Security Issues with BACnet Value Handling

Matthew Peacock, Michael N. Johnstone and Craig Valli
Security Research Institute, Edith Cowan University, Perth, Australia
{*m.peacock, m.johnstone, c.valli*}*@ecu.edu.au*

Keywords: Building Automation, State Modeling, Security, Heating Ventilation, Air Conditioning.

Abstract: Building automation systems, or building management systems, control services such as heating, air-conditioning and security access in facilities. A common protocol used to transmit data regarding the status of components is BACnet. Unfortunately, whilst security is included in the BACnet standard, it is rarely implemented by vendors of building automation systems. This lack of attention to security can lead to vulnerabilities in the protocol being exploited with the result that the systems and the buildings they control can be compromised. This paper describes a proof-of-concept protocol attack on a BACnet system and examines the potential of modeling the basis of the attack.

1 INTRODUCTION

Building automation systems (BAS) rely on reporting and logging to relay communications between devices automatically to ensure proper coordination and operation of a buildings services. From a security standpoint, protecting reporting and subsequently, the operation of the BAS is a priority for the continuity of building operation. In regards to BACnet, a popular BAS protocol; the overuse of the change of value (CoV) reporting method can cause network peaks, resulting in denial of service. Recommended implementation-based solutions shift the problem from network throughput to a security issue, where use of CoV is limited by a buffer, which can prevent core devices from receiving CoV notifications (Chipkin, 2009). In addition, as operation of BAS devices can cause cyber-physical actions to occur, priority levels for commands must be defined to resolve conflicts. The implementation of priority within BACnet on cyber-physical properties lack clarity. For example, in the current release of the BACnet standard, system behaviour is undefined when multiple devices write to the same property with the same priority level (SSPC-135, 2012). This presents a potential vulnerability as malicious writes represented as legitimate value changes and the ensuing events could have destructive consequences for the building controlled by the BACnetwork.

The paper will discuss the BACnet structure, followed by data value handling, and related security issues. A theoretical attack is presented, with an experimental setup discussed. Further, examination of modeling approaches to generate features and subsequently alerts for malicious events, to improve the resilience of building systems is discussed.

2 BACKGROUND

Security issues in BAS are an emergent threat, with security analysis undertaken by (Holmberg, 2003), (Kastner et al., 2005), (Granzer and Kastner, 2010) and later (Peacock and Johnstone, 2014). (Holmberg, 2003), (Kastner et al., 2005) identify, denial of service, eavesdropping and buffer overflows in core functions of the protocol. While (Granzer and Kastner, 2010) primarily discuss the limitations of secure communication, exchange in BAS. (Holmberg, 2003) and (Kastner et al., 2005) identify the increased connectivity between previously isolated BAS networks, corporate intranets and the Internet for remote management; an increasing trend, identified by (Peacock and Johnstone, 2014). A number of these identified vulnerabilities in BACnet have had mitigations suggested such as the BACnet firewall (Holmberg et al., 2006), however, (Peacock and Johnstone, 2014) identifies the potential of legitimate-yet-malicious commands operating inside BACnetworks, which would not pass traffic through a border firewall. A potential solution is a BAS specific intrusion detection system (IDS), the focus of work by (Johnstone et al., 2015) and (Kaur et al., 2015).

Peacock, M., Johnstone, M. and Valli, C.
Security Issues with BACnet Value Handling.
DOI: 10.5220/0006263405460552
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 546-552
ISBN: 978-989-758-209-7

3 BACnet

BACnet is an object-oriented peer-based protocol for managing building automation systems. Released in 1995 as an ASHRAE/ANSI standard, BACnet gained ISO standardisation (ISO 16 484-5) in 2003. BACnet is actively maintained, with reviews of the protocol occurring every four years until 2008, thence changing to biennial reviews (SSPC-135, 2014). BACnet focuses on the network layer and above, with the goal of being operable on any data link and physical medium. The BACnet standard defines data structures to represent the communications and devices on a BACnetwork. The core data structure is an object, of which there are 54 standard types. When combined, objects can represent a device, with each object containing properties which further define the features of the object. Similarly, communications on the network are defined in the BACnet standard as services, of which there are 38 types (SSPC-135, 2012). A core process of BACnet is communicating values across the network between devices to allow automatic actions to occur.

3.1 Data Collection in BACnet

Devices require correct, timely information in order to act appropriately to events occurring in a building. In BACnet, reportable information is defined as either value-based or event-based, both of which are time-stamped. As BACnet is a peer-based network, any device in a BACnetwork may request values from other devices, or be notified of events occurring. To accomodate peer-based communications, by default BACnet devices are passive servers which listen for requests and service received requests. Each data sharing transaction is represented as a client/server request, where the device requesting data is a client, and the targeted device providing the data is the server. A server in this case might be a temperature sensor, whilst a client might be a HVAC controller. Depending on the data type, either value or event, the retrieval process differs. Value data is shared using one or a multiple of polling, change of value (CoV) reporting or triggered collection; while event data uses event logs to collect notifications. The three methods mentioned for value based reporting are implemented using a *Trend Log object*, which can monitor one object or property on a device. When a value changes, a notification is stored in an internal buffer in the *Trend log object*, as a *Trend log record*. Each record holds the timestamp of when the change occurred, and the changed value of the monitored property.

Polling uses the default passive model previously discussed, where data retrieval queries are made to the server device at defined time intervals. A problem with polling is the potential for state and value changes between polling intervals, the active value changes of the system will not be updated unless they exist when a query is sent. A careful balance is required, as increasing the frequency of polling can impact the throughput of the network significantly (Chipkin, 2009). An alternative is an active data collection method called CoV reporting. CoV reporting uses a subscription-based method, where a client may request a subscription to a particular object or property value, shown as Figure 1. The subscription details the property to monitor, the criteria for issuance of notifications, and a lifetime for the subscription. The CoV reporting method may be *Confirmed* with acknowledgement of notifications, or *Unconfirmed* without acknowledgement. A third method is triggered collection, which defines a boolean property in a device, when the property is true, data is retrieved from the device. External network writes and internal processes such as alarms or other events can cause the trigger to become true, and cause an immediate acquisition of values to occur. Any property may be monitored using a *Trend log object*, and the property information stored in the buffer can be of any data type; making the buffer an unrestricted data type storage area, and potential vulnerability.

3.2 Conflict Resolution

As mentioned previously, all BACnet devices are peers, meaning any device, connected to the network may write to any writable property on any other device. As some property values directly cause cyber-physical actions to occur, conflict resolution in the form of a priority system is implemented. BACnet accounts for the potential of conflicting commands through a conflict resolution process where properties are split into commandable, and writeable types. Commandable properties are defined as those whose value change causes physical actions, while all other properties are defined as writeable. Conflict resolution is only applied to commandable properties, whereas writable properties have no priority mechanism, meaning the last write to the property overwrites the previous value.
The *present value* property of most objects in BACnet are classed as commandable properties, see Table 1 for a full listing.

BACnet devices interact with commandable properties using the *WriteProperty* or *WritePropertyMultiple* service requests. The request primitive for both services contain three parameters; Property Identifier,

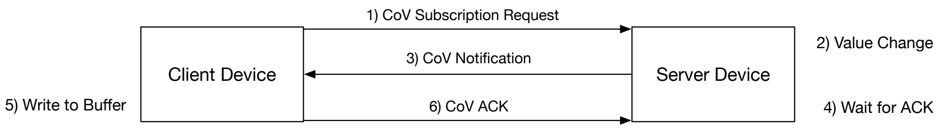

Figure 1: Confirmed CoV transaction.

Table 1: Commandable properties in BACnet adapted from ASHRAE STD 2012.

Object	Commandable property
Analog Output	Present_Value
Binary Output	Present_Value
Multi-state Output	Present_Value
Multi-state Value	Present_Value
Analog Value	Present_Value
Binary Value	Present_Value
Access Door	Present_Value
BitString Value	Present_Value
CharacterString Value	Present_Value
Date Value	Present_Value
Date Pattern Value	Present_Value
DateTime Value	Present_Value
DateTime Pattern Value	Present_Value
Large Analog Value	Present_Value
OctetSTring Value	Present_Value
Integer Value	Present_Value
Time Value	Present_Value
Time Pattern Value	Present_Value
Positive Integer	Present_Value
Channel	Present_Value
Lighting Output	Present_Value

Property Value and Priority. The three parameters contain the commandable properties ID, the desired value for the property, and the priority value respectively. The priority value is a number set between 1 and 16 for that *WriteProperty* service request, the lowest number having the highest priority. The BACnet standard defines consistent representations for the applications of command priority levels, with 5 defined applications and 11 flexible open applications which can be implementation specific, shown as Table 2. When a client device no longer requires access to the commandable property in a service provider, a relinquish command is sent to the provider, using *WriteProperty* or *WritePropertyMultiple* service requests. The relinquish request uses the same parameters as a normal write request with the property value parameter set to *NULL*. When a device is notified that no service request exists at that priority level, the next priority array element is checked for a service request. When all elements in the priority array are *NULL* the property value will be set to the value defined in the *Relinquish_Default* property of the object.

