Guarding the Modern Castle: Providing Visibility into the BACnet Protocol

Aaron Heller

Guarding the Modern Castle: Providing Visibility into the BACnet Protocol

GIAC (GCIA) Gold Certification

Author: Aaron Heller deltah24@gmail.com
Advisor: David Hoelzer
Accepted: October 14th, 2019

Abstract

Building automation devices are used to monitor and control HVAC, security, fire, lighting, and other similar functions in a building or across a campus. Over 60% of the global market for building automation relies on the BACnet protocol to enable communication between field devices (BSRIA, 2018). There are few open-source network intrusion detection or prevention systems (NIDS/NIPS) capable of interpreting and monitoring the BACnet protocol (Hurd & McCarty, 2017). This blind spot presents a significant security risk. The maloperation of building automation systems can cause physical damage and financial losses, and can allow an attacker to pivot from a building automation network into other networks (Balent & Gordy, 2013). A BACnet/IP protocol analyzer was created for an open-source NIDS/NIPS called Zeek to help minimize this network security blind spot. The analyzer was tested with publicly available BACnet capture files, including some with protocol anomalies. The new analyzer and test cases provide network defenders with a tool to implement a BACnet/IP capable NIDS/NIPS as well as insight into how to defend the modern-day "castles" that rely on the Building Automation and Control network protocol.

1. Introduction

Both modern and older retrofitted buildings contain devices involved in automatically controlling the climate, security, power utilization, lighting, fire detection and suppression, safety, and similar aspects of the environment. When these functions are brought together into a cohesive system — a distributed control system — it is referred to as a building automation system or building management system (KMC Controls, 2013). Reduced energy consumption, centralized building control, more efficient maintenance, and increased occupant comfort are some of the benefits provided by building automation systems found in "intelligent" or "smart" buildings and homes (Drăgoicea, Bucur, & Pătraşcu, 2013).

Active monitoring of building automation systems can help secure and prevent their misuse. It is possible to abuse trust relationships between building automation system devices to pivot into other corporate networks or enslave devices to participate in distributed denial of service (DDoS) attacks. There may also be "real world" consequences like damaging equipment, impacting building access and physical security via card readers, or creating hostile building environments (Balent & Gordy, 2013; Bereza, 2019). These potential consequences make it essential to monitor building automation and control networks.

Unfortunately, many cybersecurity tools cannot interpret the protocols used to communicate within industrial control systems. This lack of insight prevents intrusion detection or prevention at the field device level (Hurd & McCarty, 2017). The Building Services Research and Information Association (BSRIA) published a report in 2018 that identified BACnet as having over 60% of the global market share for building automation and control protocols (BSRIA, 2018). BACnet is an open standard that allows building automation devices to share information between themselves, regardless of manufacturer or the types of services provided. The market share and manufacturer agnostic nature of BACnet makes it an attractive starting point for defending the modern-day, automated "castle" against cyber intrusion. A variant of BACnet, encapsulated by the User Datagram Protocol (UDP) and referred to as BACnet/IP, and the open-source Zeek

network security monitor (The Zeek Project, 2019) were selected to implement a layer of visibility and defense for building automation systems.

2. Zeek

Several of the most popular open-source network intrusion detection or prevention systems (NIDS/NIPS), including Snort and Zeek, cannot parse and interpret the BACnet/IP protocol (The Snort Project, 2019; The Zeek Project, 2019). Two factors weighed heavily on selecting an IDS/IPS for which to develop a BACnet/IP analyzer. First, the varied nature of building automation devices coupled with many not using "standard" operating systems like Windows, Linux, macOS, etc. or architectures like x86, ARM, or x64 makes signature-based network intrusion detection difficult. Second, there is some expectation of stable data exchange patterns in a building automation system. These two aspects lend themselves to behavior or event-driven NIDSs/NIPSs like Zeek versus a signature matching NIDS/NIPS like Snort.

Released in 1994, Zeek, formerly known as Bro, is an actively developed and maintained, open-source, event-driven network analysis framework (Paxson, 2018). Zeek protocol analyzers detect which protocols are present in packets, and then generate events based on the types of packets and information seen. These events can be used to provide access to protocol information fields so that scripts can be written to detect anomalous behavior. Detection scripts leverage the protocol analyzer events so that only relevant scripts run.

The Zeek BACnet/IP analyzer has been implemented as a plugin, allowing it to be used and updated without having to recompile Zeek. A high-level language called binpac was used to generate C++ code that performs the protocol parsing (Pang, Sommer, Paxson, & Peterson, 2006). The binpac-generated C++ code raises 15 different events and interfaces with the Zeek scripting language. BACnet/IP Zeek scripts place all parsed data in a bacnet.log file which can be queried via other tools and scripts to detect unexpected behaviors. Details of the BACnet/IP protocol and Zeek BACnet/IP analyzer functions are presented together in the following section.

