PAPER • **OPEN ACCESS**

A test environment for evaluating the cybersecurity of building automation

To cite this article: Reto Marek *et al* 2025 *J. Phys.: Conf. Ser.* **3140** 042020

Journal of Physics: Conference Series **3140** (2025) 042020 doi:10.1088/1742-6596/3140/4/042020

A test environment for evaluating the cybersecurity of building automation

Reto Marek[1], Nicolas Oberli[2], Sebastian Obermeier[1], Raphael Bolfing[1], Daniel Wolfensberger[1], Bernhard Tellenbach[2], Olivier Steiger[1*]

[1] Lucerne University of Applied Sciences and Arts, Lucerne, Switzerland
[2] armasuisse, Thun, Switzerland

* Corresponding author olivier.steiger@hslu.ch

Abstract. The increasing digitisation of building automation and control systems (BACS) has led to a greater demand for robust cybersecurity measures. This paper describes a test environment developed to evaluate the cybersecurity of BACS. It includes various protocols, such as BACnet, KNX and DALI, as well as secure protocols like BACnet/SC. The test environment enables BACS security to be assessed under realistic operating conditions. A hackathon conducted in this environment identified critical vulnerabilities, including command injection, hardcoded credentials and denial-of-service risks. These findings emphasise the urgent need for enhanced security measures in BACS installations and highlight the importance of collaboration between IT and OT teams to ensure secure deployments.

1. Introduction

In today's digital world, cybersecurity is essential for protecting sensitive data, ensuring privacy, and preventing financial and operational disruption. Strong cybersecurity measures mitigate these risks by ensuring the integrity, availability and confidentiality of systems and information. This is particularly important for operational technology (OT) systems, which control critical processes in sectors such as energy, manufacturing, housing and transportation. A cyberattack on an OT system can have catastrophic consequences, including physical damage to equipment, disruption to essential services and threats to human safety. Unlike IT systems, OT environments often comprise legacy equipment with limited security capabilities, making them attractive targets for cybercriminals.

This paper focuses on building automation and control systems (BACS), a specific type of OT system. BACS play a critical role in managing various building functions, including heating, ventilation, air conditioning (HVAC), lighting, and security. The growing dependence on digital communications within BACS has heightened the demand for cybersecurity measures. Integrating communication protocols such as BACnet, BACnet Secure Connect (BACnet/SC), KNX and KNX Secure, as well as Digital Addressable Lighting Interface (DALI), adds to the complexity of securing these systems, as each protocol introduces its own vulnerabilities and security features. A successful cyberattack on BACS could result in the disruption of critical building services, cause energy inefficiencies, and enable attackers to manipulate physical security systems and access personal information.

The cybersecurity of BACS has been discussed before. However, no literature has been dedicated specifically to secure BACS protocols, such as BACnet/SC and KNX Secure. In their early work on the subject, Peacock and Johnstone provided an overview of security measures in IP-based BACS,

outlining key issues and methods for improving the protection of cyber-physical systems against the growing threat of cyber terrorism and hacktivism [1]. A similar review, which also considered serial protocols such as KNX TP and LON, was presented by Mundt and Wickboldt in 2016 [2]. In 2018, Peacock et al. conducted research specific to the cybersecurity of BACnet [3]. They describe two proof-of-concept protocol attacks on a BACnet system. Extensive review articles on the topic have also been published by Goltz in 2021 [4] and Li et al. in 2023 [5].

The research reported in this paper addresses two challenges: (1) How to create a realistic cybersecurity test environment for building automation systems. Such an environment is essential for conducting tests, learning about different security threats, and developing effective defences. (2) How to effectively identify the most critical security risks of common BACS solutions.

2. Cybersecurity test environment for BACS

Figure 1 shows our solution to the first challenge: a test environment in which the security of building automation and control systems (BACS) can be evaluated under realistic operating conditions. Unlike real-world BACS environments, the test environment prioritises diversity of manufacturers and communication protocols over typical setups comprising a large number of controllers from one manufacturer. This design enables us to conduct the following research: (1) analyse the behaviour of individual devices and communication telegrams, (2) analyse the compatibility and interoperability of security features and behaviour, and (3) devise test strategies that work across different manufacturers. However, the environment cannot be used to study large-scale system interactions involving hundreds of similar controllers. It contains 27 BACnet devices, including controllers, routers, gateways, room units, actuators and sensors.

Figure 1. Cybersecurity test environment for building automation, installed at Lucerne University of Applied Sciences and Arts.

2.1. Architecture

The architecture of the test environment is based on a modular, scalable network topology (Figure 2). Each ‘site’ within the test environment represents a complete building zone and includes controllers for an air handling unit, a heat generation unit (e.g. a heat pump), heating and cooling distribution groups, and room-level automation devices for lighting, shading, and thermal control. Individual sites are isolated via dedicated VLANs, each with its own BACnet port.

All sites are interconnected via a centralized backbone that hosts key network infrastructure components, including a managed firewall, a Layer 2 switch, and a server environment running multiple virtual machines. These virtual machines run various software components such as engineering tools, building management systems, and building system simulations. The lab setup also includes two management stations: a local Siemens Desigo CC system for site-specific operation, and a centralized Siemens WinCC OA platform for overarching management and coordination across all test sites.

Communication between the sites and with the building management stations is enabled by BACnet routers that serve as single points of entry for their respective VLANs. One of the sites is configured with BACnet Secure Connect (BACnet/SC) to test advanced secure communication methods, while all other sites use traditional BACnet/IP. This architecture allows the test lab to emulate multiple sites with realistic interaction between distributed automation systems and centralized management platforms.

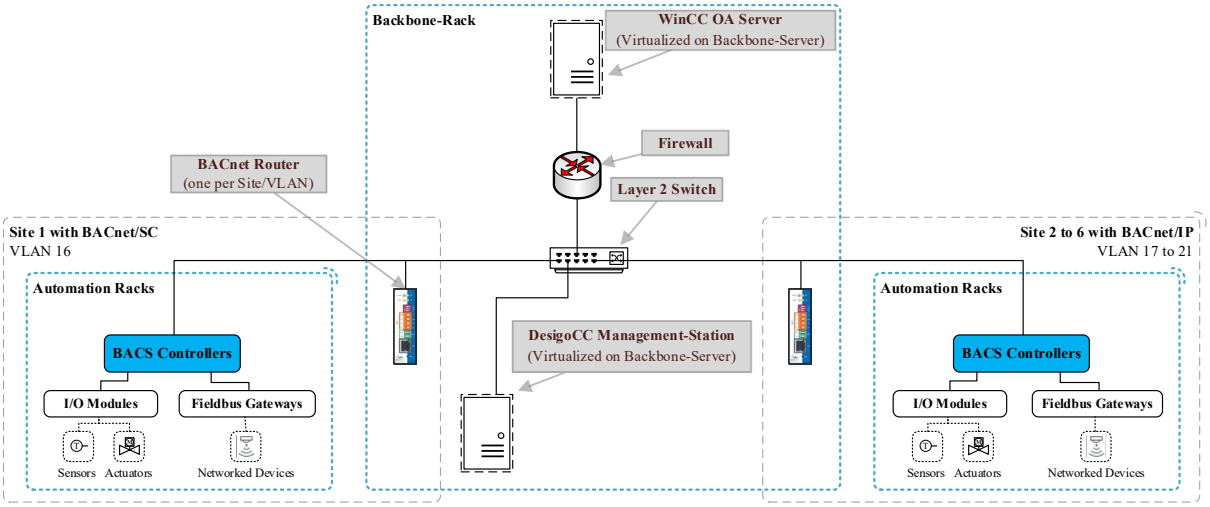

Figure 2. Network architecture of the cybersecurity test environment.

2.2. Test Racks

Each site consists of one or more test racks to which controllers and field devices are connected. Room automation communicates with primary systems and exchanges heating or cooling requests. The implemented sites support a wide range of common field protocols, including BACnet/IP, BACnet/SC, KNX, DALI, Modbus TCP/IP, RTU, M-Bus and LoRaWAN. This enables extensive multi-protocol testing. All automation controllers and field devices are mounted on mobile aluminium frame racks to facilitate transporting and reconfiguring test setups (Figure 1).

2.3. Simulation of building systems

A comprehensive simulation framework has been implemented to generate realistic network traffic, enable dynamic response testing, and trigger alarm states. The framework emulates the behaviour of building systems by intercepting controller outputs (such as actuator positions) and feeding them into a physics-based simulation engine.

The thermal behaviour of the building is simulated using a Python-based model, with the building acting as the controlled system. This model uses basic numerical integration to approximate the dynamic evolution of temperatures over time. To compute the resulting temperatures, the model incorporates actuator inputs such as valve positions, fan speeds, pump states, and heating/cooling temperatures, as well as disturbances such as adjacent room temperatures, outdoor temperature, window contact states, and blind positions. Although the model is intentionally kept simple, it could be further refined in future. However, a detailed physical representation is not required for the current purpose of generating network traffic based on dynamic conditions.

Input signals for the simulation (e.g. sensor signals) are captured by ESP-based microcontrollers using analogue-to-digital converters (ADCs) and general-purpose input/output (GPIO) ports, and are then transmitted over Wi-Fi to the simulation engine running on the AppDaemon add-on for the Home Assistant automation platform. The simulation engine calculates the corresponding physical responses and sends them back to the ESP devices. These devices use DACs and relays to convert the digital values into analogue signals. These signals are hardwired to the relevant inputs on the BACS controllers.

2.4. Backbone

The backbone serves as the central coordination and communication hub. It incorporates a managed firewall to ensure secure network segmentation and routing, a managed Layer 2 switch to host different VLANs, and a server cluster to host the entire software stack, including engineering environments, building management systems and the simulation engine. Additionally, a network-attached storage (NAS) device provides centralised storage for backups and system restores, and Wi-Fi access points facilitate wireless communication with simulation devices and the internet. Each site connects to the backbone via a BACnet router to ensure controlled data exchange and consistent BACnet traffic patterns. The backbone infrastructure is mounted on a separate mobile rack. This rack contains the core network components, servers, NAS storage and Wi-Fi access points.

2.5. Components/Hardware

Table 1 provides an overview of the main components and hardware used in the test environment. It highlights the key devices involved in communication and control between the various test sites and the backbone infrastructure.

Table 1. Overview of the main components/hardware used in the test environment, excluding field devices and IO-modules

Location	**Manufacturer/Devices**
Backbone	MBS BACnet Router UBR-01 Mk II (6x) Fortinet Firewall FortiGate 60F Cisco Switch Catalyst WS-C2960X-48FPD-L QNAP NAS TS-432PXU HPE Server ProLiant DL20 Gen 10 Plus
Site 1 (BACnet/SC)	Siemens Desigo Controller PXC7.E400L Siemens Desigo Controller PXC3.E72A-200A Siemens Desigo Controller DXR2.E18-101A
Site 2 (BACnet/IP)	Siemens Desigo Controller PXC4.E16S Siemens Desigo Controller PXC3.E72A-200A Siemens Desigo Controller DXR2.E18-101A MBS Gateway UGW Double-X RS485 for Modbus TCP/IP Integration CTA/Carel cPCO MINI Enhanced Controller for Heatpump (TCP/IP) MBS Gateway UGW Single-X RS485 for Modbus RTU Integration MBS Gateway UGW Double-X RS485-M-Bus for M-Bus Integration
Site 3 (BACnet/IP)	SBC Controller PCD3.M5360 SBC Controller PCD1.M2220-C15 (2x)
Site 4 (BACnet/IP)	Honeywell Optimizer Advanced Controller N-ADV-112-H Honeywell Unitary Controller UN-RS0844ES24NM (2x)
Site 5 (BACnet/IP)	SBC QronoX Controller PCD3.M6893 Honeywell Unitary Controller UN-RS0844ES24NM (2x)
Site 6 (BACnet/IP)	Avelon Controller/Gateway Beetle T6, 4G WAGO Controller PFC200 (2x)

3. Identification of security risks

To effectively identify the most critical security risks of common BACS solutions, the following strategies were used:

- **Network-based attacks.** The test environment relies primarily on IP-based communications, so network-based attacks must be extensively tested. These include man-in-the-middle (MITM) attacks, where an attacker intercepts and manipulates data between controllers and sensors, and denial-of-service (DoS) attacks, which flood networks with abnormal traffic, causing disruption. In addition, packet sniffing can expose unencrypted communications, allowing attackers to extract sensitive control data.
- **Firmware analysis and physical attacks.** Firmware has been analyzed to find weaknesses in the implementation. This was done either by using publicly available firmware images or by attempting to extract the firmware from the device.

3.1. BacNet fuzzer

To better understand and mitigate the risks associated with network-based attacks, a simple BACnet fuzzer was developed using Python and the Scapy library. A fuzzer is a cybersecurity tool that systematically generates and sends malformed or unexpected protocol data to a system to identify vulnerabilities. In the context of BACnet, the fuzzer helps to uncover weaknesses in device implementations, such as improper input validation, buffer overflows, or unexpected crashes.

Code Snippet 1 shows the Python implementation of the BACnet Fuzzer. It generates and selectively sends fuzzed packets based on a valid Who-Is request, allowing testing of BACnet devices. The BACnet fuzzer works in an iterative loop, sending a series of packets with a delay between transmissions. This minimizes network congestion while allowing time to observe device responses. By selectively mutating BACnet fields rather than randomizing the entire packet, the fuzzer increases the likelihood of detecting protocol parsing problems, improper input validation, or other security vulnerabilities in BACnet implementations.

A valid BACnet Who-Is Request consists of BVLC, NPDU and APDU field values as shown in Code snippet 2. This is the basis for the BACnet Fuzzer implementation. While this BACnet fuzzer is a basic implementation and has been successfully used in the lab, it can be further enhanced by adding additional BACnet data types or by incorporating automated response analysis.

```
from scapy.all import IP, UDP, Raw, send
import random
import time

# Generates selectively fuzzed BACnet packets based on BACnet Who-Is-Request
def generate_fuzzed_packet():
    ip_layer = IP(dst="192.168.1.255")
    udp_layer = UDP(dport=47808, sport=random.randint(1024, 65535))

    # BVLC: Type and function are fixed, length is fuzzed
    bvlc = bytearray([0x81, 0x0B] + random.choices(range(0x00, 0xFF + 1), k=2))

    # NPDU: Not fuzzed in this example
    npdu = bytes([0x01, 0x20, 0xff, 0xff, 0x00, 0xff])

    # APDU: PDU type and Service Choice are fuzzed
    apdu = bytearray(random.choices(range(0x00, 0xFF + 1), k=2))

    raw_bacnet = Raw(load=bytes(bvlc) + npdu + bytes(apdu))
    packet = ip_layer / udp_layer / raw_bacnet
    return packet

# Send multiple selectively fuzzed BACnet packets
def fuzz_bacnet(iterations=100, delay=1, iface="eth0"):
    print(f"Starting BACnet packet fuzzing ({iterations} iterations)...")
    for i in range(iterations):
```

```
        packet = generate_fuzzed_packet()
        print(f"[+] Sending fuzzed packet {i+1}")
        send(packet, verbose=False, iface=iface)
        time.sleep(delay)
    print("Fuzzing completed.")

# Example usage: Fuzz 100 packets with 0.5 seconds delay on interface eth0
if __name__ == "__main__":
    fuzz_bacnet(iterations=100, delay=0.5, iface="eth0")
```

Figure 3. Code snippet 1. BACnet fuzzer in Python.

```
bvlc = bytes([0x81, 0x0B, 0x00, 0x0C])
npdu = bytes([0x01, 0x20, 0xff, 0xff, 0x00, 0xff])
apdu_whois = bytes([0x10, 0x08])
```

Figure 4. Code snippet 2. BVLC, NPDU and APDU field values for a BACnet Who-Is request.

4. Results

To find and exploit security vulnerabilities using the above test environment and attack strategies, a hackathon was held with cybersecurity experts in February 2025. Several vulnerabilities were found during this event. All of them have been disclosed to the affected vendors and are currently being fixed. However, as some devices are still vulnerable at the time of writing, only the type of vulnerability is presented here. The vulnerabilities are described by their corresponding CWE identifiers. The Common Weakness Enumeration (CWE) standard is a globally recognized system maintained by the MITRE Corporation for classifying software and hardware security vulnerabilities, each with a unique ID and description.

- **CWE-78 – Command injection via web interface.** Attackers can inject system commands through web input fields, potentially gaining full control of the device.
- **CWE-321 – Hard-coded cryptographic keys.** Secret keys hardcoded into the software can be exploited by an attacker to execute unauthorized remote commands.
- **CWE-22 – Path Traversal Vulnerability.** By manipulating file paths, an attacker could access or delete protected files, compromising system integrity.
- **CWE-1191 – Unprotected Hardware Debug Interfaces.** Open debug ports or hidden connectors could be exploited to bypass authentication and gain access to the system.
- **CWE-798 – Hardcoded credentials.** Fixed usernames and passwords embedded in the firmware pose a risk if discovered because they cannot be changed.
- **CWE-1286 – Denial of Service through malformed BACnet packets.** Maliciously crafted BACnet packets can crash or disable devices, resulting in denial of service.

5. Conclusions and next steps

The hackathon was a great way to assess the state of BACS security and kick-start a broader security research initiative. Looking at the current findings, there seems to be a lack of common security controls and mitigations. Because BACS devices are in use for extended periods of time and are part of the infrastructure, the associated risks are even more serious. To make matters worse, there is a trend for manufacturers to push cloud-based solutions that require remote access to the infrastructure to function properly, further expanding the attack surface.

It is also worth noting that BACS installations are now moving to secure protocols, which means they require more and more IT resources throughout their lifecycle, such as firmware, certificate and configuration management. As they are usually installed as a by-product of facilities management, IT is rarely involved in the setup phase, but is usually involved in the deployment phase to provide some resources. This deployment methodology is no longer appropriate and both IT and BACS professionals must now work together to ensure secure installations.

Acknowledgments

This research was made possible in part by a commissioned research project for armasuisse Science and Technology.

References

[1] Matthew Peacock and Michael N. Johnstone, *An analysis of security issues in building automation systems*, 2014. [Online]. Available: https://ro.ecu.edu.au/ism/170/

[2] T. Mundt and P. Wickboldt, "Security in building automation systems - a first analysis," in *2016 International Conference On Cybersecurity And Protection Of Digital Services (Cybersecurity)*, 2016, pp. 1–8.

[3] M. Peacock, M. N. Johnstone, and C. Valli, "An Exploration of Some Security Issues Within the BACnet Protocol," in 2018, pp. 252–272. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-93354-2_12

[4] J. Goltz, "Securing Building Automation Systems," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2021, pp. 1–7.

[5] G. Li *et al.,* "A critical review of cyber-physical security for building automation systems," *Annual Reviews in Control*, vol. 55, pp. 237–254, 2023, doi: 10.1016/j.arcontrol.2023.02.004.