Securing BACnet's Pitfalls

Jaspreet Kaur, Jernej Tonejc, Steffen Wendzel, Michael Meier

Securing BACnet's Pitfalls

Jaspreet Kaur, Jernej Tonejc, Steffen Wendzel, and Michael Meier

Fraunhofer FKIE, Bonn, Germany
{jaspreet.kaur, jernej.tonejc, steffen.wendzel,
michael.meier}@fkie.fraunhofer.de

Abstract. Building Automation Systems (BAS) are crucial for monitoring and controlling buildings, ranging from small homes to critical infrastructure, such as airports or military facilities. A major concern in this context is the security of BAS communication protocols and devices. The *building automation and control networking* protocol (BACnet) is integrated into products of more than 800 vendors worldwide. However, BACnet devices are vulnerable to attacks. We present a novel solution for the two most important BACnet layers, i.e. those independent of the data link layer technology, namely the network and the application layer. We provide the first implementation and evaluation of traffic normalization for BAS traffic. Our proof of concept code is based on the open source software Snort.

Keywords: BACnet, network, security, attack, intrusion detection, traffic normalization, building automation, Snort

1 Introduction

BACnet (*Building Automation and Control Networking Protocol*) is an open data communication protocol developed by ASHRAE (*American Society of Heating, Refrigerating and Air Conditioning Engineers*), standardized by ISO 16484-5 [1] and used for building automation systems (BAS). In general, BAS are integrated in and capable of controlling and monitoring buildings. Moreover, BAS form networks which can be interconnected with other buildings and the Internet (e.g., for remote monitoring purposes). In order to support interoperability, BACnet can use different lower level network protocols to perform its functions [2]. In addition to BACnet, the *European Installation Bus* (EIB)/*Konnex* (KNX), and the *Local Operating Network* (LON) are the most common BAS protocols used in practice. The main goals of BAS are to improve the energy efficiency of buildings, to increase the comfort and safety of the people living or working in a building, and to decrease a building's operational costs.

Because of the immense growth of BAS, especially BACnet, the need for secure interconnection between BAS devices is increasing. Currently, there are neither intrusion detection nor intrusion prevention systems which are capable of detecting or preventing various network and application layer based attacks on BACnet devices. Although security features for BAS protocols are specified

in their standards and have improved over time, they are, as highlighted in discussions with our industry partners, usually neither integrated in devices nor used in practice. Due to the Internet connectivity of BACnet systems and the fact that BACnet devices can be found using the SHODAN search engine (cf. `www.shodanhq.com`), remote attacks on BACnet devices can also be performed. Such attacks can, for instance, compromise smoke detectors or other critical BAS equipment. According to [3], there were more than 100,000 Internet-connected smart devices (including media players, smart televisions and at least one refrigerator) interconnected to a network of computers, which were able to send 750,000 spam emails between December 23, 2013 and January 6, 2014. In addition, the so-called *smart building botnets* could extend the capabilities of today's botnets by taking advantage of a building's physical capabilities (sensors and actuators) [4].

In this paper, we show how to prevent the exploitation of vulnerabilities at the BACnet network and application layers. In particular, devices which interact with humans have to be both secure and safe; otherwise they can threaten and compromise human life. We improve network and application reliability and security with *traffic normalizers* (also known as *protocol scrubbers* [5] from TCP/IP networks). In TCP/IP networks, traffic normalization is capable of preventing various attacks on TCP/IP stack implementations [7]. So far, there is *no* BAS protocol for which traffic normalization has been applied. The building automation devices usually remain in buildings for decades, possess limited processing power and insecure firmware, and often cannot cope with malformed or malicious packets, hence there is an increased need for such normalizers.

We analyze the BACnet network and the application layer with its potential vulnerabilities and its potential non-compliant behavior, with the goal of ensuring that BACnet devices receive correctly formed messages. In addition, we study the well-known attacks from TCP/IP and adapt them to the corresponding BACnet network and application layer, in order to design effective countermeasures. Based on the discovered vulnerabilities, we present the first implementation of traffic normalization for building automation systems in the form of a Snort [6] extension. Additionally, our normalization rules provide a means to counter fuzzing attacks and protect BACnet devices which are are not usually updated because patching is a challenging task for BAS. We design our system to normalize traffic between BAS subnets (e.g., between different floors of a building or between separate buildings). Our normalizer implementation ensures that the transferred packets reach the receiver well-formed according to the protocol standard, without protocol vulnerabilities. The Snort extension is implemented in a way that gives us an opportunity to either limit or prevent the initiated intrusions mentioned in this paper.