3. BACnet

BACnet was created to provide a vendor-independent communication protocol, allowing integration of equipment controlling climate, lighting, access, fire detection, and similar functions (ANSI/ASHRAE, 2016). The first BACnet standard was published in 1995 by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Since then, it has evolved to adapt to new building technology demands and the changing landscape of technology. One of these evolutionary steps resulted in BACnet/IP, which leverages UDP over Internet Protocol (IP) to carry BACnet messages across a network. All BACnet and BACnet/IP descriptions in later sections of this paper refer to the ANSI/ASHRAE 135-2016 standard (ANSI/ASHRAE, 2016).

3.1. BACnet/IP Overview

BACnet/IP utilizes UDP/IP, (Version 4 or 6 of IP) but also implements pseudo-independent data link and network layers as well as an application layer. All layers of BACnet/IP are big-endian formatted. Figure 1 shows how BACnet/IP and UDP/IP align with the Open System Interconnect model for communication protocols.

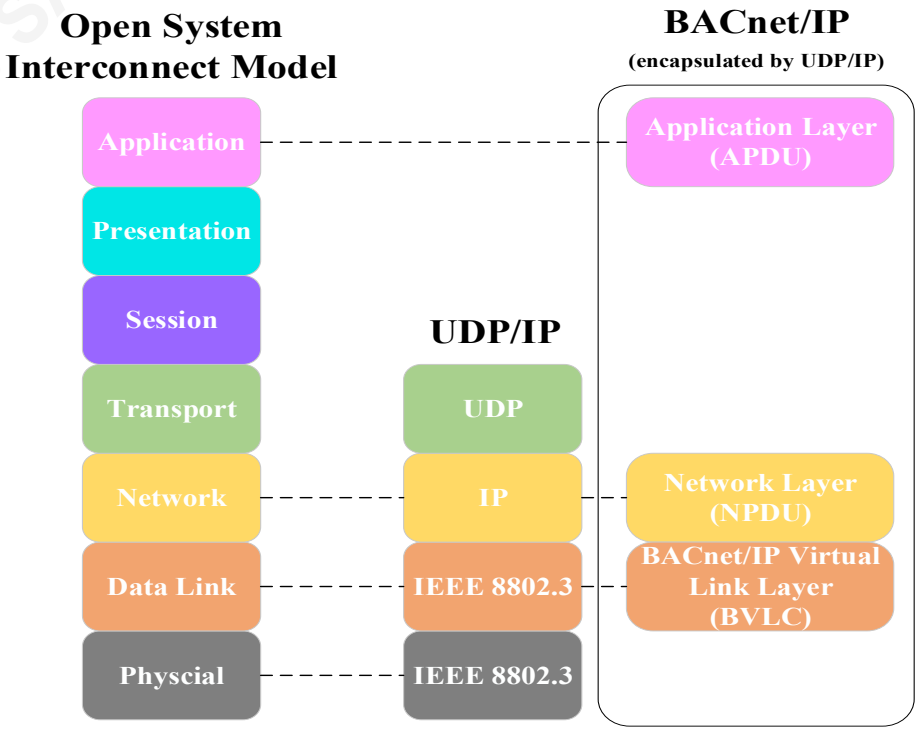

Figure 1: BACnet/IP and OSI Comparison

Aaron Heller, deltah24@gmail.com

As seen in Figure 1, there is some duplication between BACnet/IP and UDP for the network and data link layers. The BACnet protocol can exist on networks using technologies besides Ethernet / IEEE 8802.3. For example, ARCNET, LonTalk, Zigbee, and Master-Slave/Token-Passing networks, as well as point-to-point communication links, are supported. The BACnet data link and network layers provide mechanisms for devices on different networks or devices using different network technologies to connect. BACnet routers or BACnet/IP Broadcast Management Devices (BBMDs) facilitate the connections. Either type of device can also be used to distribute broadcast messages across IP subnets.

An extension to the BACnet network layer can be used to provide a measure of security. The security services offered by this extension support user and device authentication, packet integrity validation, and encryption. However, the integrity validation and encryption of packet data are only partial in some circumstances. The extension consists of a security header and digital signature and offers the option to encrypt certain types of data. Authentication and integrity verification rely on HMAC and either SHA-256 or MD5-based signatures. Encryption uses the AES-128-CBC algorithm. Encryption and signing keys are handled in pairs, distributed by a central key server, and assigned to one of several specific purposes (installation keys, general network access keys, etc.).

4. BACnet/IP and the Zeek Analyzer

The Zeek BACnet/IP analyzer can determine when to analyze a packet by either using defined UDP port numbers or dynamic protocol detection. The default behavior uses dynamic protocol detection since the BACnet/IP protocol is allowed over a range of UDP ports. The dynamic detection signature, contained in the dpd.sig file, determines whether the first byte of the UDP payload is equal to 0x81. The BACnet/IP link layer protocol data unit (PDU) will always begin with this value.