Table 2: BACnet Priority array applications, adapted from Newman, 2013.

Priority level	Application
1	Manual_Life Safety
2	Automatic_Life Safety
3	Available
4	Available
5	Critical equipment control
6	Minimum on/off
7	Available
8	Manual Operator
9	Available
10	Available
11	Available
12	Available
13	Available
14	Available
15	Available
16	Available

3.3 BACnet Security

BACnet was not designed with security as a primary requirement, with early systems operating in isolated networks without any external connection. With the increasing connection of BACnetworks to external facing networks, such as the Internet or enterprise networks for maintenance, the attack surface has increased. Holmberg (Holmberg, 2003), documented a number of the threats to BACnet, with further work being undertaken by (Kastner et al., 2005) to increase the security features of BACnet. In 2009, the suggestions from the threat assessment were ratified as BACnet security services (BSS), as Addendum g to the BACnet standard. Despite being an official addendum to the standard, BSS is not widely implemented, with Newman, the original chairman of BACnet stating “no company has yet implemented it in a commercially available product” (Newman, 2013). Additionally, Newman(Newman, 2013) suggests that BACnet/IP could be secured using IP security solutions, such as IPsec, TLS, or Kerberos. In contrast, given that a number of BACnet vulnerabilities are not solved by encryption (as in the aforementioned protocols), but rely on authentication instead, this indicates that the coverage of the attack surface is incomplete and therefore not all BACnet vulnerabilities are mitigated.

4 SECURITY ISSUES IN VALUE CHANGES

In a BACnetwork, devices can operate via a client-server architectural model where the "server" is the service provider (SP) or transmitting device (such as a thermometer) and the "client" is a controller, such as an air handling unit, or human machine interface. This model has the following properties:

1. One (or more) clients subscribe to the SP
2. Requests can be queued
3. The queue is stored on the SP

This model is well-known as the publish/subscribe model. In software development, this is implemented by the Observer design pattern (Gamma et al., 1995). Given that BACnet is object-oriented (or at least claims to be so), it is useful to briefly examine the principles of the Observer pattern as this informs our discussion about a specific BACnet vulnerability.

The Observer pattern describes a one-to-many connection between objects such that when an object undergoes a state (data) change, its connected (dependent) objects are notified automatically. Conventionally, the object that is the source of notification events is called the subject and the observer objects register with the subject in order to be notified of changes in the subject. As the subject is the sole owner of the data, the observers are reliant on the subject to update them when the data changes (i.e. a state change). Consistent with the loose coupling exhibited by many design patterns, observers can be added or removed from a list maintained by the subject at any time. The Observer pattern is quite common and is found in the model-view-controller architecture (and thus in many user interface toolkits, such as Java Swing). A concrete example would be the buttons that might be found on a web form are subjects that notify action listeners (observers) of events (e.g., a mouse click).

Returning to BACnet, the SP acts as the subject and maintains a list of subscribers (observers), namely the *Active_CoV_subscriptions* is the list which holds the CoV subscriptions on the SP. As mentioned, CoV subscriptions may be either *Confirmed* or *Unconfirmed*, a data service type reminiscent of TCP or UDP in operation. *Unconfirmed* CoV reporting sends a notification to the subscriber when a value changes within the CoV threshold of the subscription. *Confirmed* CoV reporting incorporates acknowledgement of change, with an ACK packet to be sent from the subscriber to the device serving the data.

