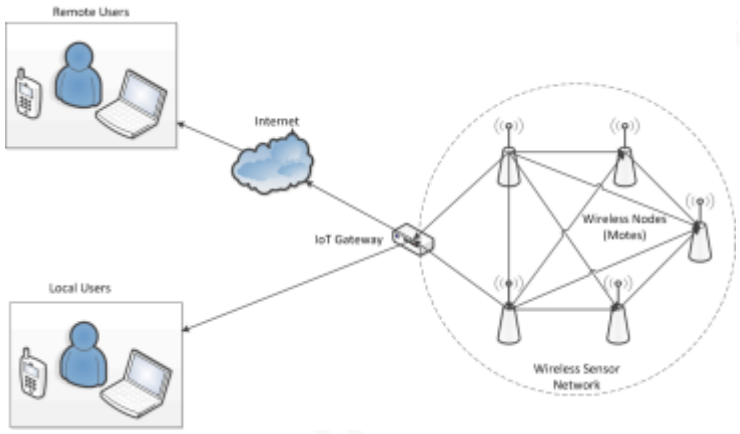


Figure 1 - Basic IoT Communication Architecture

2.1. The Layers of the IoT Model

In traditional TCP/IP networked environments, we map technologies, communication, and protocols to either the OSI reference model, or the TCP/IP stack (or both), since it helps us visualize communication. In 2014 Jim Green from Cisco published the Internet of Things Reference Model shown in figure 2. This reference model provides a nice visualization in the IoT space to better see where vulnerabilities/threats exist, and where intrusion detection can be valuable.

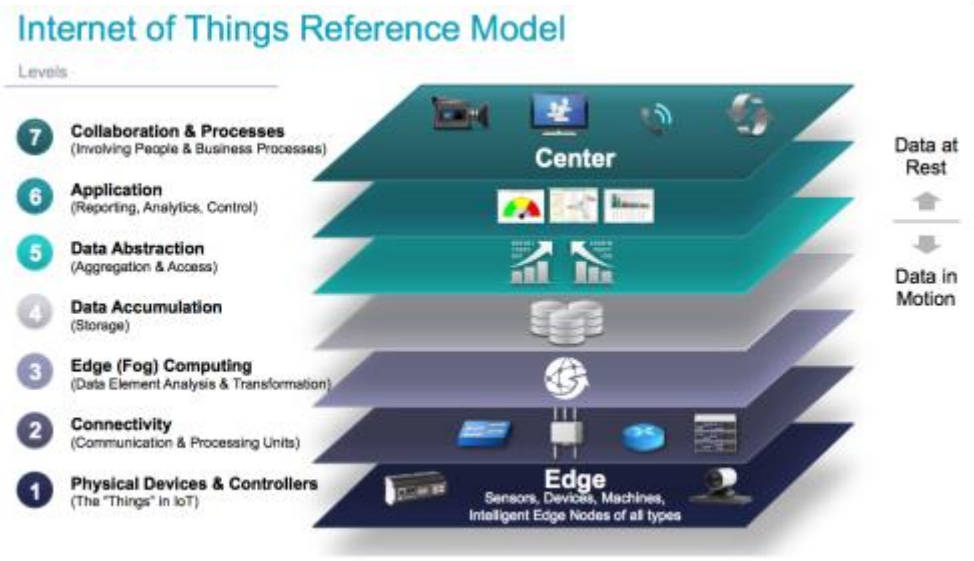


Figure 2 - The Internet of Things Reference Model (Green, 2014)

At layer 1, we can see that the physical layer is focused on the actual *things*, and include attacks against devices using interfaces like Joint Test Action Group (JTAG), Universal Asynchronous Receiver/Transmitter (UART), Serial Wire Debug (SWD), Serial Peripheral Interface (SPI) among others. Physical attacks can give an attacker access to the device console, provide control execution of firmware, leak firmware secrets such as hard-coded keys, passwords and more. JTAG and UART are both used for debugging (mainly for development and troubleshooting) and can be abused by an attacker with physical access to IoT devices. Access to these interfaces could permit an attacker to sniff and modify TDI/TDO signals, control TMS and TCK signals, and access keys used by testers (Rosenfeld & Karri, 2009).

The next layer – the connectivity layer - includes a multitude of protocols and technologies, such as traditional wireless local area networks (WiFi), personal area networks (Bluetooth, NFC, RFID, ZigBee, Z-Wave, 6LoWPAN), and cellular technologies (LTE, GPRS, CDMA, EVDO) and more (Postscapes, n.d.). Current wireless attacks affect protocols such as IEEE 802.15.4 based ZigBee and Z-Wave. "ZigBee is a mesh network specification for low-power wireless local area networks (WLANs) that cover a large area" (TechTarget, 2015). ZigBee attacks showcased at events such as BlackHat showed how to capture encryption keys by sniffing ZigBee devices joining the

network (Zillner, 2015). KillerBee[1], developed by Joshua Wright, allows one to "eavesdrop on ZigBee networks, replay traffic, attack cryptosystems and much more" (Offensive Security, 2014). Z-Wave similarly is described by the Z-Wave website as "a wireless radio frequency technology that lets smart devices talk to and connect with one another" (Z-Wave, n.d.). EZ-Wave is a popular tool that is used to attack Z-Wave, created by Joseph Hall and Ben Ramsey. "EZ-Wave uses commodity Software Defined Radio to exploit Z-Wave networks" (Szczys, 2016) and can be employed in a denial of service attack where Z-Wave-controlled lights are disabled (or destroyed).

The third layer refers to the edge devices and communication. This is where fog computing exists. Fog computing refers to a layer developed to bring the benefits of cloud closer to edge devices. This is to address the massive amounts of data that IoT devices can and potentially will send to the cloud to be processed. Multiply this by the growth of the IoT market, and fog suddenly makes a lot of sense. This also sets up an environment that could be highly beneficial with regards to intrusion detection in IoT environments. By implementing detection capability closer to edge devices, attacks can be contained and controlled much more effectively. Figure 3 below illustrates the fog computing communication flow.

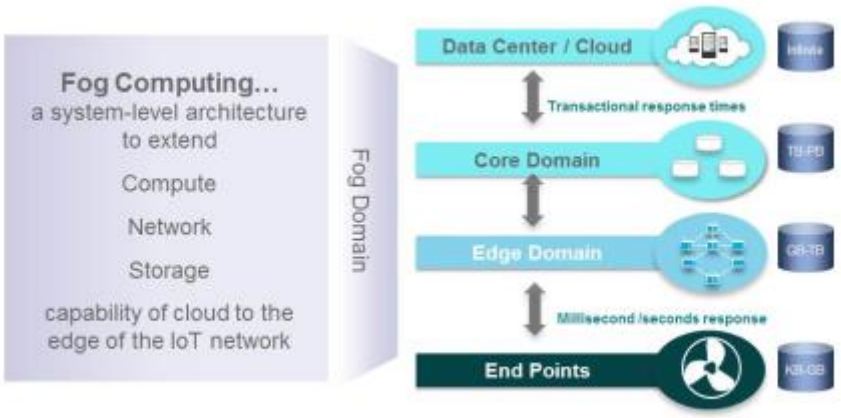


Figure 3 - Fog Computing Architecture (Kranz, 2015)

[1] https://code.google.com/p/killerbee/

The levels above fog computing (levels 4-7) can be assimilated into one application layer, for the purposes of this paper. At the application layer there are multiple points of intersection between the IoT and respective supporting application technologies. Smart devices connecting to the cloud use REST APIs, WebSockets, and HTTP/2 and are subject to traditional web application vulnerabilities and threats. IoT device management through app services and protocols such as Telnet, SSH, and HTTP/s introduce another set of attack vectors. Other common applications and protocols include MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), XMPP (Extensible Messaging and Presence Protocol), and Javascript/Node.js among others. Attacks against the application layer may target application data itself, the users of a device or service, or result in a bypass mechanism to attack lower layers (such as command injection or buffer overflows exploiting application weaknesses to take control of the physical device).

2.2. Common Threats and Vulnerabilities

There are well-documented sources available all over the internet that identify common threats and vulnerabilities. One of the more vetted and trusted sources is the Open Web Application Security Project (OWASP). The OWASP Internet of Things Project currently maintains a list of what it considers the top IoT vulnerabilities (Appendix A Table 3):

This provides insight into some vulnerabilities and security issues the industry is seeing. We can further enumerate this 'top vulnerabilities' list into a more comprehensive list of vulnerabilities and threats simply by mapping the attack surface and identifying recent attacks. The 'Device Network Services' and 'Network Traffic' categories of the OWASP IoT Attack Surface Areas list brings to light other types of attacks, specifically on IoT networking, that include routing attacks, Denial of Service (DoS) attacks, and Sybil attacks, among others (Nawir, Amir, Yaakob, & Bi Lynn, 2016).

2.3. Discovering New Threats / Vulnerabilities: Threat Modeling

Threat modeling is an important method to identify threats from a more strategic perspective. This usually involves several steps to methodically understand the system

being modeled which subsequently identifies potential threats. Microsoft Azure IoT lists the steps of threat modeling as shown in table 2 below.

Table 1 - Steps of Threat Modeling (Microsoft Azure, 2017)

Model the System	Enumerate Threats	Mitigate threats	Validate Mitigations

Ideally a given IoT application or solution would be broken down based on function and categorized as a device, a field gateway, a cloud gateway or a service. Each is separated by its own trust boundary and has separate requirements around authentication (AuthN) and authorization (AuthZ), data used etc, which will affect the threat model process. Once the system has been modeled, each component can be measured for threats using the STRIDE model. STRIDE is an acronym that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privileges. Attack trees can also be used to establish potential threats. Once the threats have been identified, an intrusion analyst can better understand what to implement and where.