The rest of the paper is structured as follows. We summarize the related work in the area of BACnet security in Sect. 2. In order to make our method of protecting vulnerabilities in the BACnet network and application layer understandable, we provide a short overview on the relevant parts of the BACnet protocol in Sect. 3. Section 4 lists several vulnerabilities that were found by look-

ing at the specifications of the standard, together with the adaptations of the TCP/IP attacks. Our normalization rules are presented in Sect. 5. The overview of the testing environment is presented in Sect. 6. In Sect. 7, we evaluate and summarize our results and discuss future work.

2 Related Work

Earlier installations of BAS were designed to work as isolated standalone systems with minimal security features. As the BAS functionality requirements increased, interconnectivity, interoperability, and especially Internet access for BAS became significant features. However, the interconnectivity of BAS enables remote attacks. Attacks on BAS can focus on gaining physical access to a building [8] (e.g., by exploiting window or door actuators), on gaining access to an organizational intranet [9], on terrorist attacks [8] (e.g., turning off fire alarms before a fire is placed), on monitoring inhabitants [10], or on disabling a building's functionality via denial-of-service (DoS) attacks [11]. Čeleda et al. [12] and Szlósarczyk et al. [13] showed that different types of DoS attacks exist for BAS. As pointed out by Bowers [14], *BACnet devices are not robust enough to deal with abnormal traffic*, since protocol implementations are vulnerable to malformed packets and various forms of attacks. The *BACnet Attack Framework* (BAF) [14] introduces attack techniques, namely attacks on the BACnet routing, network mapping, DoS and spoofing. The existing work provides a good estimate of the current attack surface for BAS networks.

In terms of defense against (some of) the attacks mentioned above, the *BACnet Firewall Router* (BFR) [15] is the first approach integrating simple firewall functionality in BACnet. BFR is an open source project that implements filters for BACnet messages and is capable of performing NAT, software-side network switching, and routing. However, the BFR does not possess any normalization capabilities. In comparison to the previous work, we present the first traffic normalization for BAS capable of

- countering typical attacks known from TCP/IP networks,
- ensuring compliance, and
- increasing robustness against vulnerability tests and fuzzing attacks.

3 Structure of BACnet NPDU and APDU

BACnet's purpose is to handle a number of application areas such as lighting, fire alarms, and heating, ventilation, and air-conditioning (HVAC) in a cost effective, interoperable, and reliable manner [16]. BACnet defines four layers (physical, data link, network, and application layer), similar to the particular functions of the OSI layers (shown in Fig. 1) and is designed to adapt to different data link and physical layer technologies to achieve data link layer-independent communication. BACnet network layer messages can be encapsulated in UDP (referred to as BACnet/IP), the BACnet-specific protocol MS/TP (RS485) or

LonTalk ZigBee. The BACnet model differs from the OSI layer model in that the BACnet application layer is additionally responsible for performing and handling the message segmentation and reassembly, a feature usually accomplished by the transport layer.

OSI Layer	**BACnet Stack Protocol**						
Application	BACnet Application Layer						
Network	BACnet Network Layer						
Data Link	BACnet/IP over ISO 8802-2 LLC		MS/TP	LONTalk	PTP	BVLL	BZLL
Physical	Ethernet	ARCNET	RS485		RS232	UDP/IP	ZigBee

Fig. 1. BACnet OSI Layers, from [1].

Our work focuses on the network and application layers. Therefore, we need to introduce the addressing scheme and the structure of the BACnet *Network Protocol Data Unit* (NPDU) and the *Application Protocol Data Unit* (APDU).

Each BACnet device has a medium access control (MAC) address which is combined with the BACnet (sub)net number to form the network level address. An essential feature in BACnet is *broadcasting*. Due to the nature of BACnet topology, three types of broadcasts are supported: local, global, and remote.