Due to the many media on which BACnet operates, some devices take longer to acknowledge a *Confirmed* CoV than others. Two device object properties, *APDU_timeout* and *Number_Of_APDU_Retries* set on the client device determine how long to wait for an acknowledgement, and how many times to retry waiting. The values for these properties are vendor specific, the BACnet standard suggests between 6000ms, and 10,000ms(Newman, 2013); the de facto standard set by the vendors is a 3000ms wait time, with 3 retries see Table 3. Some vendor guides suggest a 20,000ms or even 60,000ms wait time, dependent on the capability of older devices; one guide suggests all *APDU_timeouts* should be set to the highest value in the system. If a subscriber is offline when a *Confirmed* CoV notification is sent, the CoV server device will wait the length of the *APDU_Timeout* of the client device, and then retry the CoV notification the specified number of times before processing the next CoV notification. Given the length of some *APDU_timeout* values and the number of retries, significant network delays can occur (Chipkin, 2009). Additionally, when a subscriber goes offline, CoV messages are not stored or queued, therefore if a subscriber returns to the network, data synchronisation can be lost. If the subscription is *Unconfirmed*, there is no feasible way to determine if the subscriber has received the CoV notification, or tell if the subscribing device is offline. A combination of polling and CoV is suggested to counteract devices power cycling, however the solution is not complete, as the logging device for the system could suffer from the same issue, and the log of a CoV will never be recorded (Chipkin, 2009). Subscriptions to devices are not persistent between power cycles, meaning if a device is reset for any reason, the subscriptions to other devices will not be preserved and must be re-connected.

Table 3: BACnet Vendor APDU-Timeout default values.

Vendor	APDU_timeout value (ms)	APDU retries
Viking Controls	3000/60,000	3
Siemens	3000	3
Obvius	6000	3
ScadaEngine	500	5
Kepware	1000	3
Tridium	20,000	3
Contemporary Controls	3000	3

A solution exists for automatic re-subscriptions, where a duration property exists in the *Trend log object*, which triggers a CoV subscription to occur. However, the integrity of the BACnetwork is then dependent on client devices re-subscribing to keep a synchronised and robust system. Due to the capacity of BACnetworks, oversubscription of CoV notifications is plausible. Robust testing of oversubscription is often not carried out, due to the risk of damage to devices (Chipkin, 2009; Newman, 2013). An implementation guideline solution suggests limiting the number of subscriptions a device is capable of holding, to reduce the potential traffic density. A device on the BACnetwork may subscribe to the same object multiple times, as the unique identifier for each subscription is self-assigned. Each implementation of a device may have a limit applied to the quantity of subscriptions that each device can initiate. This bottleneck creates a network security issue, where critical devices will not receive notifications due to the subscription limit being reached (malicious or not).

4.1 Issues with Bounded Priority Arrays

An identified issue with the array implementation of BACnet is that each priority value in the array may have only one command stored at a time. As described, upon relinquishing an array position, a device will write a *NULL* value to the array position, which causes "unknown behaviour" for the queued command in the same array position (SSPC-135, 2012). As source authentication is not specified for write commands entering the priority table, the priority array attempts to create a queue of values to enter into a commandable property. An issue similar to writable commands is created, where a write first wins scenario is created, negating the purpose of the array, when same priority commands are issued. Additionally, as there is no verification of priorities on commands, any device may change the value of a commandable property at any priority. Similar to *Confirmed* CoV notifications, these commandable property writes are legitimate traffic which could be expected to be made on the network. (Johnstone et al., 2015) assessed the possibility of detection using time deltas between same packets with success for writes with the highest priority. Potentially, the same type of detection could be applied to the priority of same packets from an identical source device to determine malicious behaviour.

4.2 Theoretical Attack Scenario

A potential scenario could manifest as a malicious device sending *Confirmed*, low threshold value CoV subscriptions to every supported device on the BACnetwork, and then shutting itself down. Whenever a value on any device changes the malicious device will be notified, but as the malicious device is offline, each legitimate device will then wait for the timeout to expire before sending the next notification. A model of the attack is detailed as Figure 2.

As the *APDU_timeout* property is defined on the client of the transaction, the malicious device may set the length to wait, and the number of retry attempts. (Newman, 2013). Detection of the attack may be achievable given the vendor timeout values detailed in Table 3, as values which differ drastically from the defacto-standards may be easily identified. However, as a device may subscribe multiple times to another device, a proposed attack chain involves subscribing within the normal time values, up to 20 seconds, with three retries and multiple subscriptions thus increasing the denial of service on the device by each subsequent subscription. The impact of manipulating the timeout values is dependent on the building services provided, and the criticality of the buildings contents. Preventing values being reported on the network is of serious concern, with a clear reduction in availability to the cyber-physical systems controlled on the network. Given that the commands used to cause the attack are legitimate, and can appear to be normal traffic on a BACnetwork, a different type of detection method, with contextual analysis of the network is required to identify these situations.

5 EXPERIMENTAL ANALYSIS

Experimentation is required to examine the impact of this attack, and the potential of identification and classification. As described, robust CoV testing is often not undertaken due to fear of network and physical degradation. As such, an experimental setup was developed for generation of network traffic, and manipulation of CoV values, the materials used are noted in Table 4. The experimental setup resembles a subset of

Table 4: Experimental Setup Materials.

Software/Hardware	Details
Raspbian Jessie	R2016-09-23, V4.4
CBMS Studio	V1.3.7.1204
CBMS Engineering configuration tool	V1.3.1221.7
Bacnet Open Stack	V0.82
Windows 7	SP1
Wireshark	V1.12.1
3x Raspberry Pi 2	1GB ram, 16GB SD card
Cisco 3560 Switch	SPAN configured
Windows 7 Desktop	16gb Ram, 128GB SSD

a BACnet controlled HVAC service. The air handling unit (AHU) acts as a client, subscribing to the thermostat *present_value* property, when a value change occurs the AHU is notified. A CBMS BACnet server instance runs on the thermostat Raspberry Pi, emu-

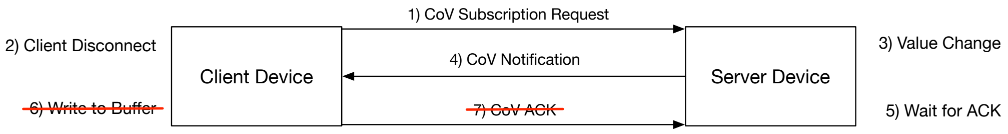

Figure 2: Malicious Confirmed CoV transaction.

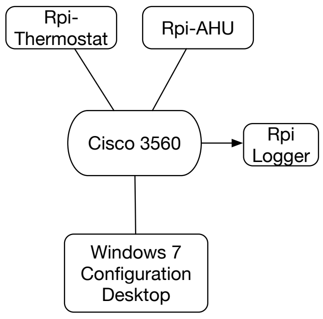

Figure 3: Experimental Setup.

lating a thermostat. The AHU subscription is undertaken using the BACnet protocol stack version 0.81. Configuration of the thermostat is undertaken using the CBMS engineering tool, running on the Windows 7 configuration desktop. All traffic on the network is port mirrored to the Raspberry Pi logger device, using Wireshark for packet capture and analysis. A graphical depiction of the experimental setup is shown as Figure 3.

Initial experimental results shown in Table 5 are encouraging, with a server device failing when a malicious client disconnected from the network. Packets 80040 and 80041 show a normal CoV notification and ACK details (shown as a reject due to a limited server implementation). Packets 80327, 80510 and 80656 show three attempted notifications, 10 seconds apart after the device has disconnected. During the 30 second window, the tracked server value was changed multiple times, none of these changes were reported as the server device waited for the acknowledgement packet from the disconnected client. Entry 81438 and 81439 show a value change and subsequent return to a normal notification and ACK, shortly after the client device was reconnected. No CoV notifications were generated by the server device while it waited for the ACK from an offline client. When the client reconnected, multiple changes in value that would normally trigger alerts were never disseminated to the client.

Further experimentation is being undertaken to incorporate further subscribing devices, to determine the impact on the subscribed device if multiple subscriptions exist, and one subscriber is not connected during a CoV occurrence.

Modelling the legitimate-yet-malicious scenarios may reveal further information, such as a network pattern or activity which could indicate a malicious event taking place from legitimate commands. Such a pattern or activity could be suitable for a detection rule.

6 DISCUSSION

We have described a vulnerability in BACnet that could have serious consequences to infrastructure. This vulnerability, in itself, is not the problem as we foresee a catastrophic second-order effect arising from the realisation of the threat. The full extent of this type of attack, and further attacks could be explored using an adversary model. Bodeau and Graubart (Bodeau and Graubart, 2013) outline the purpose of an adversary model as defining the intent, objective and strategy of a threat being realised, in relation to an adversary profile. As such, it would be beneficial to explore and define a profile for an adversary intent on disrupting building operations. A potential second-order attack could be crafted to take advantage of this vulnerability through delaying response to a temperature controller, adversely affecting a datacentre where servers that house financial data operate. The loss of banking data or even the time delay whilst such data are recovered from a backup are significant events for any financial institution.

Due to the time-dependent nature of cyber-physical systems, models that can express time are useful for verifying certain properties of the system. Clearly, the CoV communication process of the BACnet protocol, where time is a factor for successful completion of a command, is one such area.