3. IoT Intrusion Detection Requisites

Intrusion detection in IoT environments will generally fall into one of two areas; traditional IP networks and Low-power Wireless Personal Area Networks (LoWPANs). As with all network intrusion detection, both situations rely on a few things to be successful. This includes understanding the IoT communication stack, where weaknesses exist, the ability to capture traffic, and the ability to detect attacks in action.

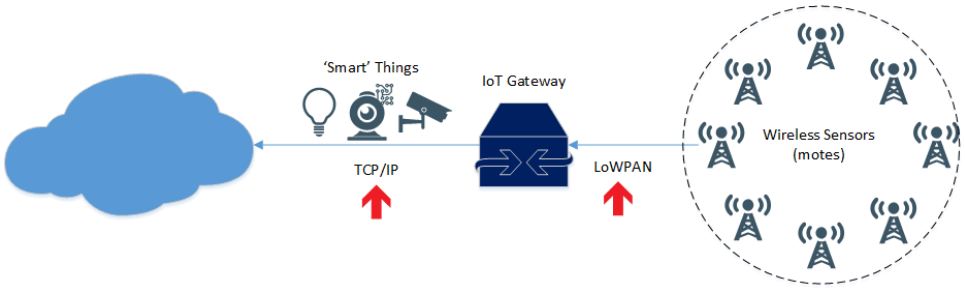


Figure 4 – Intrusion Detection In Traditional TCP/P Networks vs LoWPAN

3.1. Dissecting the IoT Communication Stack

IoT communication is a broad subject, as it covers a wide range of situations, devices, and general applications. Figure 3 illustrates the IoT communication stack and common IoT protocols. While the transport and network layers are common (TCP, UDP, IPv4, IPv6), the application and link layer protocols and technologies are a bit unique and might warrant further research.

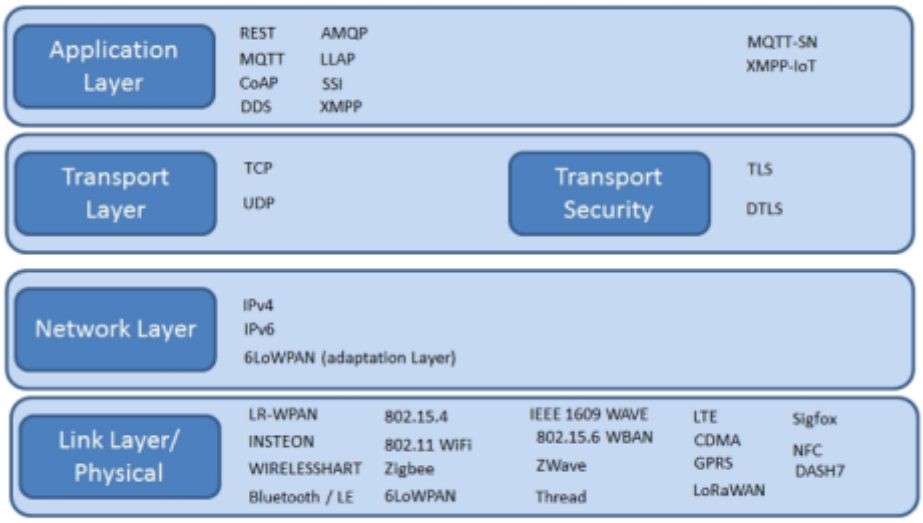


Figure 5 - IoT Communication Stack (Russell & Van Duren, 2016)

3.1.1. Some Common IoT Application Protocols

Message Queueing Telemetry Transport (MQTT), is a publish/subscribe based messaging protocol designed for M2M communications used in measuring and monitoring applications, such as weather stations. MQTT communicates over TCP/IP, relying on TCP as the transport protocol. Similar to the traditional client/server communication, MQTT uses publish/subscribe (pub/sub) to communicate between clients who are publishing with others that are subscribing. This is done using a MQTT broker, which effectively acts as a proxy. The broker decouples pub clients from sub clients, so the two have no knowledge of each other, but instead of the MQTT broker. This allows for messages to be sent with no knowledge of destination, and with no time constraints (pub/sub clients communicating do not need to run at the same time). MQTT client

messages are sent to the broker using a 'connect' packet, and responses from the broker to the client are sent using 'connack' packets.

Constrained Application Protocol (CoAP) is a web transfer protocol used in specialized IoT applications including smart energy and building automation. CoAP uses UDP as its transport protocol, so the application maintains knowledge of communication state and handles retries, errors, packet reordering etc. Unlike MQTT, CoAP operates in traditional client/server mode, using common HTTP methods GET, PUT, POST, and DELETE. Being a UDP based protocol, CoAP lends itself to constrained resources and can communicate via SMS or other packet-based communication protocols.

XMPP, the Extensible Messaging and Presence Protocol, initially began as a Jabber messaging protocol, and is currently used for a variety of collaboration, such as instant messaging, chat, voice/video calls, and general user-defined XML data. XMPP is an open, standards-based protocol, offering authentication and encryption. While XMPP is relatively popular, it has also had its fair share of vulnerabilities and weaknesses. The nmap network scanner includes script checks for XMPP (Karlsson, n.d.) to enumerate users. XMPPloit was developed to act as a gateway between client/server communication and compromise the communication between them.

3.1.2. Common Link Layer Protocols and Technologies

Bluetooth Low Energy (also known as Bluetooth LE or BLE) was designed specifically for the resource constrained devices that make up the IoT. Bluetooth LE uses small bursts of radio frequency (RF) communication which works for the IoT where continuous RF isn't required, but energy conservation is. Bluetooth LE can operate in two general modes. The first is a client/server type connection where one BLE device acts as a peripheral device and talks to another acting as a central device. An example would be a wireless sensor network; communication would be bi-directional between the two devices. The second mode is where a device operates in a unidirectional communication mode, either broadcasting data or listening for data (but not both). Peripheral devices, as well as devices operating as broadcasters, send beacon frames.

Figure 6 below illustrates a sample BLE frame.

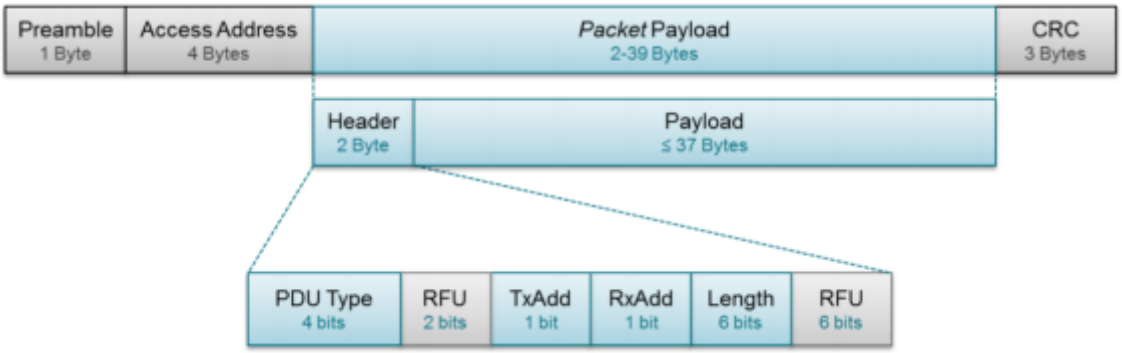


Figure 6 Bluetooth LE Data Packet (Texas Instruments, 2015)

Apple's iBeacon is an example of Bluetooth LE beacon technology. iBeacon was designed to facilitate retail environments, sending messages from iBeacon transmitters in stores, malls etc to consumers in an effort to enhance the user experience.

Similar to Bluetooth LE, ZigBee is a wireless mesh technology that was developed to address the low power RF communication required by IoT devices. ZigBee is based on the IEEE 802.15.4 specification, and spans the full stack from the link layer up to the application layer. ZigBee is commonly deployed in smart home technology, as well as the utility industry. The ZigBee 3.0 features include device-unique authentication, runtime key updates, secure firmware upgrades delivered over the air, and logical link-based encryption and replay protection. Figure 7 shows the 802.15.4 frame that ZigBee uses.

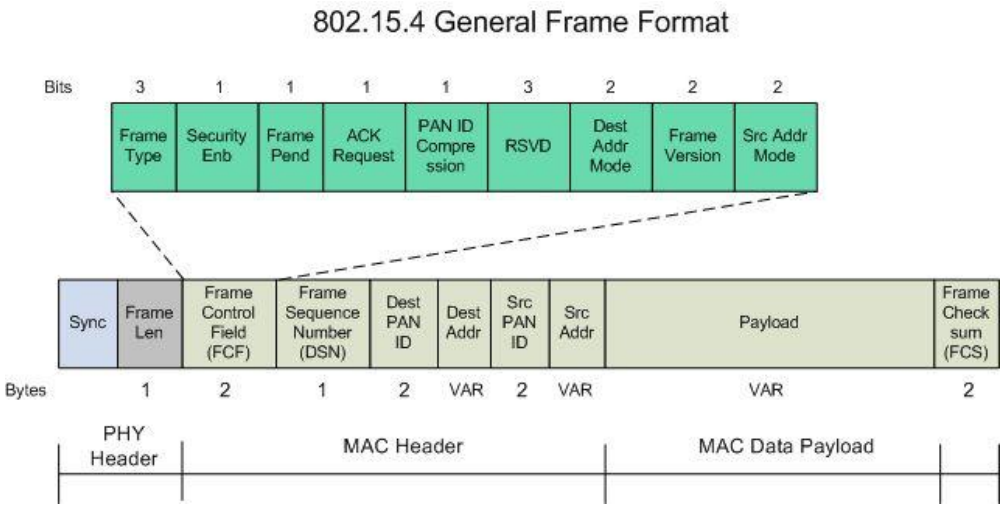


Figure 7 - IEEE 802.15.4 Frame Format

Adam Kliarsky, adam.kliarsky@gmail.com

3.2. Monitoring Communication

A fundamental dependency of network intrusion detection is the capability to monitor or sniff network traffic. While resources are abundant for traditional TCP/IP networks, this gets a bit tricky for the IoT. Tools like Wireshark have dissectors to understand a variety of protocols, however specialized radio frequency (RF) tools are needed to capture communications from and between some IoT devices

Bluetooth LE traffic can be monitored with Wireshark and a capable BLE adapter. Figure 8 shows a 'Bluefruit LE Friend' from Adafruit. Combined with Nordic BLE sniffer software, traffic can be sniffed and analyzed in Wireshark. Figure 8 also shows the detected BLE devices and wireless signal strength.