	Octets	**Description**
NPCI	1	Version
	1	NPCI Control Octet (CO)
	2	Destination Network (DNET)
	1	Destination Address Length (DLEN)
	Variable	Destination Address (DADR)
	2	Source Network (SNET)
	1	Source Address Length (SLEN)
	Variable	Source Address (SADR)
	1	Hop Count
NSDU	Variable	Network Layer Message or Application Layer Protocol Data Unit (APDU)

Bit	**Description**
7	Indication
6	Reserved
5	Dest. Specifier
4	Reserved
3	Source Specifier
2	Expecting Reply
1	Priority
0	

Fig. 2. BACnet NPDU format (*left*) and NPCI control octet (*right*). In NPCI control octet, Bit 7 indicates whether the NSDU contains a network layer message (bit is set) or an APDU (bit is unset). Based on [1].

3.1 Network Layer

Even if the BACnet network layer is embedded into various data link layer protocols, the NPDU structure remains unchanged. We will focus on BACnet/IP, i.e. BACnet encapsulated in UDP sent over IPv4 [1], for which we define our normalization rules. The structure of the BACnet NPDU is shown in Fig. 2.

3.2 Application Layer

An application program which uses the BACnet protocol interacts with a BACnet peer device. The purpose of the interaction itself is mainly to invoke device-specific behavior, e.g. switching on/off a lighting device or ringing an alarm bell of an alarm device. To realize application-specific behavior, so-called objects are specified for functional behavior and services are specified for the interaction with the devices. The application layer then defines all required objects and services for a device's interaction with an application program. Notice that the application itself is independent of the application layer and is outside the scope of the BACnet ISO standard. In particular, the standard does not specify the Application Programming Interface (API).

For normalization, the relevant part of the APDU is the first region, called the Application Protocol Control Information (APCI), which is always present and whose length varies from 2 to 6 bytes depending on the PDU type. An example of a Confirmed Request PDU is shown in Fig. 3.

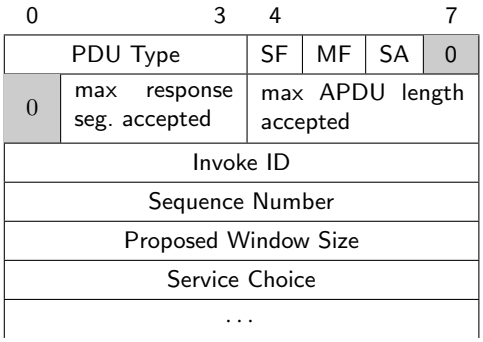

Fig. 3. BACnet APCI for Confirmed Request PDU (PDU Type = 0). From [1, p. 538].

4 Exploiting the BACnet Network and Application Layer

We base our normalization rules and the need for traffic normalization by looking at the potential security deficits in BACnet. We group our attacks on BACnet/IP into two main categories: attacks adapted from TCP/IP, and attacks specific to

BACnet. Each category represents the possible vulnerabilities allowing the exploitation of BACnet devices by taking advantage of primitive vulnerabilities in the network or application layer. We give a brief overview of the general attacker model. In this model, the attacker is outside the BACnet network with a goal to exploit a BACnet device. He is sending packets remotely to a BACnet device (e.g. fire alarm, HVAC or a simple door) through a BACnet Broadcast Management Device (BBMD), which forwards all the packets to the corresponding device. This scenario can be considered as the standard approach to attack BACnet environments remotely as BBMDs are always present to handle broadcasts between BACnet devices. The model is graphically depicted in Fig. 4 *left*.

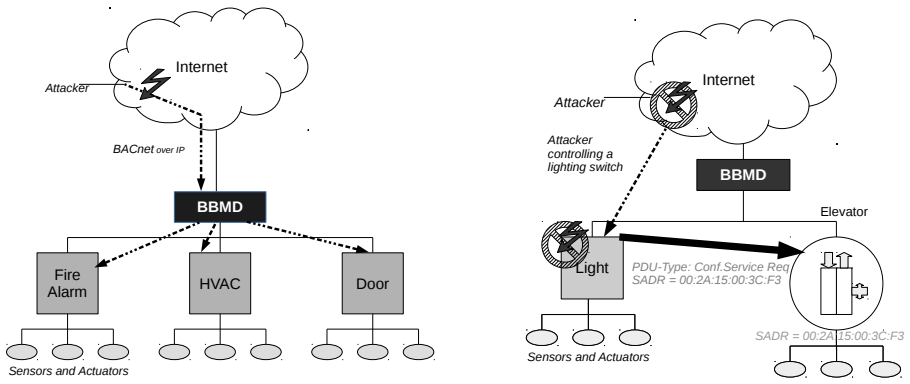

Fig. 4. General Attacker model (*left*) and an attack through a controlled BACnet device (*right*).

