On Automating BACnet Device Discovery and Property Identification

Michael Cash*, Shan Wang‡†, Bryan Pearson*, Qun Zhou*, Xinwen Fu*†
*Dept. of Computer Science, University of Central Florida, USA
Email: mcash001@knights.ucf.edu, bpearson@knights.ucf.edu, qun.zhou@ucf.edu
‡School of Cyber Science and Engineering, Southeast University
Email: wangshan@seu.edu.cn
†Dept. of Computer Science, University of Massachusetts Lowell, MA, USA
Email: xinwen_fu@uml.edu

***Abstract*—BACnet is the most popular inter-communication protocol in building automation systems (BAS) and has been deployed in a large scale. It is critical to scan and perform risk analysis of a BAS. Existing work identifies BACnet devices in a manual way and does not further discover their properties. In this paper, we design and implement an automatic tool to identify a BACnet device at a given IP and enumerate both standard and vendor-defined BACnet objects and properties. We applied our tool to a testbed real-world BAS system on a university campus and successfully validated the tool's effectiveness. Our tool is the first of its kind for risk assessment of the BAS, e.g., automatically scanning open smart buildings on the Internet. The video at https://youtu.be/YUfO8GQILxQ demonstrates that our toolkit may be used to remotely move a damper controlling a building's Heating, ventilation, and air conditioning (HVAC) system from the Internet and justifies the importance of using our tool for penetration testing of a BAS.**

1. Introduction

The BACnet protocol has allowed building automation systems (BAS) administrators to maintain central building communication control of legacy devices while simultaneously adapting to new building automation devices. However, with BACnet's use of security being optional, the lack of security makes these devices vulnerable [1], [2], [3]. BACnet is an open protocol, where the equipment between vendors can interoperate with one another without the need for proprietary equipment. This can be beneficial for adversaries, using BACnet's openness to intrude and manipulate BAS systems. This can lead to dire consequences if precautions are not taken. As the number of devices deployed to a BAS increases, it is critical to identify these BACnet devices and perform risk assessment of the entire BAS.

Current tools for scanning BACnet for the purpose of penetration testing only provide basic property identification or do not completely enumerate device contents, especially autonomously. Brandsteter and Reisinger [4] identify security tools for discovering building automation systems, such as the Nmap Scripting Engine (NSE) [5], Redpoint [6], and HVACScanner [7]. Gasser et al. [1] use ZMap [8] to port-scan the internet-wide network for publicly reachable BACnet devices, returning a list of IP addresses that are potentially for BACnet devices. Since standard ports for building control may be used for other purposes, the tool has a very high false positive rate. Their tool cannot confirm these IP addresses indeed belong to BACnet devices either.

We implement a BACnet tool to automatically discover and scan for object types and properties of BACnet devices. The BACnet automation tool confirms the identify of a BACnet device by scanning the target IP address using BACnet's `WhoIsIAm` request service. After an IP address is confirmed to belong to a BACnet device, our tool automatically enumerates BACnet object types and their corresponding properties. By automating the process of confirming the presence of BACnet devices and automating property and object enumeration, we provide a tool to display a larger outlook on the BAS system.

The major contributions of this paper are summarized as follows: (i) We carefully analyze the BACnet protocol and design and implement a tool to automatically confirm BACnet devices at given IP addresses and then scan for objects and associating properties of the BACnet devices. (ii) We have applied our tool to a real-world BAS on campus within a lab environment and validated the effectiveness of the tool. Our tool can be particularly used for risk assessment of a BAS, for example, scanning a campus network and identifying open and vulnerable BAS devices, objects and properties.

2. Background

Building automation systems (BAS) is used to automatically monitor and control the conditions of indoor environments in large buildings, such as heating, ventilation, and air-conditioning (HVAC) systems. The core benefits of BAS are energy conservation and performance improvement [9] [10]. In order to integrate various automated services that are developed individually, the BAS adopts a collection of communication protocols: LonWorks, KNX, and BACnet [11]. Of the three, BACnet is the most popular.

2.1. BACnet Protocol

BACnet is the standard communication protocol for Building Automation and Control Networks developed by

the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) [12]. BACnet has become the standard for inter-communication within BAS systems. Its increase in popularity is due to its role as a centralized communication protocol, allowing devices that use different building automation protocols to communicate. This gives building system administrators more flexibility in the deployment of building automation devices.

The BACnet protocol is based on a collapsed version of the Open Systems Interconnection (OSI) model [13]. As shown in Figure 1, it is composed of 4 layers: (i) The physical layer makes it possible for BACnet to be implemented on different network types, such as ISO 8802.3 Ethernet, ARCnet, and BACnet/IP. (ii) The data link layer provides access to the physical medium while organizing the data to be transmitted into data packets recognizable to the protocol. (iii) The network layer provides message routing from one BACnet network to another, regardless of the technologies used in the sublayers of the protocol. (iv) The application layer provides information about a device and methods for accessing that information. These are organized into BACnet objects and BACnet services.

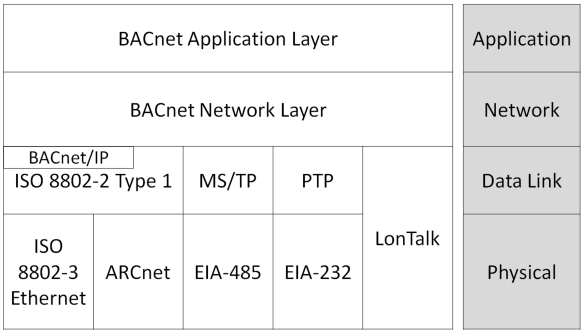

Figure 1. BACnet Protocol Architecture

2.2. BACnet Objects and Properties

To achieve interoperability between devices, the BACnet protocol defines a collection of data structures, known as BACnet objects. Each device in a BACnet network is a `Device` object which can define multiple other BACnet objects. Each object stores appropriate values based on that object's property. These properties characterize the BACnet object's functionality and purpose. Table 1 shows an example of an `Analog Output` object, which a building automation controller could use to access the readings for a temperature sensor. The left column shows the properties that can be presented in an Analog Output object. Some properties are required from every Analog Output object while others are optional.

2.3. BACnet Services

BACnet provides multiple services for acquiring data from BACnet objects, commanding BACnet devices to perform actions or announcing events that have occurred to other devices [14]. BACnet defines 5 service types: (i) **Alarm and Event Services** relay changes in conditions (e.g., if a threshold is reached) for a BACnet device. (ii) **File Access Services** read from and write to files contained on BACnet devices. These operations are atomic, meaning only one operation (read or write) can occur at a time. (iii) **Object Access Services** provide the methods for reading, modifying, and writing properties as well as adding and deleting objects. There is a `Read Property` method in this service, which can be used to read the value of target property, such as the `Present Value` of the Zone Temperature `Analog Output` object in table 1. (iv) **Remote Device Management Services** provide operational control, special message transfer, and configuration functions. For example, the `Who-Is` and `I-Am` in this service request the presence and basic information of a BACnet device. (v) **Virtual Terminal Services** provide bi-directional communication to programs which execute in a remote device.

TABLE 1. PROPERTIES OF AN ANALOG OUTPUT OBJECT

Property	Value	Type
Object Name	Zone Temperature	Required
Object Type	Analog Output	Required
Present Value	71	Required
Units	Degrees-Celsius	Required
Status Flags	Normal	Required
High Limit	75	Optional
Low Limit	60	Optional

Figure 2 shows a BACnet-based BAS example. Desigo CC is Siemens' facilities and building automation management system with a highly graphic interface. However, any tools following BAS protocols can be used to control facilities and buildings. For example, the Raspberry Pi in Figure 2 attempts to read the temperature from the temperature sensor, which is connected to the Siemens DXR2.E10PL-102B device (room automation station). The Pi sends a `ReadProperty` Request, which is a method of the Object Access Services, to the DXR2.E10PL-102B. The target property of this `ReadProperty` request is `Present Value` of the Analog Output object as shown in Table 1, which is an Analog Output object type. DXR2.E10PL-102B retrieves the `Present Value` reading of the Analog Output object, sending a `ReadProperty` response message back to the Pi saying that the `Present Value` of the Zone Temperature object is 71 °C.

3. Automating BACnet Devices Discovery

In this section, we introduce our methodology of identifying BACnet devices given an IP and enumerating objects and properties as well as basic information as seen with Nmap [5] and Redpoint [6] for the BACnet device. By doing so, we obtain a complete list of device contents. We identify both standard BACnet objects and vendor-defined objects.

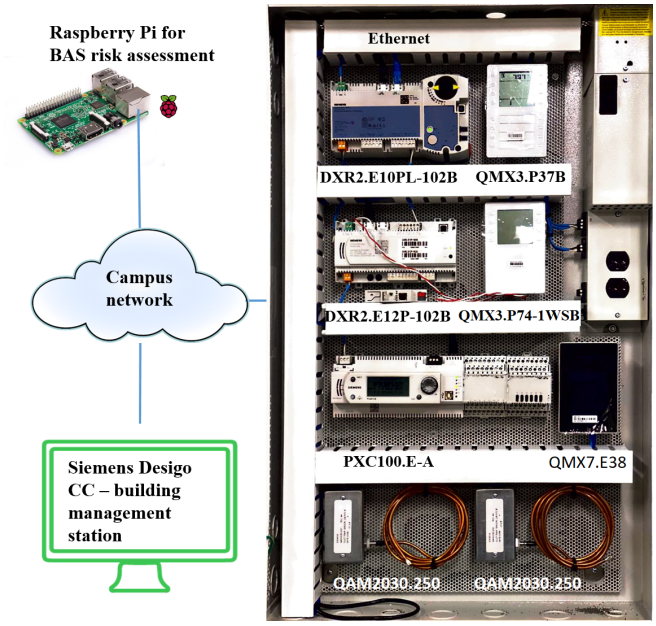

Figure 2. Example of BACnet-based BAS

For all standard BACnet objects, we further identify their standard properties.

3.1. Identifying BACnet Devices

We use the BACnet `Who-Is` and `I-Am` services to confirm the presence of a BACnet device located at an IP address. In the Remote Device Management Services of BACnet, the `Who-Is` and `I-Am` Services are two commonly used services. The `Who-Is` service is a special request (message) that is broadcast to the network in an attempt to obtain the network addresses of BACnet devices on the BACnet network that is specified by the request. When a BACnet device needs to know the address of another BACnet device it will broadcast a `Who-Is` service request, specifying the `Device` object type's instance number of the BACnet device to be known. The instance number is a unique number for an object type that differentiates itself from an object of the same type. If more than one BACnet device is to be requested, the `Who-Is` service request can also send a range of the instance numbers. Responses to these request, known as `I-Am` service response, are also broadcast to the network, providing the address information to the requester. By broadcasting the response, any other BACnet device which may need its address information can also receive the response without creating additional traffic on the network. Combined, these services remove the necessity of manually configuring addresses for each BACnet device on the network.

3.2. Standard and Vendor-Defined BACnet Objects Discovery

The BACnet protocol provides standard object types such as accumulator, analog input/output, command, file, program and schedule for vendors to communicate with building automation devices from other vendors. The list of standard object types continues to increase. The BACnet protocol's versatility also allows vendors to create specific BACnet objects, which are not accessible by other equipment used in the BAS. Vendor-defined object types do not interfere with the standard object types. When identifying all the objects for a BACnet device, the possibility of vendor defined objects must be considered.

Objects in BACnet are numerically coded, where the string name of the object equals a numeric value (i.e. analogOutput $= 1$ and binaryValue $= 5$). For standard object types, BACnet reserves the numeric codes 0-127. Codes 128-1023 are left for vendor object creation. Thus, if an object is encountered that possesses a numeric code within 0-127, we can confirm that it is a standard vendor-defined object type. If the numeric code of an object is within 128-1023, we can confirm that it is a vendor-defined object type.

A property called `objectList` of the `Device` object can be used to discover both standard and vendor-defined BACnet objects. The `Device` object represents a real-world device. It is a unique object defined by BACnet that contains metadata about the BACnet device. Table 2 shows an example of the `Device` object and its properties. The `objectList` property lists each object associated to the BACnet device. Using the `ReadProperty` method from the Object Access service, we can request the `objectList` property obtained in the `Device` object. Extracting this list provides a complete view of what objects exist on the BACnet device. Every BACnet device contains a `Device` object, and every `Device` object contains the `objectList` property. Using this method ensures discovering every object known to the BACnet device, including both standard objects and vendor-defined objects.

3.3. BACnet Properties Discovery

After we obtaining a device's BACnet objects, we identify the standard properties of all standard BACnet objects. Properties in BACnet also use numeric coding. BACnet reserves numeric codes 0-511 for standard properties. Any property using a numeric coding from 512 to 4194303 is a vendor-defined BACnet property. For example, the standard `Present Value` property in Figure 1 has a numeric code of 85. Here, we only identify the standard properties.

The BACnet service `Object Access Services` can be used to get the properties of BACnet objects. Since BACnet objects do not maintain a list of properties currently deployed, we check all candidates to identify the presence of properties. Fortunately, each standard object has their own subset from the total 512 properties. The subset is defined by the BACnet protocol. A method called `ReadProperty` in `Object Access Services` can be exploited. It accepts inputs such as IP address, object type and the property to be read. This method checks if the property exists. If the property does exist, this method returns its specified value. If the property does not exist, this method returns none. If the

TABLE 2. EXAMPLE OF PROPERTIES OF THE DEVICE OBJECT TYPE. THE 'OBJECTLIST' PROPERTY CONTAINS THE OBJECT IDENTIFIERS FOR ALL OBJECTS ON THE DEVICE.

Object Name = AS02
Object Identifier = ('device', 10001)
systemStatus = operational
vendorName = Siemens Building Technologies
vendorIdentifier = 7
modelName = DXR2.E12P-1
firmwareRevision = FW=01.21.50.128;WPC=1.7.10; SVS-300.4:SBC=13.22
applicationSoftwareVersion = AAS-20; AP=15020_VAV_US.3.001; SU=UsUn; APT=HvacLgtShd12_A; APTV=6.002;
location = B_01
protocolVersion = 1
protocolRevision = 13
protocolServicesSupported = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,]
protocolObjectTypesSupported = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]
objectList = [('analogInput', 6), ('analogInput', 32), ('analogInput', 56), ('analogInput', 57), ('analogInput', 58), ...]

property requires read access privileges, then it will output `readAccessDenied` instead.

3.4. Automatic Discovery Tool

The automation tool mainly utilizes the BACpypes library [15] [16]. BACpypes provides the application layer and network layer to allow for programming daemons, scripts, and graphical interfaces. The automation tool requires an input of a text file listing IP addresses. The tool will check whether the IP addresses are associated with BACnet devices. Algorithm 1 illustrates our process.

Accepting a text file listing IP addresses as an input, the automatic device discovery tool iterates through this list, sending a `Who-Is` request to each IP address. If an address sends back an appropriate `I-Am` response, the IP address is confirmed to be an associated BACnet device. The IP address and instance number of its `Device` object are stored in the list `device_addrs` and `device_ids`, respectively. These lists are used by the automation tool to identify all standard and vendor-defined objects for the BACnet device and also identify all standard properties of the standard objects. Both the `device_addrs` and `device_ids` lists will have same number of elements, with a one to one matching of IP address to instance number.

For each address and instance number pair in the lists, we send a `ReadProperty` request carrying the IP address, `Device` object type and `objectList` property [17]. This means the automation tool is requesting the `objectList`

Algorithm 1 Algorithm for Identifying BACnet Devices and Enumerating Objects and Properties

Input: addressList
Output: out
Initialisation :
Open addressList
Create lists $device\_ids$, $device\_addrs$
Create $object\_counter$, $property\_counter$
LOOP Process
for address K in $addressList$ **do**
  Create WhoIs request
  Send request
  if Response is $IAmRequest$ **then**
    Print pduSource, Device Identifier, Max Apdu Length Accepted, Segmentation Supported, VendorID
    Append K to $device\_addrs$
    Append Device instance number to $device\_ids$
  else
    Pass
  end if
end for
for address A, instance B in $device\_addrs$, $device\_ids$ **do**
  Create ReadProperty request for $ObjectList$ property
  Send request
  if Response is $ReadProperty$ response **then**
    Save $objectList$
    for Object O in $objectList$ **do**
      if Object O is Standard Object **then**
        for Each property P in O **do**
          Create $ReadProperty$ request for P
          Send request
          if Response if $ReadProperty$ response **then**
            Print property value
            $property\_counter$ += 1
            Pass
          else
            Pass
          end if
        end for
        $object\_counter$ += 1
      end if
      if Object O is a Vendor-defined Object **then**
        $object\_counter$ += 1
      end if
    end for
  else
    Pass
  end if
end for
Print $object\_counter$, $property\_counter$
return

property from the `Device` object type. The IP address indicates the target device. The response of the `ReadProperty` Request from the target device should contain the value of the `objectList` property. The returned value enumerates each object type associated to the BACnet device. This includes standard objects as well as vendor-defined objects. If the response carries a list of objects, we can also confirm that this IP address is indeed associated to a BACnet device, where every BACnet device has a unique `Device` object.

For each object from the list, we further request the presence of all standard properties that are defined for each type of object. The tool places the list of properties into a queue to be requested sequentially. It then sends a `ReadProperty` request, carrying the IP address, to the target BACnet object type and the property. Some properties may have read and/or write restrictions. If the property being requested has no restrictions, the tool outputs the value of the property. A request to a restricted property will result in the tool outputting that an `readAccessDenied` error occurred and will continue to the next queued property. If the property is not configured for the object, the process will continue to the next property in the queue instead. When all of the properties in the queue for the current object have been requested, the tool repeats these steps for the next object in the `objectList`.

Vendor-defined objects (objects with numeric codes greater than 127) do not have string names in the `objectList`. These objects will instead be listed by their numeric code. Since vendor-defined objects could have potentially any property, we only confirm the existence of the vendor-defined object and do not further list its properties. This is because the `ReadProperty` request uses the string object name as input in the BACpypes library. When using vendor-defined objects numeric code, this library will return an error when attempting to read its properties.

4. Evaluation

In this section, we first present the experiment setup and then present the results of our tool confirming BACnet devices at given IP addresses and discovering associated BACnet objects and properties.

4.1. Experiment Setup

Our BACnet field panel in Figure 2 for experiments is a standalone system at the University of Central Florida (UCF), connected to the campus network, but isolated from UCF's actual building control system. The panel was installed and configured by Siemens as part of collaboration with UCF. It hosts a collection of Siemens Building Automation products and uses imported data obtained from a live building automation system. Figure 2 illustrates an overview of the field panel.

Our BACnet field panel has three controllers. The DXR2.E12P-102B Room Automation Station is a fully programmable device for connecting BACnet devices for HVAC, lighting, and shading (window blinds) applications with an integrated port that supports KNX devices. The DXR2.E10PL-102B is similar to that of the DXR2.E12P-102B, except that it includes an integrated damper actuator for HVAC, lighting, and shading. The PXC100-E.A controller performs control, monitoring, and energy management functions, while operating by itself (standalone) or together with other devices (networked setting). It is a multi-tasking platform for program execution and communication with other BACnet devices while also storing program and database information.

The field panel also includes a QMX3.P74-1WSB Room Operator Unit, connected to the DXR2.E12P-102B, and QMX3.P37B Room Operator Unit, connected to the DXR2.E10PL-102B. The QMX3.P74-1WSB includes sensors for temperature, humidty, and CO2, while the QMX3.P37B is equipped with a temperature sensor. The two Siemens QAM2030 duct point temperature sensors are connected to the two DXR2 room automation stations. The touch room operator unit QMX7.E38 is connected to the DXR2.E12P-102B through Ethernet. The PXC controller and DXR2 devices are preloaded with building data while also using live data from the connected sensors.

To interact with the field panel, we use a Raspberry Pi 3 with 32 GB flash memory running Raspbian OS version 10 Buster. The BACpypes python library implementation for the BACnet protocol uses version 0.13.6. These devices are all located within the same network.

4.2. Discovery Results

To evaluate our automation tool, we use the IP addresses of our field panel devices to test how well we are able to identify objects and properties. We analyze three IP addresses of the three BACnet devices: the PXC controller, the two Room Automation Stations DXR2.E10PL and DXR2.E12P. Table 3 illustrates our results.

TABLE 3. OBJECT DISCOVERY AND PROPERTY IDENTIFICATION

Device	Standard Objects Discovered	Total Unique Standard Objects Discovered	Properties Identified
PXC100-E.A	17	3	148
DXR2.E10PL	429	14	3025
DXR2.E12P	423	16	2957

TABLE 4. OBJECT DISCOVERY AND PROPERTY IDENTIFICATION FOR VENDOR-DEFINED OBJECTS

Device	Vendor-Defined Objects Discovered	Total Unique Vendor-Defined Objects Discovered	Properties Identified
PXC100-E.A	0	0	0
DXR2.E10PL	12	4	0
DXR2.E12P	16	5	0

In scanning the PXC controller, our automation tool is able to correctly discover 17 objects and identify 148 total properties associated to those objects, shown in table 3. All 17 objects found for the PXC controller are standard objects, using 3 unique standard object types, while the associated 148 total properties are standard properties.

The DXR2.E10PL scanning discovered 429 standard objects, using 14 unique object types. For the 429 standard objects, a total of 3025 standard properties are also identified. Vendor-defined objects were also discovered, however their properties could not be identified, giving a possible total number of properties for the DXR2.E10PL to be greater than 3025 properties.

Finally, the automation tool scanning the DXR2.E12P Room Automation System discovered 423 standard objects. We discovered 16 unique standard objects types used to create the 423 standard objects. The 2957 properties identified are standard properties associated only to the standard BACnet objects. Vendor-defined objects were discovered,but no properties identified. With this, the total number of properties belonging to the DXR2.E12P would exceed 2957 total properties. However, the exact number cannot be identified by the automation tool.

Table 4 shows the results of discovering vendor-defined objects within the three BACnet devices. From the PXC controller, no vendor-defined objects were discovered, while 12 and 16 vendor-defined objects were discovered for the DXR2.E10PL and DXR2.E12P, respectively. The automation tool correctly recognizes that there are no vendor-defined objects contained on the PXC controller, according to the `objectList` property. From the 12 vendor-defined objects for the DXR2.E10PL, there were 4 unique vendor-defined objects, where each use a unique numeric object code. Likewise, the DXR2.E12P had 5 unique vendor-defined objects, each also using unique numeric codes.

4.3. Limitations

Although our automation tool can identify a large quantity of properties, it was not able to identify all of them, particularly those of the vendor-defined objects. In the discovering of the vendor-defined objects, the automation tool was not able to identify their associated properties. This is because the vendor-defined objects do not have a predefined set of properties. They could use any possible combination of both standard properties and vendor-defined properties. Current implementation for the automation tool does not allow for searching the entire possible property set for vendor-defined objects. We plan to use natural language processing (NLP) techniques to scan online documentation of interest to identify such properties of the vendor-defined objects.

5. Conclusion

In this paper, we first analyze the BACnet protocol and its use of objects and properties for data transmission. Objects and properties from BACnet devices can respond to requests from other devices outside of a BAS, as long as the request follows the BACnet protocol. Limited access control of BACnet allows the adversary to alter property value and severely damage the entire BAS. As the size of the system grows, it can become difficult to perform security analysis of a BAS. We develop a tool to automate the identification of a BACnet device at a given IP and its objects and properties. We have successfully applied our tool to a live testbed BAS on a university campus.

Acknowledgements

This research was supported in part by US National Science Foundation (NSF) Awards 1931871 and 1915780, and US Department of Energy (DOE) Award DE-EE0009152. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] O. Gasser, Q. Scheitle, C. Denis, N. Schricker, and G. Carle, "Security implications of publicly reachable building automation systems," in *2017 IEEE Security and Privacy Workshops (SPW)*, 2017, pp. 199–204.

[2] M. Peacock, M. N. Johnstone, and C. Valli, "Security issues with bacnet value handling," in *ICISSP*, 2017.

[3] D. G. Holmberg, D. G. Holmberg, and D. L. Evans, "Bacnet wide area network security threat assessment," 2003.

[4] T. Brandstetter and K. Reisinger, "(in)security in building automation how to create dark buildings with light speed," 2017.

[5] "nmap/nmap." [Online]. Available: https://github.com/nmap/nmap

[6] "digitalbond/redpoint." [Online]. Available: https://github.com/digitalbond/Redpoint

[7] Musicmancorley, "musicmancorley/hvacscanner." [Online]. Available: https://github.com/musicmancorley/HVACScanner

[8] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[9] J. Figueiredo and J. Martins, "Energy production system management – renewable energy power supply integration with building automation system," *Energy Conversion and Management*, vol. 51, no. 6, pp. 1120 – 1126, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196890409005263

[10] F. Xiao and C. Fan, "Data mining in building automation system for improving building operational performance," *Energy and Buildings*, vol. 75, pp. 109 – 118, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778814001169

[11] W. Kastner, G. Neugschwandtner, S. Soucek, and H. Newman, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, pp. 1178 – 1203, 07 2005.

[12] "The new standard protocol." [Online]. Available: http://www.bacnet.org/Bibliography/EC-9-97/EC-9-97.html

[13] T. Park and S. Hong, "Experimental case study of a bacnet-based lighting control system," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 2, pp. 322–333, 2009.

[14] B. Swan, "The language of bacnet-objects, properties and services." [Online]. Available: http://www.bacnet.org/Bibliography/ES-7-96/ES-7-96.htm

[15] J. Bender, "Bacpypes." [Online]. Available: https://github.com/JoelBender/bacpypes

[16] ——, "Welcome to bacpypes." [Online]. Available: https://bacpypes.readthedocs.io/en/latest/

[17] "Readproperty service results: Continental control systems." [Online]. Available: https://ctlsys.com/support/readproperty_service_array_results/