Figure 8 - Adafruit Bluefruit LE Friend USB Adapter, Nordic BLE Software Displaying Discovered Devices

Initial communication will display available BLE devices and corresponding RF signal information. Any one of the devices can be selected to focus sniffing, limiting the transactions displayed. Figure 7 below shows the BLE advertising beacons (PDU Type:

ADV_IND). What this provides an intrusion analyst is details into the packets that can be used for intrusion detection.

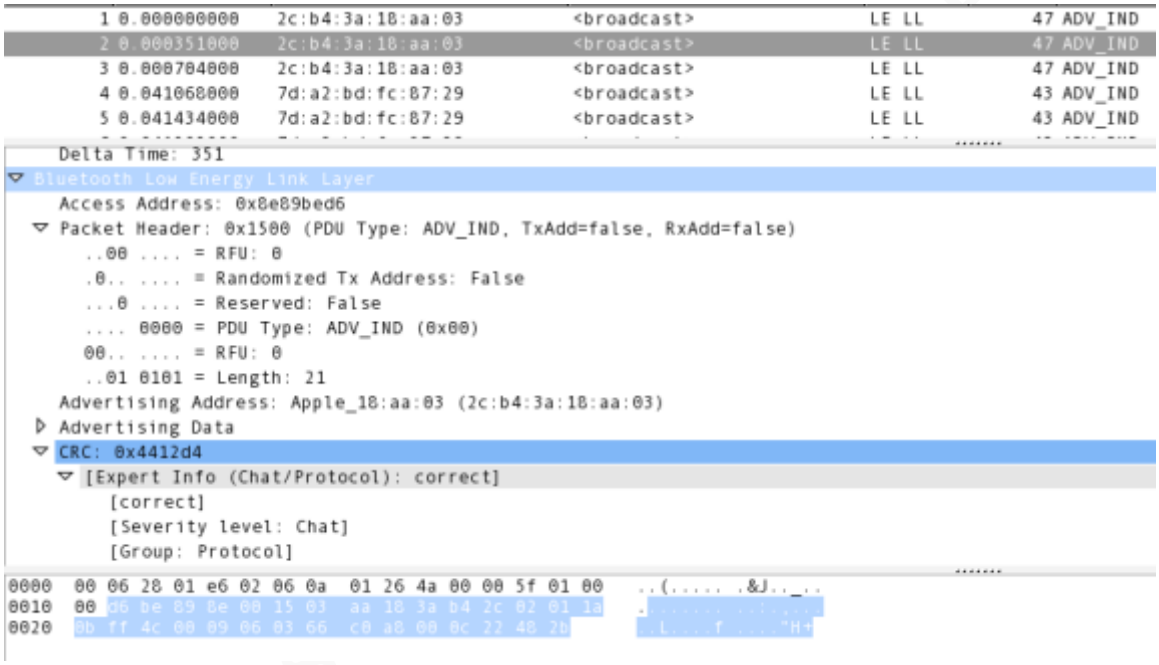
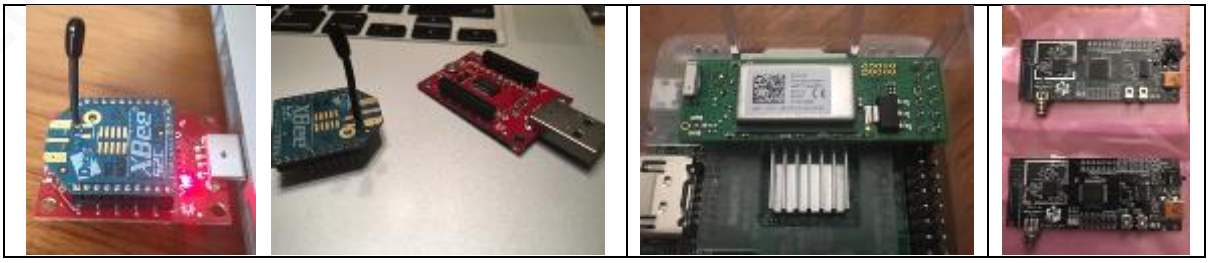


Figure 9 - Bluetooth LE Advertising Packet

Communication over IEEE 802.15.4 networks, including ZigBee and ZWave, can also be sniffed using a variety of tools. Examples below show the XBee, Raspbee on a RaspBerry Pi, and a pair of APIMOTEv4 devices used for ZigBee sniffing.

Table 2 – Xbee + Explorer, Raspbee Premium on a Raspberry Pi 3, and 2 Riverlooop Security APIMOTEv4 Devices



3.3. Intrusion Detection Systems for IoT Environments

Once the dependency of sniffing is addressed, and traffic can be captured, we can implement intrusion detection systems. There are several network intrusion detection solutions available when it comes to the Internet of Things, depending on the scenario.

Some popular tools include Snort, Bastille, BeeKeeper, and other LoWPAN based detection.

Snort

Snort is a popular open-source intrusion detection system that was initially developed by Marty Roesch in 1998, and currently maintained at Snort.org[2]. Snort is a robust intrusion detection system with a variety of capabilities, and can be deployed as a sniffer, packet logger, and intrusion detection system (Snort, n.d.). With its current capabilities Snort is poised to detect attacks across a variety of protocols with built-in preprocessors and rules. While Snort has layer 2 capability (for local segments), its strength is layer 3 and above – specifically the ability to inspect unencrypted packet payloads using ASCII or HEX content.

RPiDS: Raspberry Pi IDS

In July of 2016 researchers from the NEC Research laboratories in Heidelberg, Germany submitted a paper outlining an implementation of Snort on a Raspberry Pi they called RPiDS (Sforzin, Conti, Marmol, & Bohli, 2016). Figure 9 shows the high level architecture. This project was created specifically for an IoT deployment with portability and ease of use at its core. The RPiDS project was unique in that it was designed so that a Raspberry Pi (RPi) could be moved from one location to another, depending on the need. It was scalable, meaning more RPis could be added as needed, and could be run independently or collaborative, pulling data from other RPiDS sensors. By

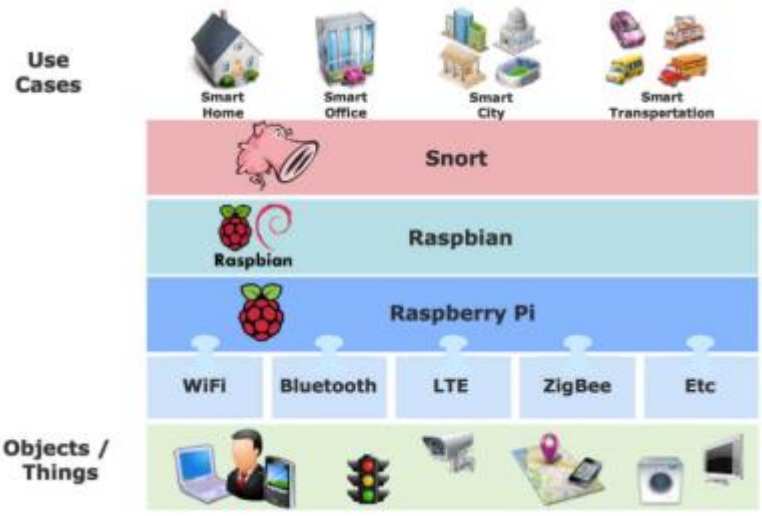


Figure 10 RPiDS Architecture (Sforzin, Conti, Marmol, & Bohli, 2016)

[2] https://www.snort.org/downloads

pulling traffic data from nearby RPiDS sensors, correlation could occur, reducing false positives, and providing the ability to detect attacks against network topology that might otherwise go unnoticed. Attacks can be localized and identified quickly in large public places.

BeeKeeper

BeeKeeper was a proof of concept project developed by River Loop Security, LLC and showcased at Troopers 14 in Heidelberg, Germany in March of 2014. BeeKeeper is an IEEE 802.15.4 based Wireless Intrusion Detection System (WIDS), built on top of the KillerBee framework. As such, it is designed to detect attacks in 802.15.4, ZigBee, and 6LoWPAN environments. BeeKeeper works by leveraging a 802.15.4 sniffer, such as the APIMOTEv4, to sniff the RF spectrum. This data is then sent via a drone (which manages the interfaces, channels etc) to the WIDS for processing. Figure 11 illustrates the KillerBee based BeeKeeper WIDS components.