4.1 Attacks adapted from TCP/IP

Covert Channels. As shown in [10], BACnet allows for creating network covert channels, i.e. the policy-breaking communication channels capable of transferring data in a stealthy manner which can enable hidden command and control communication for botnets. Disguised as BACnet traffic, malware could use BACnet covert channels to hide illegitimate or confidential data within unobtrusive Internet-based traffic.

Many hiding techniques for network covert channels are based on the idea of embedding hidden information in the unused NPDU and APDU fields. In the case of BACnet, all reserved bits can serve the purpose of embedding hidden information. Moreover, the use of particular BACnet message types and timing variations between BACnet messages can signal hidden information [10].

Abnormal behavior leading to botnets in BAS. Referring to the example of a refrigerator sending spam mails (cf. Sect. 1), we extended our research to

examine the compromised BACnet devices, including BBMDs. Our concern is to provide security measures for the BACnet protocol against the abnormal behavior. Čeleda et al. [12] introduced two examples of attacks that can serve as a backdoor for an attacker in a BACnet network. The *WriteProperty* attack can cause a BBMD or a BACnet device to switch on/off, resulting in an opportunity for an attacker gaining access to the restricted BACnet network. In general, the *WriteProperty* attack and disabling of network connection are possible by changing values in BACnet's object properties, respectively misusing BACnet services, i.e. in application layer [12].

In addition, after successfully attacking a BACnet device, an attacker has an opportunity to use the device's communication channel within the subnet. One of the possible examples of such a scenario is shown in Fig. 4 *right*. We assume the attacker has determined the details of both BACnet devices in the subnet with the help of probing. If he takes over the lighting device, as shown in Fig. 4 *right*, then he gains the ability to transfer any kind of message he wants.

For instance, he can send a packet with an APDU containing a *BACnet-Confirmed-Request* to the elevator device in which the service-related data indicates the elevator should stop immediately. Because the lighting device is in the subnet, it is trusted whenever authenticity checks are performed, hence the attacker is able to bypass this security mechanism. The elevator accepts the request, sends a *BACnet-Simple-ACK*, and stops immediately.

4.2 Attacks specific to BACnet/IP

Standard Non-conformance. We start by simply listing the vulnerabilities which are tolerated by the standard. We distinguish the following terms, related to the packets: *compliant traffic* fulfill the requirements given by the standard and *non-compliant traffic* contain packets that do not conform to the standard, e.g. malformed packets. *Malicious packets* are defined as packets designed to exploit a threat. Malicious packets can either be compliant or not. We present the following two examples to illustrate:

1. *Network Mapping*: An attacker is able to map the network with standard-compliant messages like *Who-is* requests. Additionally, he is able to send NPDUs containing application-specific APDU messages to probe the network. The messages with reachable destination addresses are always forwarded by the BBMD to the corresponding BACnet devices [17]. If a device understands the service-related data contained in the payload, it gives a valid response, otherwise the device returns an error. On getting a response, the attacker knows which kind of devices can be addressed, e.g. fire alarm, HVAC or a door. As all BACnet devices send responses, he is additionally able to infer where (i.e. in which subnet) a device is located since the requests and responses contain the destination and source addresses.
2. *Flooding Attack*: In this case, we consider the malformed packets, i.e. each packet must possess at least one incorrect bit, according to the standard, either in NPDU or APDU. Since BACnet devices do not drop the packets

but instead try to accept and process any request, the incorrectly set bits in NPCI and APCI pose a threat to them.
Thus, an attacker can break a device by sending a flood of identical or different packets, making the number of packets received by the device far higher than normal. As a consequence, this can cause a denial-of-service, or force unintentional behavior by the device.

Vulnerable protocol design. By analyzing the protocol structure of BACnet and behavior of devices during communication procedure of certain messages, we categorize the following attacks.