Defining UDP ports for the BACnet/IP analyzer to trigger may be desirable for some applications since less processing resources are required as compared to dynamic

protocol detection. The analyzer trigger can be changed by making minor edits to the main.bro and __load__.bro scripts. See comments in the main.bro script for guidance.

The BACnet/IP analyzer examines all three layers of the BACnet/IP protocol (data link layer, network layer, and application layer). However, it is limited to IPv4, and not all information from each layer is extracted. Since Zeek is a behavioral IDS, some information such as field lengths or application-specific data items is not of much interest. Each protocol and analyzer descriptive section notes what information is parsed or summarized in the bacnet.log file.

4.1. BACnet/IP Virtual Link Layer Implementation

The Zeek BACnet/IP analyzer begins packet analysis at the BACnet virtual link layer (BVLL). BACnet/IP contains a link-layer to tie together the BACnet/IP network layer and the lower protocol layer services provided by 8802.3 Ethernet and UDP. Fundamentally, BVLL messages communicate link-layer information to BACnet network devices, routers, and BACnet BBMDs.

Each BVLL message identifies its type (always 0x81 for BACnet/IP), the BACnet Virtual Link Control (BVLC) function number, and the length of the BVLL message. Thirteen possible BVLC functions determine what other parameters may exist in the BVLL message. Before going into these functions and what data the BACnet/IP analyzer provides, one needs an understanding of BACnet/IP broadcasting, foreign devices, routing, and addressing.

4.1.1. BACnet Routers and Addresses

BACnet routers can provide services similar to IP network address translation (NAT) (Srisuresh & Egevang, 2001). They allow BACnet devices on networks with fundamentally different technologies and network addresses to communicate. Figure 2 shows one example:

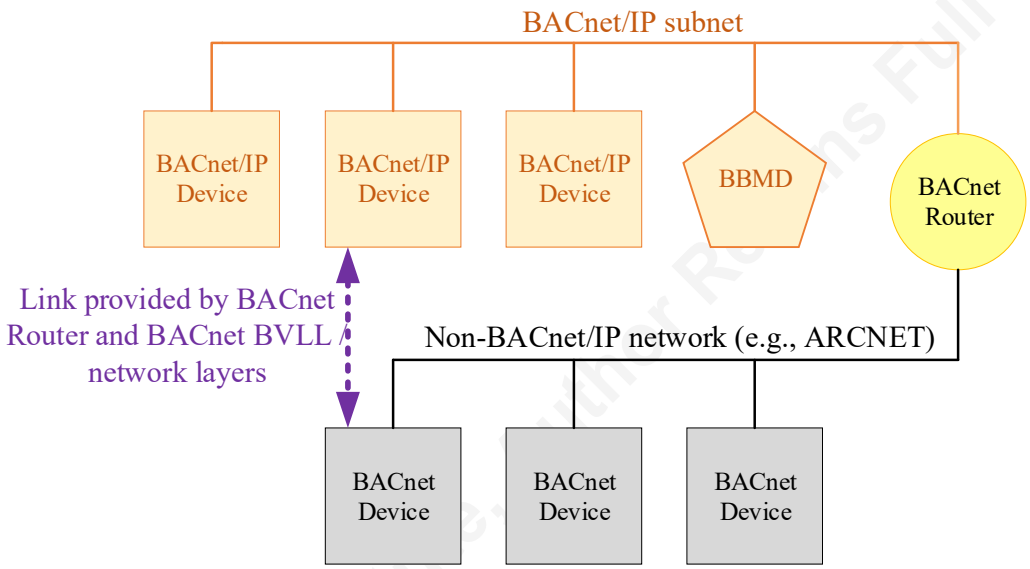

Figure 2: BACnet communication across different network types

All BACnet devices have a unique address that is partially determined by the type of network they reside on. The BACnet address functions similarly to a MAC address and provides a way for devices on different types of networks, with various addressing schemes, to communicate. These addresses are exposed by the Zeek BACnet/IP protocol analyzer through a combination of IP addresses/UDP port numbers or SRC (source) / DST (destination) networks and addresses for non-IP-based networks.

BACnet/IP addresses are composed of the device's IP address (4 bytes) and the UDP port number (2 bytes), as seen in Figure 3. The default UDP port for BACnet/IP is 47808, which is somewhat easy to remember since the hex equivalent is 0xBAC0. However, BACnet devices are required to support configurable UDP port ranges from 47808 – 47823 and 49152 – 65535.

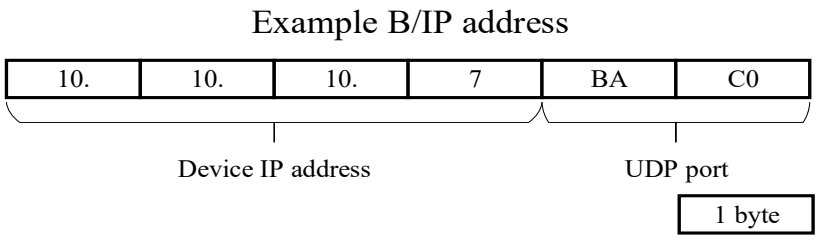

Figure 3: BACnet/IP Device Address

4.1.2. Broadcasting

BACnet/IP has two different mechanisms to broadcast packets to pieces of equipment called “foreign devices" and individual subnets. BACnet/IP foreign devices reside outside of a given subnet, but BBMDs treat them as if they are part of the subnet. Broadcast Distribution Tables (BDTs) and Foreign Device Tables (FDTs), which are maintained and acted upon by BBMDs, control the distribution of broadcast packets. The BACnet/IP analyzer logs events configuring or reading BDTs/FDTs. Any reads or manipulation of BDTs and FDTs have security implications since they can indicate reconnaissance efforts or directly affect network behavior.

In the first broadcast distribution method, BBMDs on different subnets communicate with each other using BACnet broadcast addresses. A BACnet/IP broadcast address (Figure 4) contains two parts. The first part contains a four octet address that is the result of masking the receiving BBMD’s IP address with the BBMD’s subnet and setting all host bits of the resulting host IP address bits to 1's. The second part of the BACnet/IP broadcast address consists of the UDP port of the devices on the destination BACnet/IP network. Figure 4 shows an example of how to derive a BACnet/IP broadcast address.

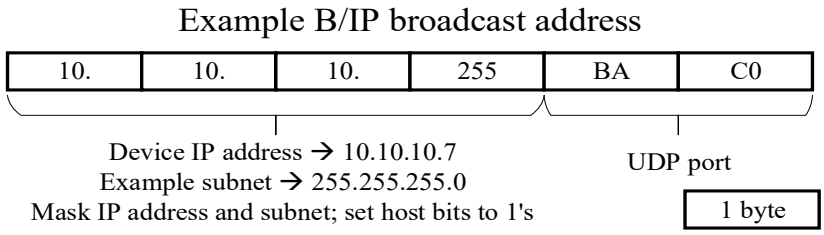

Figure 4: BACnet/IP Broadcast Address

Figure 5 shows an example broadcast eventFigure 5 via the blue arrows. The broadcast is initiated on BACnet/IP subnet 1 by Device 2 sending a BVLL message with an “Original-Broadcast-NPDU” control function to BBMD 1. Using entries in a BDT and FDT, BBMD 1 distributes the message to the appropriate BBMDs and foreign devices. BBMD 2 on subnet 2 receives the Original-Broadcast-NPDU and forwards it to the intended devices on the subnet, per its BDT, and any foreign devices in its FDT as a “Forwarded-NPDU” message type.

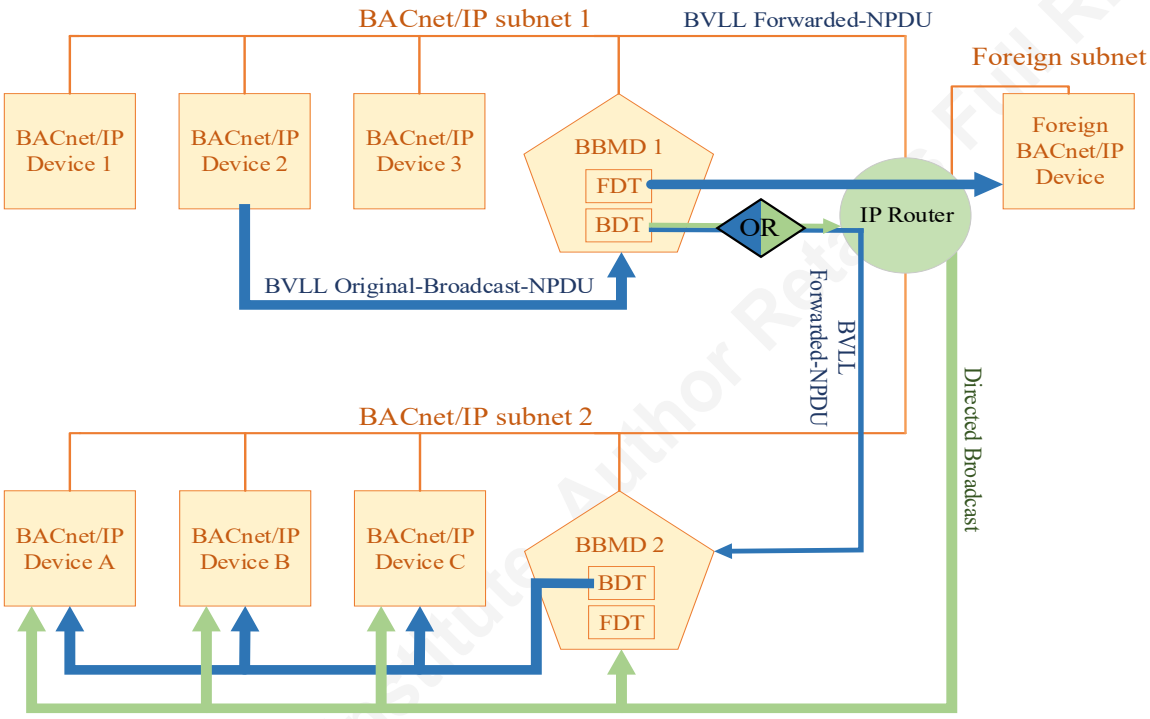

Figure 5: BACnet/IP Broadcasts

The second broadcast distribution method starts in the same manner with BBMD 1 receiving the Original-Broadcast-NPDU message and forwarding it per its BDT and FDT as a Forwarded-NPDU message. This time, instead of forwarding the message to BBMD 2, the message forwards to a directed broadcast capable router (Figure 5, green arrows). The router direct broadcasts to devices on subnet 2, including BBMD 2. A broadcast distribution mask in BBMD 2's BDT allows it to determine how to treat a received broadcast message. The message will either be broadcasted to subnet 2 (as it would in the first case above) or only distributed to foreign devices in the FDT (as it would in this case).

A registered foreign device (one that exists in a BBMD's FDT) may also initiate a broadcast. The initial message from a broadcasting foreign device will be a Distribute-Broadcast-To-Network message vs. an Original-Broadcast-NPDU. However, either broadcast method may carry out the broadcast request. Regardless of the broadcast or initiation method, message events, sources, and destinations, the Zeek BACnet/IP analyzer will log the messages. The DADR bacnet.log field will also indicate if the

message is part of a broadcast event and if the message originated from outside the subnet.

4.1.3. BACnet/IP Virtual Link Control Functions and Logging

Each BVLC function generates a semi-unique Zeek BACnet/IP analyzer event. For example, a "Results" BVLC generates a unique Zeek event, but "Read-Broadcast-Distribution-Table" and "Read-Foreign-Device-Table" BLVCs share the same event since their data structures are the same.

The BACnet/IP analyzer provides summary statistics for the number of packets that contain only BVLL data. These packets encompass 9 of the 13 BVLC functions and deal with tasks such as configuring foreign device and broadcast distribution tables, registering foreign devices, and transmitting secure BVLL messages. The bacnet.log file contains a count of each BVLL packet type on a per-connection basis.

BVLL-related information in the bacnet.log file can be used to baseline routine BVLL communications for each connection. In turn, deviations from the baseline can be used to troubleshoot network or equipment issues, including those which may have cybersecurity implications.

4.2. BACnet/IP Network Layer

Network Protocol Data Units (NPDUs) contain information about the BACnet/IP network layer. Each NPDU either communicates information about BACnet networks via a Message Type or encapsulates an Application Protocol Data Unit (APDU). Forwarded-NPDU, Distribute-Broadcast-To-Network, Original-Unicast-NPDU, or Secure-BVLL are the only four BVLC functions that can encapsulate an NPDU.

Table 1: NPDU Fields

Field	Byte Size
Version	1
Control	1
DNET	2
DLEN	1
DADR	DLEN
SNET	2
SLEN	1
SADR	SLEN
Hop Count	1
Msg Type	1
Vendor ID	2
APDU	Variable

4.2.1. Version and Control Fields

Regardless of the NPDU purpose, all NPDUs start with a single-byte field describing the BACnet version. As of ASHRAE 135-2016, the BACnet/IP version is always equal to '0x01' and is therefore not logged. The second

field, Control, is also one byte. The Control field describes the other fields present in the NPDU, whether to expect a specific type of reply message, and the network priority level. See Table 1 and Table 2. Table 1Table 2

Table 2: NPDU Control Field Interpretation

Bit	Interpretation
7	1 = Message Type field present (no APDU) 0 = APDU present (no Message Type field)
6	Reserved. Always 0.
5	1 = DNET, DLEN, DADR, and Hop Count are present. If DLEN = 0 then DADR is absent (broadcast) 0 = DNET, DLEN, DADR, and Hop Count are absent
4	Reserved. Always 0.
3	1 = SNET, SLEN, and SADR are present 0 = SNET, SLEN, and SADR are absent
2	1 = an application layer PDU of type 0x0 or 0x3, or a network layer message is expecting a reply 0 = anything other than an application layer PDU of type 0x0 or 0x3, or a network layer message expecting a reply is present
1	0 = Normal message 1 = Urgent message 2 = Critical Equipment message 3 = Life Safety message
0	

Two of the eight bits in the Control field aren't used (reserved) and always equal to zero. Bits 3, 5, 7, and the value of the DLEN field are left to determine which NPDU fields are present in a given packet. Control bits 0 and 1 indicate the network priority level, while bit 2 describes whether to expect certain reply messages. Each connection recognized by Zeek and the BACnet/IP analyzer will capture the number of packets for each priority level in the bacnet.log file.

4.2.2. NPDU Source, Destination, and Hop Count Fields

Source network (SNET), source address (SADR), destination network (DNET), and destination address (DADR) NPDU fields identify BACnet devices on different subnets or network types. The length of these four fields can vary based on the underlying network technology. SLEN and DLEN fields are used to communicate the kind of network technology and the size (measured in bytes),of the SADR and DADR. Network management messages also leverage DNETs and DADRs. For example, recipients of a "Who-Is-Router" network management message use the DNET field to determine if a response should be sent (only a router connected to the specified DNET should respond).

BACnet/IP source and destination addresses are six bytes long and have corresponding SLENs or DLENs equal to six. Of note is that an SLEN = 0 is considered

invalid by the ASHRAE 135-2016 standard. However, publicly available capture files show several occurrences of incorrect implementations.

The Hop Count field only appears with the DNET field and has a default value of 255. Decremented by one for every router a packet passes through, its function is similar to the time-to-live (TTL) field in IPv4 in preventing a packet from existing forever on a network. BACnet routers discard a packet when the Hop Count reaches zero. Unlike IPv4, there is no error message sent from the router to the originator when a packet is discarded.

The Zeek BACnet/IP analyzer logs summary statistics for how many NPDU packets occur in each connection as well as the source and destination networks and addresses. For example, Zeek may show a BACnet/IP router communicating to BACnet/IP device. A BACnet device on a LonTolk network may communicate through the router to the BACnet/IP device. The bacnet.log file will show SNET and SADR fields with LonTalk-specific information as well as identify that the network technology is LonTalk. Logging occurs in a similar manner for other BACnet compatible networking technologies.

4.2.3. Message Type, Vendor ID, and Security Wrappers

The NPDU Message Type field indicates whether information about BACnet routers, networks, security information, or vendor proprietary messages is being requested or provided. If a Message Type field exists, then there will not be an APDU in the same packet *unles*s it is encapsulated by a security wrapper.

NPDUs with message types 0x0A through 0x11 contain a security wrapper encapsulating an APDU or another NPDU

Table 3: NPDU and Security Wrapper

	Field	Type
	Version	NPDU Network Protocol Control Information
	Control	
	DNET/DLEN/DADR	
	SNET/SLEN/SADR	
	Hop Count	
	Msg Type	
Signature coverage	Control	Security Wrapper
	Key Rev / Key ID	
	Src. Instance	
	Message ID	
	Timestamp	
	Dst. Instance	
	DNET/DLEN/DADR	
	SNET/SLEN/SADR	
	Service Data	
	Padding	
	Signature	

= potentially encrypted

message type in a Service Data field, as seen in Table 3. The security wrapper provides a signature to protect the integrity and authenticity of the Service Data field. An APDU or second NPDU Message Type, several security wrapper fields, as well as any Message Type field in the NPDU may reside in the Service Data field. Several fields in the security wrapper can also be encrypted, as previously discussed in Section 3.1.

The BACnet/IP analyzer records the quantity of each NPDU Message Type seen on each connection in the bacnet.log file, which can be used to characterize expected network operation. For example, security-related message types can indicate requesting, setting, and updating various security keys. These types of messages should occur in low volumes on a correctly configured BACnet network. Spikes in the quantity of security-related message types can indicate attempted compromises. Similar concerns exist if the source of any key update messages (message types 0x0E and 0x0F) is different than the address of the designated key server.

4.2.4. NPDU size considerations

There are different maximum transmission unit (MTU) sizes for the various link-layer technologies supported by the BACnet standard. BACnet/IP has a maximum NPDU length of 1497 bytes, which is the largest size for any data link technology used by BACnet. NPDUs sent from BACnet/IP networks to BACnet devices on ARCNET, LonTalk, ZigBee, or point-to-point network types have smaller NPDU size limitations. Packet sizes can indicate what kind of network technology is used by the BACnet endpoint, but it is not always conclusive, so the BACnet/IP analyzer does not log size information. For example, device A could communicate packets to BACnet/IP device B which all have lengths less than 300 bytes. A maximum packet size of 300 bytes can prove that device A is not on a LonTalk network (MTU is 228 bytes), but it does not prove anything more. Device A could still reside on any BACnet-supported network technology besides LonTalk. There also may be device buffer size limitations that could lead to false conclusions. Limited buffer sizes could result in all packets being less than 200 bytes, but the devices involved could both be on BACnet/IP networks. If average packet sizes for a connection participant is of interest, then it can be calculated from the conn.log file, which is automatically generated by Zeek.

4.3. BACnet/IP Application Layer

The uppermost layer of BACnet/IP contains the APDU. Some APDU types include connection information typically found in the transport layer of protocols like Transmission Control Protocol (TCP) (Braden, 1989). This connection information allows data lengths greater than the MTU to be segmented across multiple APDUs.

Table 4: APDU Fields

Field	Size
PDU Type	Nibble
Segmented	Bit
More Segments	Bit
Segments Accepted	Bit
Negative ACK	Bit
Server/Client	Bit
Orig. Invoke ID	Byte
Segment Number	Byte
Proposed/Actual Window Size	Byte
Service (ACK) Choice	Byte
Error Choice	Byte
Reject Reason	Byte
Abort Reason	Byte
Service ACK	Variable
Service Request	Variable
Error	Variable

The first byte of all APDUs is interpreted as two separate nibbles. The first (highest order) nibble identifies the APDU type. The second nibble may or may not contain flags related to segmentation, negative acknowledgments, or the client vs. server origin of an APDU. The BACnet/IP analyzer logs the APDU type, number of segmented requests, acknowledgment of segments received, and the total number of segments seen on each connection. The bacnet.log file also contains the number of APDU types that transmit only Error Choice, Rejection Reason, or Abort Reason information.

Other fields in the APDU, as shown in Table 4, are not logged by the BACnet/IP analyzer. The information in the unlogged fields is specific to the type of BACnet device and controlled process (temperatures, pressures, etc.). Therefore, it does not provide much insight into network and communication behavior.

5. Limitations

There are a few limitations to the BACnet/IP analyzer and what information can be logged or exposed to custom detection scripts. First, IPv6 is not supported. IPv6 addresses will cause the BACnet/IP parsing routines to report incorrect data for any fields following an IPv6 address. Second, there are no mechanisms to access or log data within the encrypted portions of secure BVLL or secure NPDU/APDU messages.

Certain features of the BACnet/IP analyzer also have not been thoroughly tested. These areas primarily deal with parsing NPDU SNET/SADR and DNET/DADR information from packets originating from LonTalk-based networks. Notes in the source code also highlight these limitations.

6. Example BACnet/IP Attacks and Detection Script

It is possible to recreate certain types of attacks seen with TCP/IP based protocols and apply them to BACnet/IP. The following section explains an example attack and a corresponding detection script to show how the Zeek BACnet/IP analyzer can effectively detect various BACnet/IP enabled attacks.

6.1. Segment Number Prediction

BACnet/IP is vulnerable to an attack similar to TCP sequence number prediction, which is an attack that targets the predictable starting value and increment of 32-bit TCP sequence numbers. If an attacker guesses a correct TCP sequence number, then they may be able to spoof one of the TCP endpoints into accepting data from the attacker (Morris, 1985).

BACnet/IP can have segmented messages in which several packets are needed to transmit all of the requested data. Both data requests and responses can have segments. A combination of network address, message Invoke ID, and segment number determine if a packet belongs to a segmented connection. The Invoke ID and segment number are each 8 bits, but the segment number always starts at zero and increases by one for each packet sent. An attacker can predict the Invoke ID (a 1 in 255 chance) and a segment number (less than a 1 in 255, since it always starts at zero) with greater frequency than similar TCP sequence number prediction attacks.

The BACnet-detect-seg-guess.bro script is written to track endpoints involved in any segmented message exchanges and place a bacnet::Segment_Guessing notice in the Zeek notices.log file if more than two endpoints are involved. The script logic triggers off of the three APDU types that can partake in segmented messages: Confirmed-Request-PDUs, ComplexACK-PDUs, and SegmentACK-PDUs. The Zeek StatsSum framework is leveraged to track the endpoints and the Invoke IDs associated with each

segmented message stream. The test case capture file was created using segmented communication packets extracted from "TestRun4 – Outside of Router.pcap" (Karg, 2019) using Wireshark (Combs, 2019). Newly crafted packets were added using Scapy (Philippe, 2019) and a Scapy-bacnet add-on (Forsberg, 2015). The Wireshark mergecap tool (Renfro, 2019) was used to assemble the packets into a cohesive capture file, seg_guess.pcap.

7. Areas for Further Research

There does not appear to be much cybersecurity research focused on attacking the BACnet protocol, unlike TCP, IP, and other more traditional communication protocols. Further investigation into BACnet-specific attacks would better inform what types of detection mechanisms are best suited to protect BACnet networks.

The next evolution of BACnet is currently under development and will be called BACnet/SC. The primary purpose of BACnet/SC is to bring further security to the BACnet protocol by incorporating the widely used Transportation Layer Security (TLS) protocol. Security research will be needed on BACnet/SC to determine if there are any implementation flaws, and what behavioral characteristics of the protocol should be monitored. BACnet/SC will not make BACnet/IP obsolete overnight. The types of devices that communicate using BACnet are often not upgraded or replaced for much more extended periods than traditional IT equipment like computers and servers.

8. Conclusion

BACnet/IP is a control system protocol that is used by many building automation systems found in "intelligent" or "smart" buildings and homes. Like other traditional IT and control system protocols, the features and implementation of BACnet/IP can be abused in a variety of ways that can have security implications. An open-source analyzer plug-in for the Zeek NIDS was developed and presented in this paper. The intent is to help the cybersecurity community better understand and defend building automation systems using the BACnet/IP protocol. The Zeek BACnet/IP analyzer plug-in and source code are available at https://github.com/delta-2-4/Zeek-BACnetIP.

Aaron Heller, deltah24@gmail.com

9. Development Environment, Tools, and Resources

Name	Function	Location	Version
Zeek	Network analysis framework	https://www.zeek.org	2.6.2
Zeek BACnet/IP plugin and test pcaps	BACnet/IP analyzer plug-in for Zeek	https://github.com/delta-2-4/Zeek-BACnetIP	Initial
binpac	Protocol parsing C++ code generator	https://github.com/zeek/binpac	0.52
Scapy	Packet crafting framework	https://scapy.net/	2.4.3
scapy-bacnet	BACnet/IP extension for Scapy	https://github.com/desolat/scapy-bacnet	March 11, 2015
Wireshark mergecap tshark	Protocol analyzer and tools	https://www.wireshark.org/	2.6.8
VirtualBox	Virtual machine hypervisor	https://www.virtualbox.org/	6.0.8
Ubuntu	Operating system	https://ubuntu.com/	18.04.3

Aaron Heller, deltah24@gmail.com

References

ANSI/ASHRAE. (2016). *ANSI/ASHRAE Standard 135-2016 BACnet A Data Communication Protocol for Building Automation and Control Networks.* Atlanta: ASHRAE.

Balent, M., & Gordy, F. (2013). Protecting Facility Networks From Cyber Attack. *Heating/Piping/Air Conditioning Engineering*, 18–26.

Beach, D. (2008, October 29). *Port Exhaustion and You (or, why the Netstat tool is your friend).* Retrieved from Microsoft Technet: https://blogs.technet.microsoft.com/askds/2008/10/29/port-exhaustion-and-you-or-why-the-netstat-tool-is-your-friend/

Bereza, M. (2019, August 9). *From Building Control to Damage Control: A Case Study in Industrial Security Featuring Delta's enteliBUS Manager*. Retrieved from McAfee, LLC Website: https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/from-building-control-to-damage-control-a-case-study-in-industrial-security-featuring-deltas-entelibus-manager/

Braden, R. (1989). *Requirements for Internet Hosts -- Communication Layers.* RFC Editor. RFC Editor.

BSRIA. (2018). *Market Penetration of Communication Protocols.* Chicago: BSRIA Inc.

Combs, G. (2019). *Wireshark.* Retrieved June 30, 2019 from https://www.wireshark.org

Drăgoicea, M., Bucur, L., & Pătraşcu, M. (2013). A Service Oriented Simulation Architecture for Intelligent Building Management. *Exploring Services Science.* Berlin, Heidelberg: Springer Berlin Heidelberg.

Forsberg, E. (2015, March 11). *A BACnet layer for Scapy*. Retrieved September 1, 2019 from GitHub: https://github.com/desolat/scapy-bacnet

Hurd, C. M., & McCarty, M. V. (2017). *A Survey of Security Tools for the Industrial Control System Environment.* United States. doi:10.2172/1376870.

Karg, S. (2019, January 4). *Index of /captures*. Retrieved from KARGS: http://kargs.net/captures/

Aaron Heller, deltah24@gmail.com

KMC Controls. (2013, May 19). *Understanding Building Automation and Control Systems*. Retrieved from KMC Controls: https://web.archive.org/web/20130519124213/http://www.kmccontrols.com/products/Understanding_Building_Automation_and_Control_Systems.aspx

Morris, R. T. (1985). *A Weakness in the 4.2BSD Unix TCP/IP Software.* Murray Hill: AT&T Bell Laboratories.

National Institute of Standards and Technology. (2015). *NIST SP800-82: Guide to Industrial Control Systems (ICS) Security, Revision 2.* Gaithersburg, MD, USA: National Institute of Standards and Technology.

Pang, R., Sommer, R., Paxson, V., & Peterson, L. (2006). binpac: A yacc for Writing Application Protocol Parsers. *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, (pp. 289-300). Rio de Janeriro.

Paxson, V. (2018, October 11). *Renaming the Bro Project*. Retrieved from Zeek Blog: https://blog.zeek.org//2018/10/renaming-bro-project_11.html

Philippe, B. (2019). *Scapy.* Retrieved September 1, 2019 from https://scapy.net/

Renfro, S. (2019, June 30). *mergecap.* Retrieved from https://www.wireshark.org/

Srisuresh, P., & Egevang, K. B. (2001). *Traditional IP Network Address Translator (Traditional NAT).* RFC Editor.

The Snort Project. (2019, February 26). *Snort (R) Users Manual.* Retrieved from The Snort Project: https://www.snort.org

The Zeek Project. (2019, June 10). *Protocol Analyzers*. Retrieved from The Zeek Network Security Monitor: https://docs.zeek.org/en/stable/script-reference/proto-analyzers.html