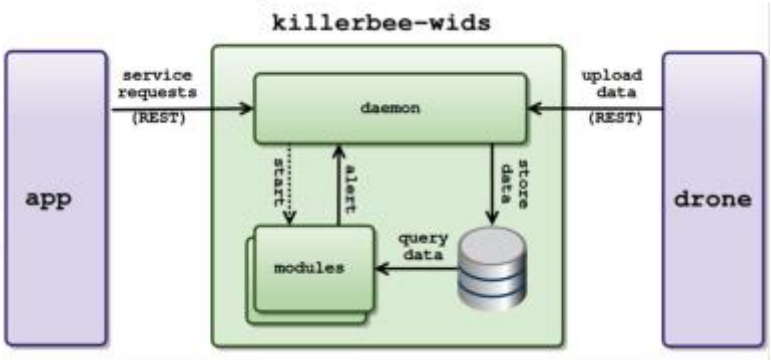


Figure 11 – BeeKeeper WIDS (Bratus, Speers, & Vazquez, 2014)

Bastille

Bastille Networks is a company specializing in "threat detection through software-defined radio" (Bastille, 2017). Bastille is a commercial product, and can scan frequencies from 60MHz to 6GHz to identify RF based threats to IoT networks. What makes Bastille stand out is their security research with regards to discovering both the MouseJack and KeyJack vulnerabilities, as well is their technology. The technology includes 'Collaborative Bandit Sensing' to monitor RF spectrum for attacks, 'Bayesian Device Fingerprinting' to identify different types of connected 'things' (and people with

things), and 'Distributed Tomographic Localization' to identify where devices are physically located (Bastille, 2017).

4. IoT Attacks and Detection

To better illustrate the detection capability, we can look at attacks against IoT environments and how intrusion detection can be used to identify attacks both in traditional IP networks, and in LoWPANs. Intrusion detection in IP based networks are fairly mature with standard detection mechanisms, however detection in wireless environments are less mature, and require a bit of tailoring for each scenario. We will look at a major attack in the IP network space with Mirai, and then some specific wireless attacks with ZigBee, 6LoWPAN, and Bluetooth LE.

4.1. IoT Botnets: Detecting a Mirai Compromise Attempt

A significant attack that affected connected devices on IP networks in 2016 was the Mirai botnet. The Mirai botnet targeted a group of connected devices using known default credentials, which illustrates the problem with the lack of standards based manufacturing and the rush to market development. The source code was released to the public (Anna-senpai, 2016) which provided intrusion analysts insight into the attack and associated intrusion detection.

4.1.1. The Attack

Of particular interest to intrusion analysts is the inbound scans to find and compromise vulnerable devices. This scanning function is initiated by existing bots which will conduct brute force scanning of IP addresses and then try to login by running through a list of known device credentials (Barker, 2016).

4.1.2. Intrusion Detection

From the intrusion analyst perspective, the published source code is of immense help. A cursory review of the 'scanner.c' file (Appendix B) shows some useful information. The file shows the setup of the network packet (IP and TCP header), and full list of all credentials to try on vulnerable devices. This is where Snort's content matching ability can be used to identify attacks based on payload data.

If we run 'strings' against a binary Mirai network capture file, we can see the unencrypted ASCII contents that are relevant, specifically the login attempts – which is indicative of the initial device compromise.

```
sn0rt:giac adam$ strings mirai.pcap | more
dvrdvs login: x
root
root
Password: x
xc3511x
```

Figure 12 – Issuing the 'strings' command showing the first lines of output.

The output results in over 4,000 human readable strings, however, the initial output show credentials listed in the 'scanner.c' source code file (Appendix B).

```
// Set up passwords
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                 // root     xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                      // root     vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                      // root     admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                  // admin    admin
add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                  // root     888888
```

Figure 13 - A Snippet of the Full Default Passwords Used

The value 'xc3511' can be found in Wireshark using the 'Find Packet' feature (Appendix B), providing details (figure 14) for the 'content' keyword and its modifiers.

```
▽ Telnet
    Data: xc3511

0000  8c e7 48 52 a0 de 00 e0  4c 68 df 56 08 00 45 20   ..HR.... Lh.V..E
0010  00 2e f5 8b 40 00 2d 06  1b de 5f be d1 1b 0a 80   ....@.-. .._.....
0020  00 e7 d8 60 00 17 76 11  e4 e7 03 0c cb c2 50 18   ...`..v. ......P.
0030  05 ac 8e d0 00 00 78 63  33 35 31 31               ......xc 3511
```

Figure 14 - Packet Content Showing Hex and ASCII Values

A rule will now look like figure 19 (or figure 20 if the hex values are used):

```
alert tcp any any -> any 23 (msg:"Mirai Test Rule – inbound login attempt";
flow:to_server,established; content:"xc3511";depth:6; sid:1000000;)
```

Figure 15 – Using ASCII Content

Using the rule with the ASCII match of *xc3511*, the following command (1) was executed, resulting in the subsequent alert (2) shown below:

```
(1) adam@malwarelab:~/giac$ sudo snort -c snort/etc/snort.conf -l log/ -r
mirai.pcap
--------------------------------------------------------------------------------
adam@malwarelab:~/giac$ cat log/alert
<snip>
(2) [**] [1:1000000:0] Mirai Test Rule – inbound login attempt [**]
[Priority: 0]
01/30-11:15:30.534758 192.168.1.25:12345 -> 192.168.10.5:23
TCP TTL:45 TOS:0x20 ID:54321 IpLen:20 DgmLen:46 DF
***AP*** Seq: 0x7611E4E7  Ack: 0x40DCBD2  Win: 0x5AC  TcpLen: 20
<snip>
```

Figure 16- Running Snort against the Mirai capture file

4.2. Detecting a DoS Attack Against ZigBee AES-CTR

4.2.1. The Attack

The AES-CTR security mode used by ZigBee was designed to provide AES encryption as well as replay protection. To protect against replay attacks, it inspects the frame counter to ensure that duplicate values don't exist. If a frame is received with a duplicate frame counter value, it is discarded. This leads to the DoS attack. If a crafted packet was sent by an attacker with a frame counter value of FFFFFFFF (the max value for the frame counter field), any frame received after that will be discarded. So an attacker could send this one packet, and essentially kill all subsequent communication.

```
scapy = Dot15d4(fcf_frametype="Data")/Dot15d4Data()
scapy.seqnum = (args.seqnum + randint(1, 10)) % 255
scapy.src_addr = args.source
scapy.dest_addr = args.destination
scapy.src_panid = scapy.dest_panid = args.panid
print "DoSing packets from sender 0x%04x to destination 0x%04x." % (scapy.src_addr, scapy.dest_addr)

# Weaponize this frame for the DoS Attack on AES-CTR
scapy.fcf_security = True
sechdrtemp = scapy.aux_sec_header #TODO oddly having issue w/ doing directly to main Dot15d4
sechdrtemp.sec_sc_seclevel = "ENC"       #confidentiality but no integrity
sechdrtemp.sec_framecounter = 0xFFFFFFFF #max value
sechdrtemp.sec_sc_keyidmode = "1oKeyIndex"
sechdrtemp.sec_keyid_keyindex = 0xFF     #max value
scapy.aux_sec_header = sechdrtemp
```

Figure 17 - Attack Code for the AES-CTR DoS (River Loop Security, 2014)

4.2.2. Intrusion Detection

Using the BeeKeeper WIDS framework, with two APIMOTEv4 devices, we can detect the attack. The APIMOTEs capture traffic, and that traffic is sent to the drone

modules. The drones in turn interact with the WIDS (Wireless Intrusion Detection System) module. The WIDS manages the drones which manage the interface for channel selection, packet capture, and does all of the analyzing of captured packets to determine which are malicious. The code in figure 22 and 23 illustrate part of the WIDS module that will detect the AES-CTR DoS attack.

```
# Task drones to capture packets which would cause a denial of service
# to systems using AES-CTR security mode due to sending a frame counter
# set to a maximal (or near maximal) value.
parameters = {'callback': self.config.upload_url,
              'filter'  : {
                 'fcf': (0x0b00, 0x0900), # fcf_frametype = data && fcf_security = true
                 'byteoffset': (11, 0xff, 0xff) # 090801ffffffff00 000000ff db2d
              }}
```

Figure 18 - Instructions to Drone Modules to Monitor Traffic (River Loop Security, 2014)

```
# Get packets from database and run statistics
while not self.shutdown_event.is_set():
    pkts = self.getPackets(uuidFilterList=[uuid_task1], new=True)
    self.logutil.debug("Found {0} packets since last check.".format(len(pkts)))

    for pkt in pkts:
        spkt = Dot15d4FCS(pkt.pbytes)   # scapify the packet's bytes
        if spkt.fcf_security != True:
            self.logutil.log("Packets are excepted to have 802.15.4 security for this analytic.")
        elif spkt.aux_sec_header.sec_framecounter > 0xFF000000:
            self.logutil.log('Generating Alert: HighFrameCounterDetection')
            self.generateAlert('HighFrameCounterDetection')
            self.registerEvent(name='HighFrameCounterDetection',
                               details={'channel':channel, 'pkt':pkt})
```

Figure 19 - Detection Code Snippet from WIDS AES-DOS Module (River Loop Security, 2014)

As these frames are detected, as the code above notes, the message "HighFrameCounterDetection" will display to indicate the attack.

4.3. Detecting Routing Attacks in 6LoWPAN Networks

6LoWPAN stands for IPv6 over Low Power Wireless Personal Area Networks and exists to bridge the gap between IP networks and Low-Power and Lossy Networks (LLNs). The RPL (Routing Protocol for Low-Power and Lossy Networks) protocol, which is used by 6LoWPAN, is vulnerable to routing attacks, including selective forwarding, Sybil, sinkhole, and wormhole attacks among others. For the sake of brevity, the sinkhole and wormhole attacks are illustrated in this paper.