1. *Smurf Attack*: In BACnet, if an attacker is able to modify the source (SADR and SNET) at the network layer, he will be able to spoof the address of broadcast requests and can cause a denial-of-service for selected BACnet devices. A smurf attack on BACnet is feasible as many BACnet devices are either old (BAS hardware is seldom altered over decades) or possess substantially lower processing capabilities than today's desktop computers and smart phones.
2. *Router Advertisement Flooding*: A similar attack is possible if an attacker is able to spoof a target device's source address and source network (SADR and SNET) to send a *Who-is-Router-to-Network* message (requesting a router advertisement for a given network). The result is that the target will receive router advertisements from all the routers in the local network. If the attacker repeats this procedure and sends too many repeated messages, the target is likely to receive more responses in a time window than it can normally handle, causing a denial-of-service.
3. *Traffic Redirection*: An attacker can spoof *I-Am-Router-to-Network* [12] or Router-Available-to-Network messages, i.e. messages indicating the availability of a router, with the goal to redirect selected traffic over itself to gain confidential monitoring data (e.g., presence sensor data of a given room to plan a physical break-in [18]).
4. *Re-Routing DoS, Type 1*: To cause a message flood on a router R or a BBMD, an attacker can broadcast spoofed *I-Am-Router-To-Network* messages to the network using the source address of R. Therefore, all possible destination network addresses can be used as a parameter for the router advertisement. This attack forces R to handle all responses to the *I-Am-Router-To-Network* message and, moreover, forces R to handle all remote traffic.
5. *Re-Routing DoS, Type 2*: If the target device of the Type 1 attack is not a router, an attacker can redirect all the traffic for remote networks to a device incapable of forwarding messages, thus, isolating the communication of a subnet. The scenario is similar to the one where the victim's address can be spoofed in a *Router-Available-to-Network* message. It is important to mention that broadcast floods in BACnet networks can also be caused by devices which are not configured properly. At least three examples from practice are known [13]: i) the wrong setup of layer two switches that can lead to loops;

ii) the use of multiple broadcasting BBMDs in a chain without a broadcast-limiting router device in the chain; iii) the combination of BACnet/Ethernet ISO 8802-3 and BACnet/IP routers within the same infrastructure configured to use the same UDP port (leads to permanent broadcast exchanges between the two layers).

5 A Snort-Based BACnet Normalizer

We implemented a Snort-based normalizer extension capable of normalizing BACnet/IP traffic based on a configuration file. Supporting BACnet/IP allows extending our work to non-IP-based data link-layer protocols used by BACnet. The Snort extension includes countermeasures for each discussed attack vectors. The rules serve to remove ambiguities within the traffic in order to achieve compliant traffic.

5.1 Standard conformity

Ensuring robustness for the protocol stacks of BACnet devices is essential as firmware is seldom updated. Therefore, malformed packets violating the rules of the BACnet standard must be modified or discarded to achieve specificity. We list countermeasures in the form of normalization rules for the variants of malformed packets which succeed in compromising a BACnet device or the whole network, as mentioned in Sect. 4. The rules are split into five categories: NPCI field, BACnet non-security message types, BACnet security message types, APCI field, and the handling of BACnet priority messages.

NPCI field. Being an ever present component of a BACnet NPDU, the NPCI field including the NPCI control octet (CO) can always be normalized. Reasons to **DROP** messages:

1. Protocol Version Number != 0x01
2. DNET = 0, or SNET = 0, or SNET = 0xFFFF
3. Multicasts and local broadcasts with DNET=0xFFFFFFFFFFFF using ISO 8802-3, DNET=0x00 using ARCNET or LonTalk, DNET=0xFF using MS/TP
4. Bit 3 of CO is 1 and SLEN = 0
5. Unicast message with DNET = 0xFFFF

Reasons to **MODIFY** messages:

1. Set DLEN = 0 and DADR=0 if the message is a remote broadcast
2. Set bits 6 and 4 of CO to 0

BACnet Non-Security Message Types. As explained in Sect. 3, bit 7 of the NPCI control octet indicates whether a BACnet message represents a network layer message or an APDU. Table 1 shows the possible network layer message types. We determined the following normalization rules for non-security network layer message types. We **DROP** the message, if the message type is any of the following:

Table 1. BACnet Network Layer Message Types. Security message types are marked with *.