It is likely that any modelling notation/technique that can express a queue and has some notion of state, could represent this type of problem and be able to verify aspects of the system. Z (Spivey, 1989), Communicating Sequential Processes (Hoare, 1978) or even possibly the Object Constraint Language ((OMG), 2014) would be excellent candidates.

Each of these formal notations could model the BACnet change of value reporting. The application to detection of these legitimate events may be linked to the time between events, therefore monitoring the timeout durations on devices waiting for acknowledgements is likely the way forward. The peer-based nature of the protocol, coupled with multiple sub-

Table 5: Initial experimentation results.

No.	Time	Source	Destination	Protocol	Length	Info
80040	19:58:32.5	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[67] device,9999 analog-output,1 present-value
80041	19:58:32.5	192.168.1.5	192.168.1.12	BACnet	60	Reject unrecognized-service[67]
80327	19:58:48.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device,9999 analog-output,1 present-value
80510	19:58:58.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device,9999 analog-output,1 present-value
80656	19:59:08.4	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[69] device,9999 analog-output,1 present-value
81438	19:59:48.5	192.168.1.12	192.168.1.5	BACnet	85	Confirmed-REQ confirmedCOVNotification[70] device,9999 analog-output,1 present-value
81439	19:59:48.5	192.168.1.5	192.168.1.12	BACnet	60	Reject unrecognized-service[70]

scriptions and source authentication means that there is the potential to have this duration value increased by a malicious host.

7 CONCLUSION

This paper described a proof-of-concept attack on any building automation system that uses the BACnet protocol change of value reporting function as part of their communication and control, and suggested that a formal proof of the attack would be valuable. We found that while BACnet has a security addendum to the standard which defines source authentication, this is not implemented in practice. We deployed an experimental setup to investigate the phenomena derived from the BACnet standard, with initial results promising. We defined a situation where multiple subscriptions and a lack of source authentication can cause a failure of critical infrastructure, leading to a more serious second-order effect, using banking systems as an example.

In future work, we intend to expand on our experimental testbed, to incorporate more subscribed devices, and widen the scope to other services. Further experiments will be undertaken, to test the array relinquish issue to determine what actually happens given network traces and memory allocation to the device. The CoV experiments will be expanded, with investigation into the use of different stack implementations, to verify if the behaviour can be generalised. Finally, we will explore modeling of the protocol to derive identification patterns, along with defining an adversary model for building automation.

REFERENCES

Bodeau, D. and Graubart, R. (2013). Characterizing effects on the cyber adversary: A vocabulary for analysis and assessment. Technical report, MITRE.

Chipkin, P. (2009). Bacnet for field technicians. Technical report, Chipkin Automation Systems.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Granzer, W. and Kastner, W. (2010). Communication services for secure building automation networks. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 3380–3385.

Hoare, C. A. R. (1978). Communicating sequential processes. *Commun. ACM*, 21(8):666–677.

Holmberg, D. G. (2003). Bacnet wide area network security threat assessment. Technical report, NIST.

Holmberg, D. G., Bender, J. J., and Galler, M. A. (2006). Using the bacnet firewall router. *ASHRAE American Society for Heating, Refrigeration and Air Conditioning Journal*, 48(11).

Johnstone, M. N., Peacock, M., and den Hartog, J. (2015). Timing attack detection on bacnet via a machine learning approach. In *Proceedings of the 13th Australian Information Security Management Conference*, pages pp57–64.

Kastner, W., Neugschwandtner, G., Soucek, S., and Newman, H. (2005). Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203.

Kaur, J., Tonejc, J., Wendzel, S., and Meier, M. (2015). Securing bacnet's pitfalls. In Federrath, H. and Gollmann, D., editors, *ICT Systems Security and Privacy Protection*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 616–629. Springer International Publishing.

Newman, H. M. (2013). *BACnet: The Global Standard for Building Automation and Control Networks.*

(OMG), O. M. G. (2014). *Object Constraint Language (OCL). Version 2.4.*

Peacock, M. and Johnstone, M. N. (2014). An analysis of security issues in building automation systems. In *Proceedings of the 12th Australian Information Security Management Conference*, pages 100–104.

Spivey, J. M. (1989). *The Z Notation: A Reference Manual.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

SSPC-135 (2012). Bacnet: A data communciation protocol for building automation and control networks.

SSPC-135 (2014). Bacnet addenda and companion standards.