4.3.1. The Attacks

Sinkhole attacks are highly destructive routing attacks in 6LoWPAN environments. In these attacks, a malicious node announces to nearby nodes that it knows the shortest communication path, or route, to a destination. By controlling traffic in a given area of a 6LoWPAN network, it can manipulate traffic, and discard packets thereby disrupting communication altogether.

Wormhole similarly works by attacker advertising to neighbors. This method uses two nodes to establish a tunnel of sorts (hence the term wormhole) and can be used to legitimately route messages faster based on priority. These nodes can both exist in one 6LoWPAN environment, or more dangerously, between one node in the 6LoWPAN network, and another remote node beyond the 6LoWPAN border router (6BR).

4.3.2. Intrusion Detection

For the sinkhole attack, a tool called INTI (**I**ntrusion Detection for Si**N**khole Attacks Over 6LoWPAN for Interne**T** of Th**I**ngs) was developed to not only detect sinkhole attacks, but isolate the affected nodes using node reputation, watchdog, and trust that member nodes develop and communicate amongst each other. INTI uses four modules for intrusion detection; cluster configuration, rout monitoring, attack detection and attack isolation (Cervantes, Poplade, Nogueira, & Santos, 2015). The configuration of clusters is a hierarchical configuration whereby nodes establish themselves as members, leaders etc based on communication. The route monitoring checks incoming streams to outgoing streams to see if there are any irregularities. A 1:1 relationship would indicate normal behavior, however if a node has more incoming streams than outgoing streams that is a sign of a possible sinkhole attack. This communication establishes hierarchy of node relationships, as well as reputation based trust among the nodes. INTI uses these values for intrusion detection. Figure 20 illustrates in (a) an associated node marked 'S' that becomes a sinkhole attacker node, and (b) nodes n14 + n6 detect the attack. Attack detection includes identifying the malicious node and sending the

reconstruction notification to isolate the attacker and rebuild/re-establish routes.

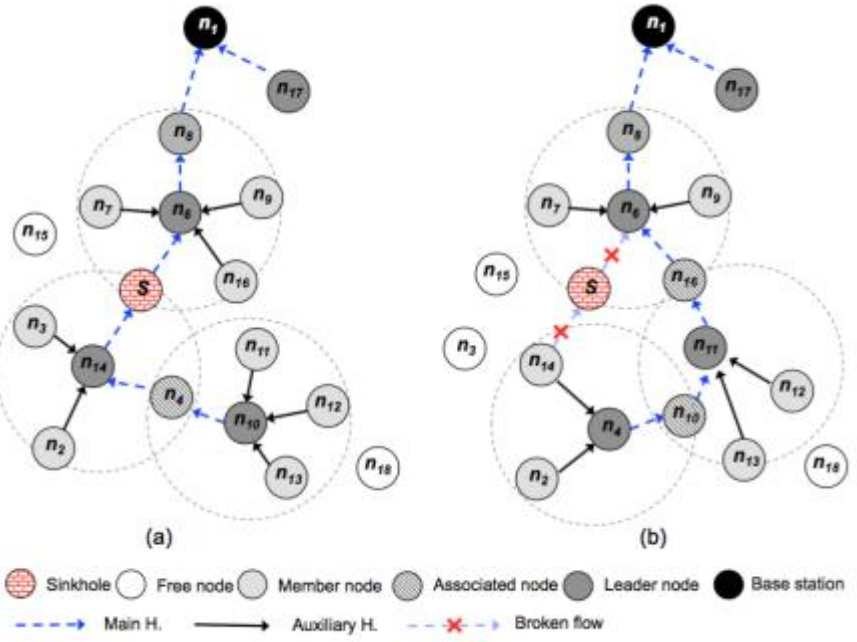


Figure 20 - Sinkhole Detection

Wormhole attacks can be detected using similar IDS technology (route and node communication), only using a distributed mechanism leveraging the 6BR gateway for an IDS the and wireless sensor nodes for distributed detection. Since the 6BR will be able to handle more tasks and system resources than wireless nodes, advanced capabilities such as anomaly detection can be implemented on the 6BR. The 6BR acts as the centralized node and analyzes neighbor node placement, RSSI levels, and ultimately attack detection. Similarly, the distributed nodes monitor node communication, and send that back to the 6BR. The RSSI levels are also collected and sent back to the 6BR which will ultimately make the decision of whether an attack is in progress or not. Figure 21 shows the establishment of a wormhole between nodes 10 and 13. As nearby nodes advertise this change, the detection module of the 6BR will recognize that routes advertised as "close" should actually be out of range, and thereby an attack is in progress (Pongle & Chavan, 2015).

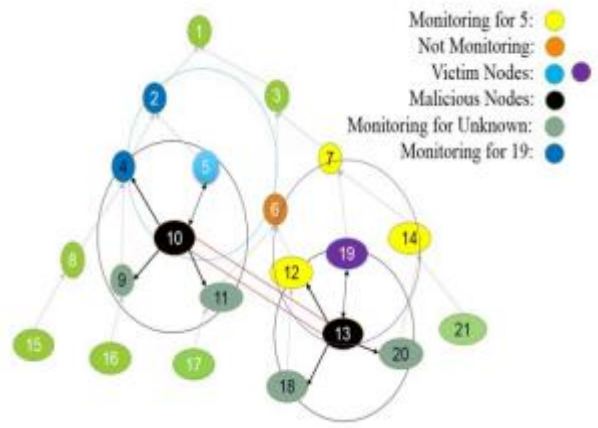


Figure 21 – Wormhole Attack Between Nodes 10 + 13 (Pongle & Chavan, 2015)

A promising solution that was developed for general DoS style attacks in 6LoWPAN environment was using Suricata. The architecture deploys a Suricata IDS in a 6LoWPAN environment, with distributed IDS probes on wireless nodes communicate to the Suricata IDS, which is then able to forward alerts to a SIEM (figure 22).

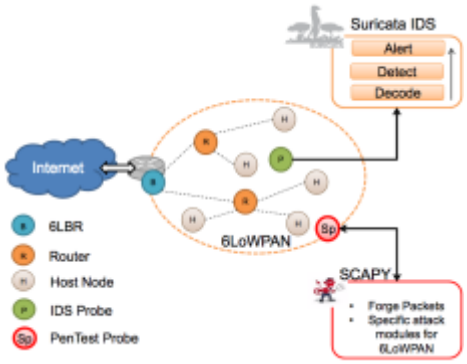


Figure 22 - 6LoWPAN Intrusion Detection with Suricata

5. Future Considerations and Challenges

Other considerations that will affect the IoT IDS deployments include encryption, IPv6, scalability and management, and deployment complexity.

Deploying intrusion detection systems to identify attacks at layer three and above will have challenges. Intrusion detection relies on the ability to inspect packet payloads to search for content, and as more applications leverage encryption to maintain security and privacy, this will cause blindness in monitoring. To deal with encryption, there needs to

be some type of key offload mechanism, where packets are decrypted and inspected. This can be costly and complex, and is a contentious issue regarding assumed privacy.

The depletion of IPv4 and explosive growth of the IoT means more packets will be traversing the network with IPv6. For IDS deployments where teams develop their own signatures, this means extending the skillset to include support for IPv6 signatures. Snort, for example, does support IPv6, so custom signatures can leverage the different options that an IPv6 packet supports. Figure 23 below illustrates the differences.

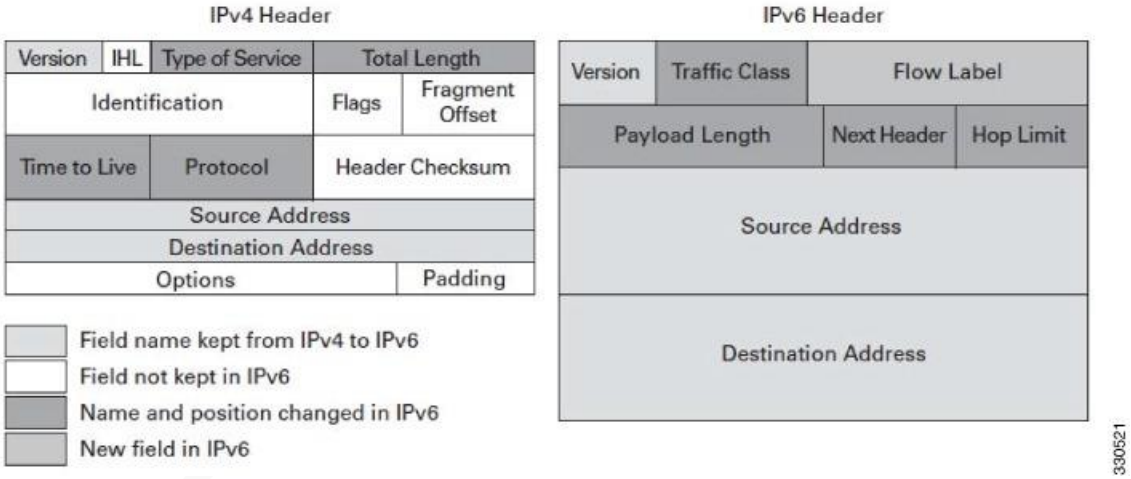


Figure 23 – Comparison of the IPv4 Header to the IPv6 header (Cisco, 2013)

The deployment of intrusion detection in IoT environments will vary based on the use case. For radio-based intrusion detection systems, the limitation will be distance and RF limitations. As the environment grows, consideration needs to be made to address how the deployment will scale and subsequently be managed. This is an issue for placement between wireless sensors and gateways, as well as between gateways and core TCP/IP infrastructure (depending on the deployment).