Type	Description
0x00	Who-Is-Router-To-Network
0x01	I-Am-Router-To-Network
0x02	I-Could-Be-Router-To-Network
0x03	Reject-Message-To-Network
0x04	Router-Busy-To-Network
0x05	Router-Available-To-Network
0x06	Initialize-Routing-Table
0x07	Initialize-Routing-Table-Ack
0x08	Establish-Connection-To-Network
0x09	Disconnect-Connection-To-Network
0x0A*	Challenge-Request

Type	Description
0x0B*	Security-Payload
0x0C*	Security-Response
0x0D*	Request-Key-Update
0x0E*	Update-Key-Set
0x0F*	Update-Distribution-Key
0x10*	Request-Master-Key
0x11*	Set-Master-Key
0x12	What-Is-Network-Number
0x13	Network-Number-Is
0x14-0x7F	Reserved for use by ASHRAE
0x80-0xFF	Available for vendor proprietary messages

1. 0x02, 0x03, 0x08 or 0x13, and 4 or more bytes follow the type field
2. 0x01, 0x04 or 0x05, and an odd number of bytes follows the type field
3. 0x06 or 0x07, and more than 4× NUMBER_OF_PORTS + Sum of all PORT_INFO_LENGTHs bytes follow the type field
4. 0x00 or 0x09, and more than 2 bytes follow after message type field
5. 0x00: Limit the number of messages to m per second
6. 0x01 and the message is not transmitted with a broadcast MAC
7. 0x12 and the message is not transmitted with a local address, or if Hop-Count > 0, or if SADR is the same during n minutes
8. 0x13 and SNET/SADR or DNET/DADR is set or the message is sent to a local unicast address

Parameters m and n are configurable and depend on the particular hardware used (for example, in collaboration with industry partners we determined $m = 180$ as a reasonable value).

BACnet Security Message Types. We define rules for security messages as stated in [17]. Error messages in each case should always be sent signed-trusted. We **DROP** the message, if the message type is any of the following:

1. 0x0A, 0x0E, 0x0F or 0x11, and the message is broadcast
2. 0x0A and more than 9 bytes follow the type field
3. 0x0B and more payload is transferred than specified
4. 0x0C and
   (a) the RESPONSE_CODE (RC) is 0x06 and the length ℓ of RESPONSE_SPECIFIC_PARAMETERS is > 4, or
   (b) RC=0x07 and $\ell > 2$, or
   (c) RC=0x0E and ℓ is even and the first byte is not 0x00, or
   (d) RC=0x0F and $\ell > 2$, or
   (e) RC=0x15 and $\ell > 1$, or
   (f) RC=0x16 and $\ell > 3$, or
   (g) RC=0x17 and $\ell > 2$, or
   (h) RC=0x18 and $\ell > 1$
5. 0x0D and more than 19 bytes follow the type field
6. 0x0E and more than 21 bytes + bytes of keys follow the type field

7. 0x0F and more than 1 byte + bytes of keys follow the type field

We **MODIFY** the messages as follows:

1. Set bit 2 of CO to 1 if the type is 0x0A, 0x0D, 0x0E, 0x0F or 0x11
2. Set bit 2 of CO to 0 if the type is 0x0C or 0x10

APCI field. Whenever bit 7 of the NPCI control octet is 0, the content of the NSDU is an APDU, in which case the APCI field is present and can thus be normalized. Therefore, we implemented additional normalization rules. We **MODIFY** the messages as follows (cf. Fig. 3):

1. Set the first bit in PDU Type to 0
2. Set bits 7 and 8 of APCI to 0 if PDU Type is 0
3. Set bits 4 – 7 of APCI to 0 if PDU Type is 1, 2, 5 or 6
4. Set bits 6 and 7 of APCI to 0 if PDU Type is 3
5. Set bits 4 and 5 of APCI to 0 if PDU Type is 4
6. Set bits 4 – 6 of APCI to 0 if PDU Type is 7

Handling of BACnet Priority Messages. BACnet allows assigning each message a *priority* [1, pp. 68]. The priority is indicated by bits 1 and 0 within the NPCI control octet (11=*Life Safety*, 10=*Critical Equip*, 01=*Urgent*, 00=*Normal*). The highest possible priority of a packet is the life safety message and can be handled in a prioritized way by the receiving devices. However, in practice this feature is rarely used. We aim to introduce normalization for packets of all priority levels if they are not well-formed according to the ISO standard [1]. Nevertheless, we must take into account that not all malformed packets are caused by an attacker, and that dropping a life safety message can result in significant side-effects if the message is not delivered but contains life-essential information. Therefore, we require that life safety messages must be modified (in order to match compliant NPDUs according to the standard as closely as possible) and always forwarded, and should never be dropped. We are aware of the fact that an attacker could explicitly take advantage of such a rule. In order to mitigate such attacks, one could require that BACnet messages sent with life-safety priority MUST always have a trusted level, i.e. they must be either encrypted or physically secured according to [17], so that the receiving device is aware of the sending device. Life safety messages not encoded this way would be dropped. This approach, however, requires the support in the devices, which is not always possible.

5.2 Prevention of Network Covert Channels

The covert storage channels that an attacker could embed in the reserved bits of the BACnet NPDU and APDU are disabled, as the normalizer clears these bits. It is important to mention that other covert channels (especially timing channels) cannot be eliminated with this approach and that many additional hiding techniques for network covert channels are available. To counter additional covert channels, extensions for caching packets before forwarding them would be required, in order to limit the capacity of covert timing channels.

5.3 Closing Protocol Security Flaws

Unconstrained broadcasting in BACnet networks is a problem and allows for a wide spectrum of attacks, as outlined in Sect. 4. In order to prevent flooding and spoofing attacks, the appropriate limits for broadcast messages per time interval must be defined. These limits depend on the particular BACnet devices and are thus vendor-specific. In one empirical study, performed in co-operation with a vendor of BACnet products, we measured that most tested BACnet devices cannot process more than 180 messages per second.

6 BACnet Testbed

Our test environment is implemented using the open source BACnet stack (available at `http://bacnet.sourceforge.net`) and the virtual machines, each representing multiple BACnet devices (see Fig. 5).

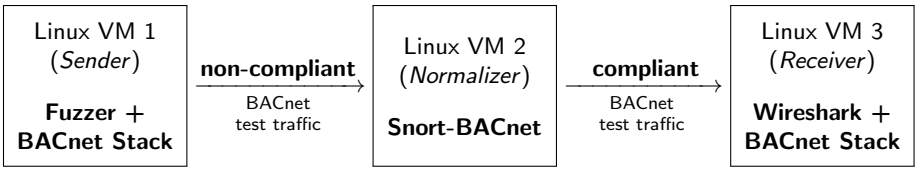

Fig. 5. BACnet testbed for evaluating the Snort extension [13].

We setup the testing environment using three virtual machines (VMs). VM 1 represents the *attacker* who sends non-compliant BACnet traffic using the *fuzzer*. The messages are first examined by the *normalizer* (VM 2) before they are forwarded to the *virtual BACnet device* (VM 3). The destination host VM 3 monitors the received packets using Wireshark.

The fuzzer is implemented using the Scapy packet manipulation tool [19]. It sends invalid and malformed packets to our test system in order to measure the behavior and the performance of the test system. The fuzzer follows the rules related to the structure of the messages as described in the ISO standard. To simulate the denial-of-service scenarios we implemented the packet sending in C, thus achieving very high packet send rates (upwards of 800,000 packets per second).

The normalizer is created as an extension of the existing Snort [6] code with our normalization rules for BACnet messages. Our extension is able to recognize BACnet messages which are sent through UDP over IPv4. Each byte of the NPDU and APDU can be analyzed in order to decide whether to forward, drop or modify each packet according to a predefined rule set.

7 Evaluation and Future Work

The purpose of implementing the testbed is to verify the correctness and the performance of the *Snort-based normalizer* by testing the prevention of attacks (described in Sect. 4). To achieve this, we divided the BACnet/IP packets into two sets, for each normalization rule (rules described in Sect. 5). The first set contained non-compliant and malformed packets and the second set contained compliant and legitimate packets. We created at least one malformed packet for each normalization rule individually. Each set was further subdivided into various subsets of different *message types*. We created different unit and generic test cases to examine the behavior of packets from both sets.

As per the test environment setup, the messages were sent from VM 1 to VM 3. For generic testing, we transmitted the messages using a fuzzer which has the capability to send thousands of messages of a particular message type at a time. This helps to evaluate the performance of the system in handling flooding of messages. We tested scenarios with between 10,000 and 100,000 messages. By using Wireshark on the destination VM (VM 3), we observed that the non-compliant messages were either dropped/blocked or modified correctly, and compliant messages were transmitted to the destination without any interruptions, so that all the received packets were handled as stated in Sect. 5. We carried out this scenario (as stress-testing) to measure performance of the destination, VM 3 – the virtual machine representing the attacked BACnet – with and without a preceding normalizer. We flooded the device with a various numbers of malformed packets. Without the normalizer, the target device was unable to cope with the traffic. When the normalizer was enabled, none of the malformed packets reached the device. During the test, the CPU load on the normalizer VM (VM 2) was below 65% at all times.

For unit testing, we created 48 non-compliant test messages for different message types. Each packet was created to test the normalization rules laid down in Sect. 5. We also created 45 compliant test messages. These test messages were sent individually to test the behavior of the normalizer VM 2. Our tests showed that the normalizer was clearly able to differentiate between compliant and non-compliant traffic and performed necessary actions whenever required before forwarding packets to the destination VM 3.

Our future work will focus on the problem of detecting abnormal behavior of infected or exploited BACnet devices. Čeleda et al. [12] introduced a network flow-based detection tool and provided a theoretical comparison to determine anomalies in BACnet control flows using their network monitor tool. The authors were able to record key information in order to detect a flood. Prevention techniques, however, were not implemented in the tool. We plan to expand our Snort normalizer extension to include detection and prevention of abnormal traffic. By analyzing data collected from actual BACnet networks we will develop application-specific rules and create a state machine that can distinguish abnormal states from compliant ones. Furthermore, we plan to expand the normalizer to handle segmentation-based attacks by performing packet reassembly and caching within the normalizer.

References

1. ISO 16484-5:2012 Building automation and control systems – Part 5: Data communication protocol.
2. Merz, H., Hansemann, T., Hübner, C.: Building Automation: Communication systems with EIB/KNX, LON and BACnet. Signals and Communication Technology, Springer, 2009
3. Proofpoint Inc.: Proofpoint Uncovers Internet of Things (IoT) Cyberattack. Report, January 2014. `http://goo.gl/ENgpTR`
4. Wendzel, S., Zwanger, V., Meier, M., Szlósarczyk, S.: Envisioning Smart Building Botnets. GI Sicherheit, LNI 228, 319–329 (2014)
5. Malan, G.R., Watson, D., Jahanian F. and Howell, P.: Transport and application protocol scrubbing, Proc. IEEE Conf. Computer Communications (INFOCOM), 1381–1390, 2000.
6. Snort – open source network intrusion prevention system and network intrusion detection system. `https://www.snort.org/`
7. Handley, M., Paxson, V. and Kreibich, C.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In Proc. USENIX Security Symposium, Berkeley, 2001.
8. Holmberg, D. G.: Enemies at the gates. BACnet Today, pp. B24–B28, 2003.
9. Soucek, S. and Zucker, G.: Current developments and challenges in building automation. In: e&i (Elektrotechnik und Informationstechnik), 129(4), 278–285, 2012.
10. Wendzel, S., Kahler, B., Rist, T.: Covert Channels and their Prevention in Building Automation Protocols – A Prototype Exemplified Using BACnet. In Proc. 2nd Workshop on Security of Systems and Software Resiliency, 731–736. IEEE, 2012.
11. Granzer, W., Kastner, W., Neugschwandtner, G. and Praus, F.: Security in networked building automation systems. In Proc. 2006 IEEE International Workshop on Factory Communication Systems, 283–292, 2006.
12. Čeleda, P., Krejcí, R., Krmícek, V.: Flow-Based Security Issue Detection in Building Automation and Control Networks. In Information and Communication Technologies, LNCS 7479, 64–75 (2012)
13. Szlósarczyk, S., Wendzel, S., Kaur, J., Meier, M., Schubert, F.: Towards Suppressing Attacks on and Improving Resilience of Building Automation Systems - an Approach Exemplified Using BACnet. GI Sicherheit, LNI 228, 407–418 (2014)
14. Bowers, B.: How to Own a Building: Exploiting the Physical World with BacNET and the BACnet Attack Framework, Shmoocon 2013. `http://goo.gl/Ea1LZu`
15. Holmberg, D. G., Bender, J., and Galler, M.: Using the BACnet firewall router. BACnet Today, pp. B10–B14, November 2006.
16. Tom, S.: BACnet for a City – Saving Energy one Small Building at a Time; BACnet Today and the Smart Grid, pp. B4–B9, November 2012.
17. ASHRAE: Proposed Addendum ai to Standard 135-2012, BACnet – A Data Communication Protocol for Building Automation and Control Networks, 2014.
18. Wendzel, S.: The Problem of Traffic Normalization in a Covert Channel's Network Environment Learning Phase. Sicherheit 2012, pp. 149–161, LNI 195, GI, 2012.
19. Biondi, P., the Scapy community: Scapy Documentation, Release 2.1.1, 2010. `http://goo.gl/nPEUFx`