In some circumstances the complexity to an IoT deployment will create challenging considerations with intrusion detection. Fog environments where wireless sensors connect to a fog server or gateway, and then that gateway communicates back to the core network over a cellular connection is one example. In this situation there isn't a separation between administrative traffic and transactional traffic. Unless an administrative connection is established, alerts generated in the fog layer will need to

communicate on the same link as transactional traffic. Ultimately IDS placement will depend on what actual areas of risk exist, based on a threat model of the deployment.

6. Conclusion

The smart, connected, 'Internet of Things', market continues to grow at a rapid rate. Many new and existing vendors are developing a myriad of new products and technologies to meet market demand and stay on the cutting edge of technology. This has implications on security that stems from manufacturing and extends to operations. The rush of manufacturers to push product to market leaves security in the product development lifecycle questionable. The lack of standardization further exacerbates this problem, as manufacturers develop and implement technology (and security) as they see fit. In current traditional IP based computing environments, we have endpoint management (patch, vulnerability, threat) and intrusion detection that augment daily operations. For the IoT this is a bit tricky due to resource constrained devices and networks supporting them. Intrusion analysts need to be able to dissect and understand the technology and communication in use, and conduct threat analysis (and modeling) to determine where risk exists. Intrusion detection can then be deployed appropriately to identify attacks that affect assets being defended and monitored.

Adam Kliarsky, adam.kliarsky@gmail.com

References

Anna-senpai. (2016, September 30). *[FREE] World's Largest Net:Mirai Botnet, Client, Echo Loader, CNC source code release*. Retrieved from HackForums.net: https://hackforums.net/showthread.php?tid=5420472

Ashok, I. (2016, August 31). *One million IoT devices infected by Bashlite malware-driven DDoS botnet*. Retrieved from Yahoo! News: https://uk.news.yahoo.com/one-million-iot-devices-infected-081813684.html

Barker, C. (2016, October 5). *Mirai (DDoS) Source Code Review*. Retrieved from Medium: https://medium.com/@cjbarker/mirai-ddos-source-code-review-57269c4a68f#.xe6q7l6ee

Bastille. (2017). *Bastille Security for the Internet of Radios*. Retrieved from Bastille Networks: http://www.bastille.net/mission/

Bratus, S., Speers, R., & Vazquez, J. (2014, March 19). *Making (and Breaking) an 802.15.4 WIDS*. Retrieved from Troopers: https://www.troopers.de/events/troopers14/24_making_and_breaking_an_802154_wids/

Cervantes, C., Poplade, D., Nogueira, M., & Santos, A. (2015). Detection of Sinkhole Attacks for Supporting Secure Routing on 6LoWPAN for Internet of Things. Brazil.

Cisco. (2013, October 29). *Routing and Bridging Guide vA5(1.0), Cisco ACE Application Control Engine*. Retrieved from Cisco: http://www.cisco.com/c/en/us/td/docs/interfaces_modules/services_modules/ace/vA5_1_0/configuration/rtg_brdg/guide/rtbrgdgd/ipv6.html

Green, J. (2014, November 4). *https://www.iotwf.com/resources*. Retrieved from Internet of Things World Forum: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf

Hayes, E. (2016, October 18). *The Underwhelming Security Of The Internet Of Things (IoT)*. Retrieved from Jacobian Engineering Blog: https://jacobianengineering.com/blog/2016/10/underwhelming-security-internet-things/

Hilton, S. (2016, October 21). *Company News*. Retrieved from Dyn: http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

Kan, M. (2016, October 18). *Hackers create more IoT botnets with Mirai source code*. Retrieved from PCWorld: http://www.pcworld.com/article/3132571/hackers-create-more-iot-botnets-with-mirai-source-code.html

Karlsson, P. (n.d.). *xmpp-brute*. Retrieved from NMAP: https://nmap.org/nsedoc/scripts/xmpp-brute.html

Knudson, J. (2015, May 13). *Security Challenges of the Internet of Things*. Retrieved from Enterprise Networking Planet:

http://www.enterprisenetworkingplanet.com/netsecur/security-challenges-of-the-internet-of-things.html

Kranz, M. (2015, August 10). *Fog Computing: Bringing Cloud Capabilities Down to Earth*. Retrieved from Cisco Blogs: http://blogs.cisco.com/digital/fog-computing-bringing-cloud-capabilities-down-to-earth

Krebs, B. (2016, September 21). *KrebsOnSecurity Hit With Record DDoS*. Retrieved from KrebsOnSecurity: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

Laliberte, M. (2016, October 28). *Understanding IoT botnets*. Retrieved from helpnetsecurity.: https://www.helpnetsecurity.com/2016/10/28/understanding-iot-botnets/

Level 3 Threat Research Labs. (2016, August 25). *Attack of Things!* Retrieved from Beyond Bandwidth Level 3 Communications Blog: http://blog.level3.com/security/attack-of-things/

marketsandmarkets.com. (2016, September). *Internet of Things Technology Market by Hardware (Processor, Sensor, Connectivity Technology), Platform (Device Management Platform, Application Management Platform, Network Management Platform) Software Solutions, and Services, Application, and Geography - Forecast to 2022*. Retrieved November 3, 2016, from Markets and Markets: http://www.marketsandmarkets.com/Market-Reports/iot-application-technology-market-258239167.html

Microsoft Azure. (2017, January 4). *Internet of Things security architecture*. Retrieved from Microsoft Azure: https://docs.microsoft.com/en-us/azure/iot-suite/iot-security-architecture

Nawir, M., Amir, A., Yaakob, N., & Bi Lynn, O. (2016, August). Internet of Things (IoT): Taxonomy of Security Attacks. Phuket,, Thailand.

Offensive Security. (2014, February 18). *KillerBee*. Retrieved from Kali Tools: http://tools.kali.org/wireless-attacks/killerbee

OWASP. (2017, February 9). *OWASP Internet of Things Project*. Retrieved from OWASP: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Vulnerabilities

OWASP Internet of Things Project. (2017, January 18). *IoT Vulnerabilities Project*. Retrieved from OWASP Internet of Things Project: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Vulnerabilities

Pongle, P., & Chavan, G. (2015, July). Real Time Intrusion and Wormhole Attack Detection in Internet of Things. Pune, India.

Postscapes. (n.d.). *Tracking the Internet of Things*. Retrieved from Postscapes: http://postscapes2.webhook.org/internet-of-things-technologies

River Loop Security. (2014, March 19). *riverloopsec/beekeeperwids.* Retrieved from GitHub: https://github.com/riverloopsec/beekeeperwids/blob/master/demo/dos_aesctr_replay.py

Adam Kliarsky, adam.kliarsky@gmail.com

Rosenfeld, K., & Karri, R. (2009, August 12). *Attacks and Defenses for JTAG.* Retrieved from Verifying Physical Trustworthiness of ICs and Systems: http://isis.poly.edu/~kurt/papers/design_and_test_final.pdf

Russell, B., & Van Duren, D. (2016). *Practical Internet of Things Security.* Birmingham, UK: Packt Publishing.

Sforzin, A., Conti, M., Marmol, F. G., & Bohli, J.-M. (2016, July 18-21). *RPiDS: Raspberry Pi IDS A Fruitful Intrusion Detection System for IoT.* Retrieved from IEEE Xplore: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7816876

Snort. (n.d.). *Snort Users Manual*. Retrieved from Snort: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/

Szczys, M. (2016, January 16). *SHMOOCON 2016: Z-WAVE PROTOCOL HACKED WITH SDR*. Retrieved from HACKADAY: http://hackaday.com/2016/01/16/shmoocon-2016-z-wave-protocol-hacked-with-sdr/

TechTarget. (2015, March). *ZigBee*. Retrieved from Internet of Things Agenda: http://internetofthingsagenda.techtarget.com/definition/ZigBee

Texas Instruments. (2015, January). *Bluetooth® Low Energy Beacons*. Retrieved from Texas Instruments: http://www.ti.com/lit/an/swra475a/swra475a.pdf

Trustwave. (n.d.). *Trustwave* . Retrieved from Internet of Things Security: https://www.trustwave.com/Solutions/By-Challenge/Internet-of-Things-Security/

Verizon. (2016, April 5). *State of the Market: Internet of Things 2016*. Retrieved November 13, 2016, from Verizon: https://www.verizon.com/about/sites/default/files/state-of-the-internet-of-things-market-report-2016.pdf

Zillner, T. (2015, August 6). *ZigBee Exploited the Good, the Bad, and the Ugly.* Retrieved from Black Hat USA 2015: https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf

Z-Wave. (n.d.). *Z-Wave: The Basics*. Retrieved from Z-Wave: http://www.z-wave.com/faq

Adam Kliarsky, adam.kliarsky@gmail.com

Appendix A

Vulnerability	Attack Surface	Summary
Username Enumeration	• Administrative Interface • Device Web Interface • Cloud Interface • Mobile Application	• Ability to collect a set of valid usernames by interacting with the authentication mechanism
Weak Passwords	• Administrative Interface • Device Web Interface • Cloud Interface • Mobile Application	• Ability to set account passwords to '1234' or '123456' for example.
Account Lockout	• Administrative Interface • Device Web Interface • Cloud Interface • Mobile Application	• Ability to continue sending authentication attempts after 3 - 5 failed login attempts
Unencrypted Services	• Device Network Services	• Network services are not properly encrypted to prevent eavesdropping by attackers
Two-factor Authentication	• Administrative Interface • Cloud Web Interface • Mobile Application	• Lack of two-factor authentication mechanisms such as a security token or fingerprint scanner
Poorly Implemented Encryption	• Device Network Services	• Encryption is implemented however it is improperly configured or is not being properly updated, e.g. using SSL v2
Update Sent Without Encryption	• Update Mechanism	• Updates are transmitted over the network without using TLS or encrypting the update file itself
Update Location Writable	• Update Mechanism	• Storage location for update files is world writable potentially allowing firmware to be modified and distributed to all users
Denial of Service	• Device Network Services	• Service can be attacked in a way that denies service to that service or the entire device

Adam Kliarsky, adam.kliarsky@gmail.com

Removal of Storage Media	• Device Physical Interfaces	• Ability to physically remove the storage media from the device
No Manual Update Mechanism	• Update Mechanism	• No ability to manually force an update check for the device
Missing Update Mechanism	• Update Mechanism	• No ability to update device
Firmware Version Display and/or Last Update Date	• Device Firmware	• Current firmware version is not displayed and/or the last update date is not displayed
Firmware and storage extraction	• JTAG / SWD interface • In-Situ dumping • Intercepting a OTA update • Downloading from the manufacturers web page • eMMC tapping • Unsoldering the SPI Flash / eMMC chip and reading it in a adapter	• Firmware contains a lot of useful information, like source code and binaries of running services, pre-set passwords, ssh keys etc.
Manipulating the code execution flow of the device	• JTAG / SWD interface • Side channel attacks like glitching	• With the help of a JTAG adapter and gdb we can modify the execution of firmware in the device and bypass almost all software based security controls. • Side channel attacks can also modify the execution flow or can be used to leak interesting information from the device
Obtaining console access	• Serial interfaces (SPI / UART)	• By connecting to a serial interface, we will obtain full console access to a device • Usually security measures include custom bootloaders that prevent the attacker from entering single user mode, but that can also be bypassed.

Table 3 - The OWASP Top IoT Vulnerabilities (OWASP, 2017)

Appendix B

Adam Kliarsky, adam.kliarsky@gmail.com

7. Determining Content Match and Modifiers

The source file for the bot scanner function shows data that will be present in unencrypted network traffic. This shows us two important points; 1) part of the Snort header; it is a tcp based packet on destination port 23 and 2) the content to match on.

```
iph = (struct iphdr *)scanner_rawpkt;
tcph = (struct tcphdr *)(iph + 1);

// Set up IPv4 header
iph->ihl = 5;
iph->version = 4;
iph->tot_len = htons(sizeof (struct iphdr) + sizeof (struct tcphdr));
iph->id = rand_next();
iph->ttl = 64;
iph->protocol = IPPROTO_TCP;

// Set up TCP header
tcph->dest = htons(23);
tcph->source = source_port;
tcph->doff = 5;
tcph->window = rand_next() & 0xffff;
tcph->syn = TRUE;

// Set up passwords
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                // root     xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                     // root     vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                     // root     admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                 // admin    admin
add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                 // root     888888
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);             // root     xmhdipc
add_auth_entry("\x50\x4D\x4D\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);             // root     default
add_auth_entry("\x50\x4D\x4D\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);         // root     juantech
add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17\x14", 5);                 // root     123456
add_auth_entry("\x50\x4D\x4D\x56", "\x17\x16\x11\x10\x13", 5);                     // root     54321
add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5); // support  support
add_auth_entry("\x50\x4D\x4D\x56", "", 4);                                         // root     (none)
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4);     // admin    password
add_auth_entry("\x50\x4D\x4D\x56", "\x50\x4D\x4D\x56", 4);                         // root     root
```

Figure 24 - Snippet of 'scanner.c' Showing IP and TCP header details, as well as some payload data.

As previously seen with the mirai.pcap file, the password xc3511 is seen in the strings output. This can be found using Wireshark's "find packet" feature. Once the packet is identified, details about the packet can be gleaned from the packet payload.

```
sn0rt:giac adam$ strings mirai.pcap | more
dvrdvs login: x
root
root
Password: x
xc3511x
```

Figure 25 – Issuing the 'strings' command showing the first lines of output.

The output results in over 4,000 human readable strings, however, the initial output show credentials listed in the 'scanner.c' source code file (Appendix B).

```
// Set up passwords
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                // root     xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                     // root     vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                     // root     admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                 // admin    admin
add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                 // root     888888
```

Figure 26 - A Snippet of the Full Default Passwords Used

Use Wireshark's 'find packet' feature by going to (Edit → Find Packet) and doing a search on the string value 'xc3511' as shown below

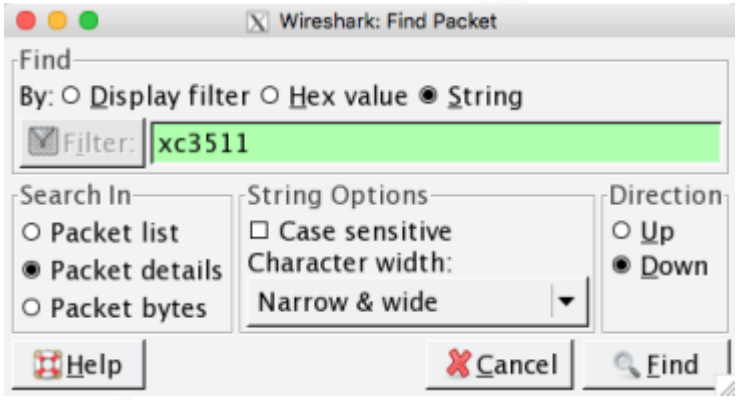


Figure 27 - Using Wireshark's 'Find Packet' Feature

This will find the first match of the content we are looking for, which is all that is needed for this particular rule.

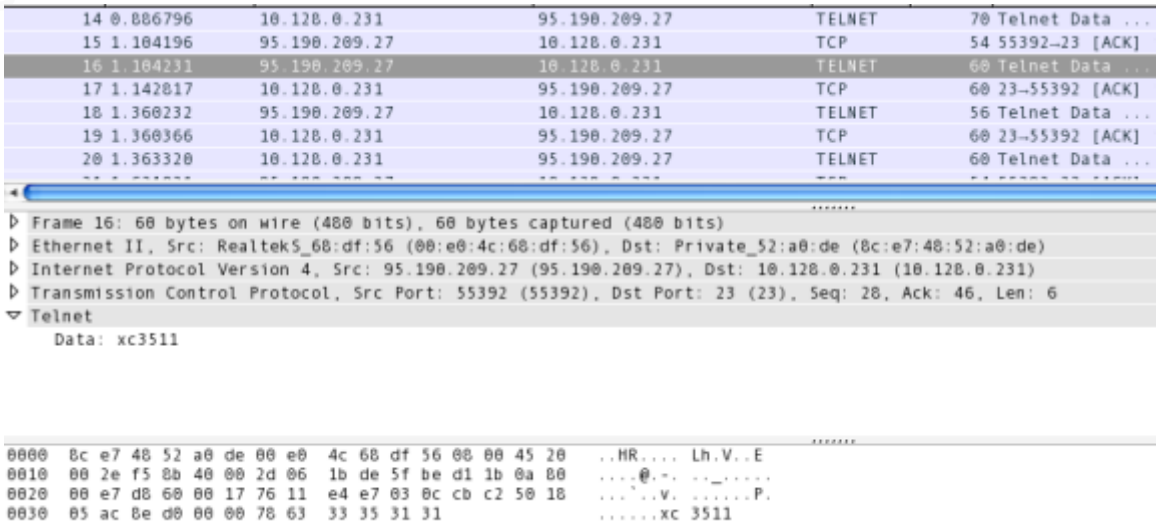


Figure 28 - Wireshark's 'Find Packet' Identifies Packet of Interest

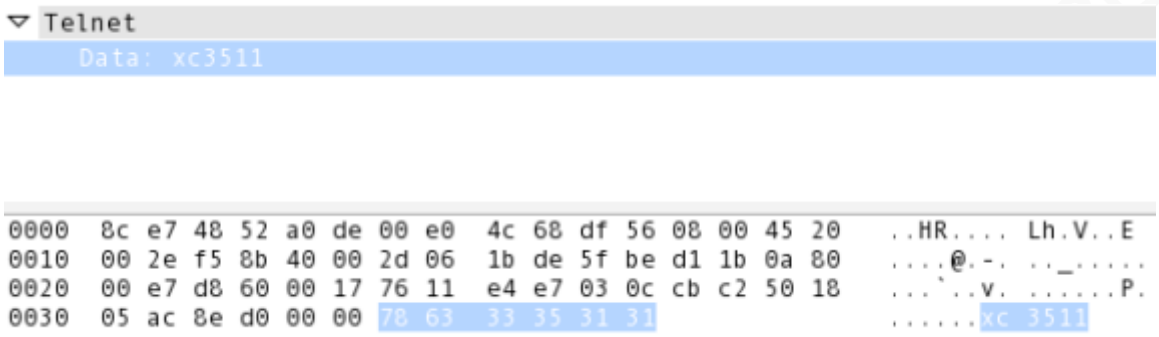


Figure 29 - Packet Details

The rule header is relatively trivial; we want to alert on the content, between any source (any) and internal destination ($HOME_NET). Source port will also be any, and destination port will be 23/tcp. The header then will be 'alert tcp any any -> $HOME_NET 23'. In the rule options, the 'content' keyword will be used to identify 'xc3511'. Content options will be used to specify where exactly the content exists, to increase efficiency of the rule.

'Content' *modifier* keywords further enhance the detection by specifying details about the expected content. These specifics increase detection accuracy and save on system resources. Some standard modifiers include:

- **'offset'** – defines an offset from within the data payload to start searching, but doesn't specify where to stop. An offset of 0 would be at the beginning of the data payload and is implied unless otherwise stated.
- **'depth'** – goes with 'offset'; defines in bytes how far into a packet to start searching from the offset point. A depth of 4 would mean search for specified content in 4 bytes from the defined offset (or offset 0 if not defined). Content that exists after the 4 bytes would be ignored.
- **'distance'** – this is used to define a second content match, and defines how long, in bytes, after the previous match was identified to start searching. A distance of 0 would be immediately after the first content match occurred.
- **'within'** – this acts on 'distance' above, saying that once searching begins after a defined 'distance', only do so 'within' a certain number of bytes.
- **'nocase'** – this tells Snort to ignore the case of the content keyword
- **'rawbytes'** – permits content inspection based on non-normalized traffic
- **'pcre'** – enables the use of Perl Compatible Regular Expressions (PCRE)

Content modifier keywords also provide the ability to specific multiple content matches and in what order. For example, given an HTTP GET request, where there are multiple parts, multiple 'content' matches can be used together to form a specific signature. Figure 8 below shows three elements in the GET request method that can be matched; "GET + /description.xml + HTTP/1.1".

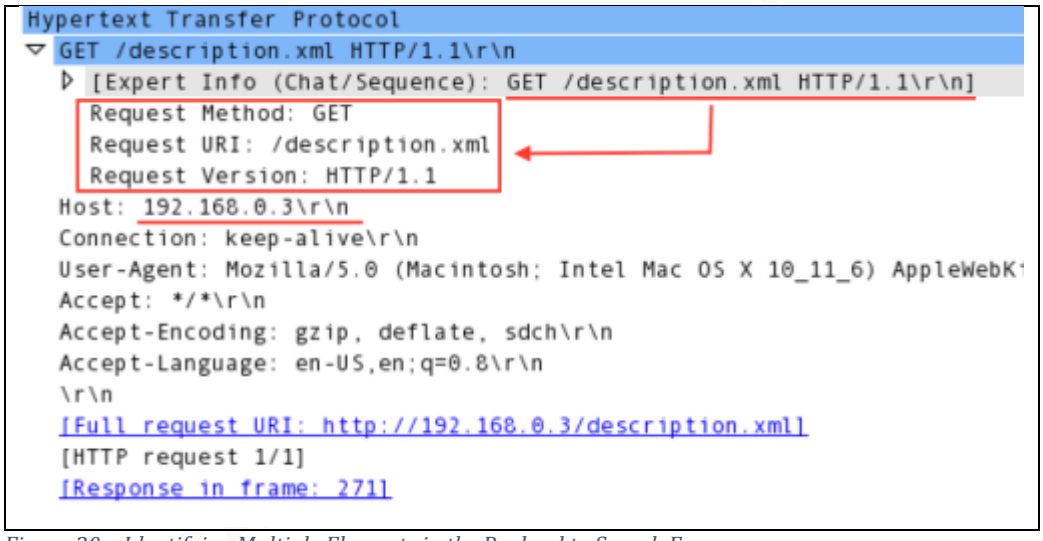


Figure 30 – Identifying Multiple Elements in the Payload to Search For.

Viewing the packet in Wireshark, the exact location in the payload and size of content can be determined, resulting in 'offset', 'depth', 'distance', and 'within' content keyword modifier values. The offset value is '0', since the GET parameter is at the beginning of the data payload and is three bytes long. Since the offset is 0, it is implied and not required. There is a one-byte spacer between 'GET' and the next match of 'description.xml', which is sixteen bytes long. This puts 'distance' at one and 'within' at sixteen. There is one-byte before 'HTTP/1.1', which is eight bytes, putting 'distance' at one, and 'within' at 8. Figure 28 illustrates the values.

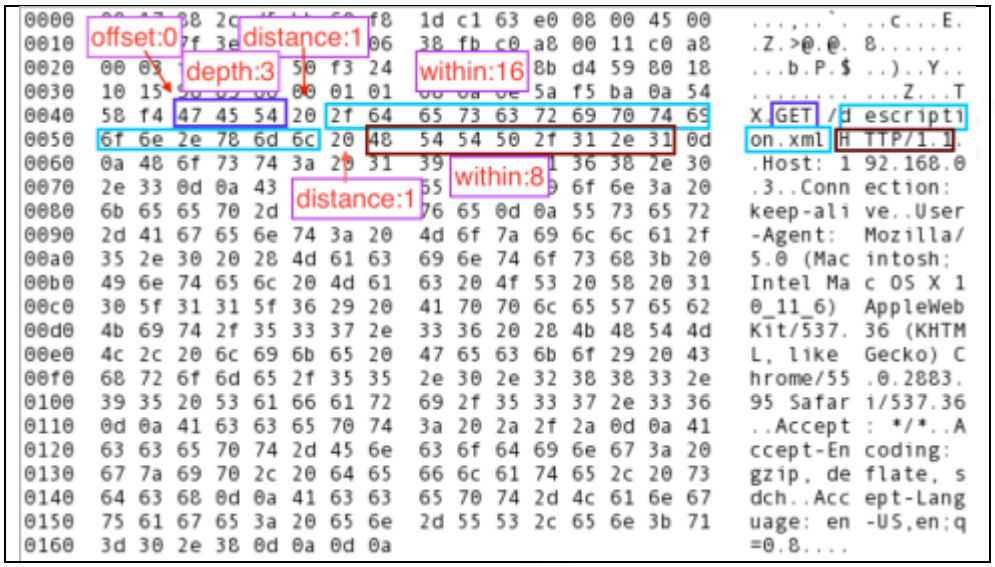


Figure 31 - 'offset', 'depth', 'distance', and 'within'

The rule as illustrated in figure 29 would then be:

```
alert tcp any any -> 192.168.0.3 80 (msg:"Test Content Rule with distance"; content:"GET";
depth:3; content:"description.xml"; distance:1; within:16; content:"HTTP/1.2"; distance:1;
within:8; sid:1000000)
```

Figure 32 – Rule Using Content Keyword Modifiers

And the resulting alert shown in figure 30 would be:

```
[**] [1:1000000:0] Test Content Rule with distance [**]
[Priority: 0]
01/30-23:38:10.232609 60:F8:1D:C1:63:E0 -> 00:17:88:2C:D5:BB type:0x800 len:0x168
192.168.0.17:63841 -> 192.168.0.3:80 TCP TTL:64 TOS:0x0 ID:55740 IpLen:20 DgmLen:346 DF
***AP*** Seq: 0xC1FC75D1  Ack: 0x24276516  Win: 0x1015  TcpLen: 32
TCP Options (3) => NOP NOP TS: 240842165 173299955
```

Figure 33 – Resulting Alert

Last Updated: April 15th, 2018

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Zurich 2018	Zurich, CH	Apr 16, 2018 - Apr 21, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MDUS	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS Seattle Spring 2018	Seattle, WAUS	Apr 23, 2018 - Apr 28, 2018	Live Event
Blue Team Summit & Training 2018	Louisville, KYUS	Apr 23, 2018 - Apr 30, 2018	Live Event
SANS Riyadh April 2018	Riyadh, SA	Apr 28, 2018 - May 03, 2018	Live Event
SANS Doha 2018	Doha, QA	Apr 28, 2018 - May 03, 2018	Live Event
SANS SEC460: Enterprise Threat Beta Two	Crystal City, VAUS	Apr 30, 2018 - May 05, 2018	Live Event
Automotive Cybersecurity Summit & Training 2018	Chicago, ILUS	May 01, 2018 - May 08, 2018	Live Event
SANS SEC504 in Thai 2018	Bangkok, TH	May 07, 2018 - May 12, 2018	Live Event
SANS Security West 2018	San Diego, CAUS	May 11, 2018 - May 18, 2018	Live Event
SANS Melbourne 2018	Melbourne, AU	May 14, 2018 - May 26, 2018	Live Event
SANS Northern VA Reston Spring 2018	Reston, VAUS	May 20, 2018 - May 25, 2018	Live Event
SANS Amsterdam May 2018	Amsterdam, NL	May 28, 2018 - Jun 02, 2018	Live Event
SANS Atlanta 2018	Atlanta, GAUS	May 29, 2018 - Jun 03, 2018	Live Event
SANS Rocky Mountain 2018	Denver, COUS	Jun 04, 2018 - Jun 09, 2018	Live Event
SANS London June 2018	London, GB	Jun 04, 2018 - Jun 12, 2018	Live Event
SEC487: Open-Source Intel Beta Two	Denver, COUS	Jun 04, 2018 - Jun 09, 2018	Live Event
DFIR Summit & Training 2018	Austin, TXUS	Jun 07, 2018 - Jun 14, 2018	Live Event
SANS Milan June 2018	Milan, IT	Jun 11, 2018 - Jun 16, 2018	Live Event
SANS Cyber Defence Japan 2018	Tokyo, JP	Jun 18, 2018 - Jun 30, 2018	Live Event
SANS Crystal City 2018	Arlington, VAUS	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS Philippines 2018	Manila, PH	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS Oslo June 2018	Oslo, NO	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS ICS Europe Summit and Training 2018	Munich, DE	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS Cyber Defence Canberra 2018	Canberra, AU	Jun 25, 2018 - Jul 07, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MNUS	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Paris June 2018	Paris, FR	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Vancouver 2018	Vancouver, BCCA	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, GB	Jul 02, 2018 - Jul 07, 2018	Live Event
SANS Cyber Defence Singapore 2018	Singapore, SG	Jul 09, 2018 - Jul 14, 2018	Live Event
SANS Charlotte 2018	Charlotte, NCUS	Jul 09, 2018 - Jul 14, 2018	Live Event
SANSFIRE 2018	Washington, DCUS	Jul 14, 2018 - Jul 21, 2018	Live Event
SANS London April 2018	OnlineGB	Apr 16, 2018 - Apr 21, 2018	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced