DOI: http://dx.doi.org/10.14236/ewic/ics-csr2014.7

A Practical Attack Against a KNX-based Building Automation System

Alessio Antonini
Politecnico di Milano
DEIB
Via Ponzio 34/5
Milan
IT
http://www.deib.polimi.it/
antonini@elet.polimi.it

Federico Maggi
Politecnico di Milano
DEIB
Via Ponzio 34/5
Milan
IT
http://www.deib.polimi.it
fmaggi@elet.polimi.it

Stefano Zanero
Politecnico di Milano
DEIB
Via Ponzio 34/5
Milan
IT
http://www.deib.polimi.it
zanero@elet.polimi.it

Building automation systems rely heavily on general-purpose computers and communication protocols, which are often affected by security vulnerabilities. In this paper, we first analyze the attack surface of a real building automation system - based on the widely used KNX protocol-connected to a general-purpose IP network. To this end, we analyze the vulnerabilities of KNX-based networks highlighted by previous research work, which, however,did not corroborate their findings with experimental results. To verify the practical exploitability of these vulnerabilities and their potential impact, we implement a full-fledged testbed infrastructure that reproduces the typical deployment of a building automation system. On this testbed, we show the feasibility of a practical attack that leverages and combines the aforementioned vulnerabilities. We show the ease of reverse engineering the vendor-specific components of the KNX protocol. Our attack leverages the IP-to-KNX connectivity to send arbitrary commands which are executed by the actuators. We conclude that the vulnerabilities highlighted by previous work are effectively exploitable in practice, with severe results. Although we use KNX as a target, our work can be generalized to other communication protocols, often characterized by similar issues. Finally, we analyze the countermeasures proposed in previous literature and reveal the limitations that prevent their adoption in practice. We suggest a practical stopgap measure to protect real KNX-based BASs from our attack.

Keywords: cyber-physical systems security, building automation, KNX

1. INTRODUCTION

The development of information technology has deeply changed our lives, as well as the majority of industrial processes. The future smart-green houses will be equipped with modern appliances (e.g., intelligent water, electricity and gas meters) with advanced computational and communication capabilities as in Chung-Ming Tung (2012). This process is driven, among other things, by the imperative need to better manage energy resources, which are limited due to a growth in demand, which is quickly saturating the supply capacity. Therefore, the need to switch to smart digital systems, which would allow planned management of resources in civilian and commercial buildings, is now more evident than ever. For instance, a recent research of Junghoon Lee and Gyung-Leen Park and Sang-Wook Kim and Hye-Jin Kim and Chang Oa Sung (2011) showed that smart power-consumption scheduling can reduce the peak load in individual households and in the system-wide power transmission network.

Following this wave, vendors developed home and building automation technologies that will soon play a central role in our daily activities. As reported by a recent market research of Paul Korzeniowski (2012), installations of building automation system (BAS) are growing significantly.

A BAS is composed of several sensors and actuators that control equipments such as lighting, blinds, shutters, security systems, energy distribution, heating and air conditioning, or metering. In order to transfer control data across different components, a common communication protocol (usually denoted as a “bus”) is needed. The obvious advantage of a BAS is that the behavior of the controlled systems can be easily modified via software reconfiguration, without modifications of the physical system-as it was the case for more traditional systems. On the other hand, this inherent flexibility opens the BAS, and the controlled system, to security weaknesses, which may seriously impact the physical surrounding environment.

Recent researches revealed serious security weaknesses as in Wolfgang Granzer and Wolfgang Kastner (2011); Daniel Lechner and Wolfgang Granzer and Wolfgang Kastner (2008); Salvatore Cavalieri and Giovanni Cutuli (2009); Fritz Praus and Wolfgang Kastner (2009), especially when these are not physically airgapped from other communication systems present inside the building, or from the Internet. Some possible solutions to these vulnerabilities, mainly based on the introduction of cryptographic solutions built with specific hardware as in Daniel Lechner and Wolfgang Granzer and Wolfgang Kastner (2008); Salvatore Cavalieri and Giovanni Cutuli (2009); Fritz Praus and Wolfgang Kastner (2009), have been proposed.

However, these works have a common shortcoming: they lack an experimental evaluation of the actual impact of the described vulnerabilities, and similarly lack the evaluation of feasibility of the proposed solutions.

Our analysis of the literature shows a lack of experimental verification of the actual vulnerability of BAS systems that can highlight the feasibility of an attack and the possible impact, making it also easier to suggest effective countermeasures, not necessarily based on the use of specific hardware solutions.

The purpose of this paper is to comprehensively evaluate the attack surface of a network that uses KNX as its basic protocol. We analyze it as the most representative protocol, but many of our findings are likely to affect other protocols as reported in Wolfgang Granzer and Wolfgang Kastner (2011), which analyzes the similar lack of security mechanisms in BACnet [1], LonTalk [2] and ZigBee [3].

This paper makes the following original contributions:

- We review previous research on the security vulnerabilities and the possible attacks against KNX systems;
- We show the practical feasibility of attacking a KNX system without requiring physical access to any device; we describe our attack, and evaluate it on a testbed BAS which reproduces a real-world scenario;

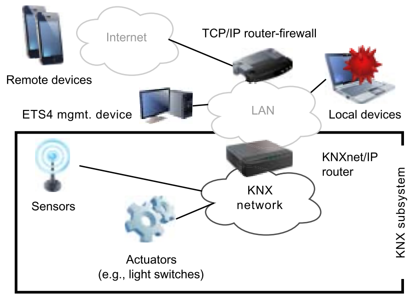


Figure 1: *A typical BAS deployment comprises a local device, a management device, a router connected to the KNX subsystem through the KNXnet/IP router, and remote devices.*

- We analyze existing remediation approaches, describe their limitations and discuss the obstacles that prevent their use. We propose a simple mitigation based on the topology of the system installation.

2. SECURITY OF BUILDING AUTOMATION SYSTEMS

In traditional power systems, operations (e.g., control, actuation) are performed by the means of switches or other physical interfaces. Such switches were limited to basic on/off functions, and additionally, the controlled behavior would depend on the physical wiring of the system. In other words, changing the behavior of a switch or a load would require extensive re-wiring.

A BAS is fundamentally different, being composed of a network of sensors and actuators that communicate with each other or with a central control unit via a bus connection. Actuators perform actions (such as activating or deactivating loads and devices on the physical side of the system, which they are connected to) in response, for instance, to specific inputs on the sensors.

Consequently, a BAS differs from traditional systems in the fact that it decouples its logic from the physical power wires, by adding a communication system (usually denoted as a "bus") for the exchange of information between the various devices. Among other things, this results in the ability to add or remove features to a device acting via software at any time, with no need of rewiring.

So, a BAS is comprised of so-called "smart devices" (i.e. sensors and actuators) and at least a *communication protocol*.

[1] http://www.bacnet.org
[2] http://www.enerlon.com
[3] http://www.zigbee.org

Although several communication protocols exist for the bus (e.g., Modbus [4], LonTalk [5], Zig-Bee, BACnet) KNX [6] is one of the most widely-adopted ones.

KNX is an international standard (ISO/IEC 14543/3), nationally implemented worldwide (e.g., United States AN-SI/ASHRAE 135, Europe CENELEC EN 50090-CEN EN 13321/1, and China GB/Z 20965).

The widespread adoption of KNX is shown by the support of the major manufacturers of BAS devices in the world[7] (e.g., ABB, Siemens and GEWISS).

Currently, many software houses and open-source projects[8] are developing end-user applications for KXN-based devices.

A typical configuration of a BAS connected to a LAN and to the Internet is shown in Figure 1. There can be several computers connected to the LAN, some of them used as management devices. In our case we rely on ETS4 [9], which is a manufacturer-independent configuration tool to design and program KNX hardware (e.g., actuators, sensors).

The KNXnet/IP router connects the TCP/IP and KNX networks.

Clearly, any device connected to the LAN is a potential weak spot for attacking the KNX network.

The spread of KNX-based applications, as well as the variety and the importance of services managed by this system, make it important to assess its potential security weaknesses.

Unfortunately, KNX implements only basic security measures, as previous research has already highlighted.

2.1. State of the Art

Previous work identified lack of authentication and encryption as the two main weaknesses of KNX, and concentrated on proposing solutions that would prevent unauthorized access to the BAS. For instance, in Fritz Praus and Wolfgang Kastner (2009) and Daniel Lechner and Wolfgang Granzer and Wolfgang Kastner (2008) the authors addressed the problem of securing the IP-level connectivity between different KNXnet/IP routers. To do so, they introduced a specific hardware-software solution, because KNX devices were not designed to implement any native cryptography algorithm, and thus they may lack the computational capacity to run standard encryption algorithms. This solution has two drawbacks: the development of nonstandard KNXnet/IP routers and the exchange of data in plain at the KNX network level.

In Salvatore Cavalieri and Giovanni Cutuli (2009), the authors proposed to implement a system based on symmetric and asymmetric cryptography; they suggested to do so without modifying the KNX standard, but by introducing a “controller” which would be delegated to manage the keys. The proposed solution simply moves the problem to the security of the proposed controller.

While we analyze KNX as the most representative protocol, other protocols such as BACnet, LonTalk and ZigBee suffer from similar lacks in security measures as shown in Wolfgang Granzer and Wolfgang Kastner (2011). As claimed in Wolfgang Granzer and Wolfgang Kastner and Fritz Praus (2010), BASs are exposed to several kinds of malicious attacks (e.g., denial of service, network sniffing, man-in-the-middle attacks, code injection, message replay). The risk of attack grows when BASs are not isolated, but rather connected to other networks such as the Internet, as it often happens in real cases.

In summary, we notice that the literature lacks experimental research applied to real-world KNX-based systems. None of the aforementioned researches attempted to carry out attacks on a real testbed, or to practically evaluate countermeasures.

3. MOTIVATING EXPERIMENTS

The above analysis motivated us to implement a realistic KNX network, first to verify and confirm the practical feasibility of the security findings described in the literature, and secondly to systematize such findings in a real-world attack that shows the actual impact of the lack of security mechanisms.

3.1. Experimental Setup

Using GEWISS components, we built a testbed that (partially) reproduces a typical, if minimal, BAS deployment for building automation, as shown in Figure 2. It is composed of the following elements:

Contact Interface The GW90720 supports 4 independent inputs (e.g., buttons, switches, sensors), and can send appropriate commands to actuators.

Power supply The GW90709 is attached to the KNX bus.

[4] http://www.modbus.org
[5] ttp://www.enerlon.com
[6] ttp://www.knx.org
[7] http://www.knx.org/it/knx-partners/knxeib-partners/knx-partners-result
[8] http://ask.aboutknx.com/questions/1217/list-of-knx-open-source-or-free-software
[9] http://www.knx.org/knx-tools/ets-apps/description/

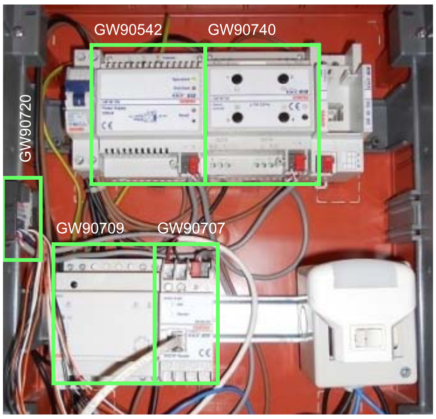


Figure 2: *KNX testbed detail.*

Actuator 1 The GW90740 enables or disables electrical loads through a 16A relay. It has 4 output channels, which have a terminal connected to a switch contact.

Actuator 2 The GW90542 has 2 channels for switching and adjusting light bulbs or other analogical loads (e.g., servomotors).

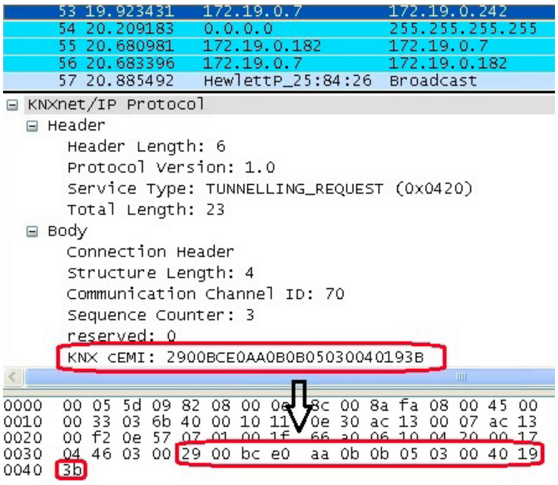
KNXnet/IP router The GW90707 allows the communication between the TCP/IP and KNX network.

We describe the brand and version of the equipment just for the sake of completeness: the findings and attack described in the reminder of this paper are by no means limited to GEWISS components.

To configure our systems, we use ETS4 which is a manufacturer-independent configuration tool to design and program KNX hardware (e.g., actuators, sensors).

3.2. Protocol Reverse Engineering

Although KNX is an open standard, reverse engineering of the semantic of the packets that are sent to the actuators is needed to practically carry out an attack. In our testbed, for instance, the actuators can be programmed via ETS4 only after importing the encrypted configuration files provided by GEWISS. These files contain the specific physical and functional characteristics of the actuator, or the functions that can be carried out (e.g., on-off) and the physical memory address where the software writes EEPROM programming. The overall reverse-engineering approach presented hereby applies to other brands of KNX-based actuators.



```
     53 19.923431   172.19.0.7          172.19.0.242
     54 20.209183   0.0.0.0             255.255.255.255
     55 20.680981   172.19.0.182        172.19.0.7
     56 20.683396   172.19.0.7          172.19.0.182
     57 20.885492   HewlettP_25:84:26   Broadcast
⊟ KNXnet/IP Protocol
  ⊟ Header
      Header Length: 6
      Protocol Version: 1.0
      Service Type: TUNNELLING_REQUEST (0x0420)
      Total Length: 23
  ⊟ Body
      Connection Header
      Structure Length: 4
      Communication Channel ID: 70
      Sequence Counter: 3
      reserved: 0
      KNX cEMI: 2900BCE0AA0B0B05030040193B

0000  00 05 5d 09 82 08 00 0e 8c 00 8a fa 08 00 45 00
0010  00 33 03 6b 40 00 10 11 0e 30 ac 13 00 07 ac 13
0020  00 f2 0e 57 07 01 00 1f 66 a0 06 10 04 20 00 17
0030  04 46 03 00 29 00 bc e0 aa 0b 0b 05 03 00 40 19
0040  3b
```

Figure 3: *Wireshark cEMI data*

Figure 5 shows the encapsulation of a KNX packet into a TCP/IP packet. Specifically, in this example, besides the source and destination addresses, the packet header identifies the actuator's EEPROM area where the "data segment" payload is written when the actuator receives this packet.

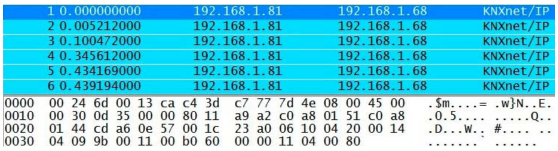
In this example, our actuator had 4 channels, and only the first channel was enabled. We begin by sending a packet from the personal computer to the KNXnet/IP router and capturing the traffic which passes on the LAN. We used Wireshark with the KNXnet/IP plug-in [10] to intercept the TCP/IP packets originating from the ETS4 while sending programming packets to the actuators. We repeat the same procedure for other types of packets (e.g., actuator programming, switching).

As detailed in Figure 3, the KNXnet/IP plug-in does not decode the KNX data segment. More precisely, it does not dissect the encapsulated common external message interface (cEMI) packets as in EIB (1999)), which encode data such as the KNX destination address or the datapoint types (e.g., dimmer setvalues). This immediately confirmed that packets sent and received from the KNX network are not encrypted as shown in Wolfgang Granzer and Wolfgang Kastner (2011). The specific type of router in use, the KNXNet/IP GW90707, seems not to support data encryption (as specified by standard ISO/IEC 14543/3) at all.

Figure 4 shows an example of a sniffing packet related to a programming message of the actuator GW90740.

We manually inspected the intercepted KNX packets and identified the precise location and the semantic of the hex codes that reprogram and execute commands on the actuators. Table 1 shows a sample fragment with the bytes related to channels of the actuator highlighted in bold. The `fe` bytes mean

[10] http://knxnetipdissect.sourceforge.net



```
  1 0.000000000     192.168.1.81       192.168.1.68        KNXnet/IP
  2 0.005212000     192.168.1.81       192.168.1.68        KNXnet/IP
  3 0.100472000     192.168.1.81       192.168.1.68        KNXnet/IP
  4 0.345612000     192.168.1.81       192.168.1.68        KNXnet/IP
  5 0.434169000     192.168.1.81       192.168.1.68        KNXnet/IP
  6 0.439194000     192.168.1.81       192.168.1.68        KNXnet/IP
0000  00 24 6d 00 13 ca c4 3d  c7 77 7d 4e 08 00 45 00   .$m....= .w}N..E.
0010  00 30 0d 35 00 00 80 11  a9 a2 c0 a8 01 51 c0 a8   .0.5.... .....Q..
0020  01 44 cd a6 0e 57 00 1c  23 a0 06 10 04 20 00 14   .D...W.. #.... ..
0030  04 09 9b 00 11 00 b0 60  00 00 11 04 00 80         .......  ......
```

Figure 4: *Sniffing of actuator-reprogramming packages*

that the corresponding channel is disabled, while `01` means that the corresponding channel is enabled.

Using similar techniques, we managed to send arbitrary packets to the other KNX devices in our testbed. As a result, we created a library of basic programming functions, by means of “template” packets. An example is shown in Listing 1, which details a code snippet to enable or disable the first channel of the GW90740 actuator. Our library can be extended by repeating the above procedure on other devices.

3.3. Password Protection

The transmission channel itself is insecure, even during reprogramming, as already mentioned in Section 3.2. This allows an attacker to intercept the password sent in clear by ETS4 during setup. Indeed, as highlighted in Fritz Praus and Wolfgang Kastner (2009), we verified that the only authentication mechanism present in KNX is a 4-bytes password that the installer can send to all actuators using ETS4 during the setup phase. This password is the same for all devices on the same system.

After the setup phase, the system requires no authentication. In fact, any command (e.g., switching a light on or off) is sent without sending any password to the actuators, even if actuators were setup with a password. The password is only required to reprogram the actuators.

Target addr.	Data segment
01c8	2c fe 00 [01] 01 fe 02 fe 03 fe 04 fe
01d4	05 fe 06 fe 07 fe 08 fe 09 fe 0a fe
01e0	0b [fe] 0c fe 0d fe 0e fe 0f fe 10 fe
01ec	11 fe 12 fe 13 fe 14 fe 15 fe 16 [fe]
01f8	17 fe 18 fe 19 fe 1a fe 1b fe 1c fe
0204	1d fe 1e fe 1f fe 20 fe 21 [fe] 22 fe
0210	23 fe 24 fe 25 fe 26 fe 27 fe 28 fe
021c	29 fe 2a fe 2b

Table 1: *Example of KNX packet analysis: 8 fragments of data segments extracted from 8 distinct KNX packets, each representing a specific programming instruction (in this example, the highlighted bytes indicate the 4 channels of the actuator). As a result, the actuator’s EEPROM will be written from byte `01c8` to `021c` (+4 bytes).*

At a first glance, an attacker could obtain the 4-byte password with brute force. A closer look reveals that in a real-world deployment this is more complex than it appears.

We evaluated the performance of a brute-force attack to obtain the 4-byte password. To this end, we implemented a simple random password generator in C#, which forges an authorization packet through the *A_Authorize_Request(key)* packet, as detailed in Section 4.2. We measured that actuators take about one second to respond about the validity of a randomly-generated password. This is due to KNX’s inherent low speed, limited to 9600bps.

It is realistic to assume that an attack in real time on a single actuator/installation is not feasible.

4. A PRACTICAL ATTACK

The results of our experiments corroborate and strengthen the findings detailed in Section 2. This motivated us to systematize such findings in a real-world attack that demonstrates the actual inefficacy of current security mechanisms and, most importantly, to stimulate the research community and industry toward the development of better and usable protection mechanisms.

4.1. Attacker Model and Limitations

We show the feasibility of a systematic attack, against a specific type of devices in the KNX network, that requires no physical access to the BAS.

The attack is implemented by delivering an executable payload to any machine within the KNX system’s LAN. This is well within the capabilities of most attackers, and can be performed through a variety of means (e.g., email attachments, spear phishing, drive-by downloads, or portable USB devices). The current prevalence of malware infections easily shows this. In addition, today’s increased inter-connection between IP and non-IP networks (e.g., KNX) ensures higher chances for our attacker model to fit real-world scenarios.

```
// packet with the variable parameters ‘‘addrDevice’’ (address
of the actuator) and ‘‘ch1OnOff’’ (actuator channel).

byte[]b46 = {(byte)0x11, (byte)0x00, (byte)0xb0, (byte)0x60,
             (byte)0x00, (byte)0x00, (byte)0x11, addrDevice,
             (byte)0x0f, (byte)0x42, (byte)0x8c, (byte)0x01,
             (byte)0xc8, (byte)0x2c, (byte)0xfe, (byte)0x00,
             ch1OnOff,  (byte)0x01, (byte)0xfe, (byte)0x02,
             (byte)0xfe, (byte)0x03, (byte)0xfe, (byte)0x04,
             (byte)0xfe} ;
```

Listing 1: hex code library example of actuator channel programming

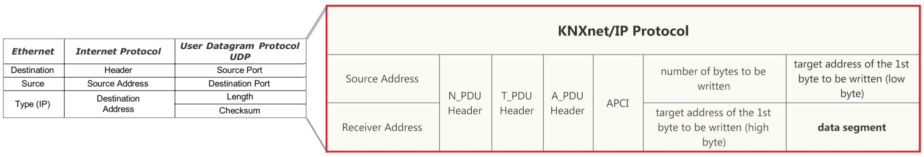


Figure 5: *KNX packet encapsuled into a TCP/IP packet.*

The attacker also needs to have access to at least one device of the same type of each of the attacked devices in order to perform the protocol reverse engineering described in Section 3.2. This is not a huge limitation in practice, as this is a targeted attack and not a generic, self propagating malware. Obtaining a test device is trivial, as these are commercially available. Also, information on the type of devices connected to a router can be easily obtained as we show in Section 3.2.

The attacker goal is to be able to reset an arbitrary actuator to its default settings, take control of it, and in general disable the KNX-managed systems. For all practical purposes, the final objective is to make the end user(s) unable to control any BAS functions until a complete reprogramming of the system takes place.

4.2. Attack Description

We implemented our attack in a proof-of-concept malware written in C#, on top of the Calimero Java library [11] to connect to KNX systems.

In order demonstrate the feasibility of a systematic attack against a specific type of devices in the KNX network without having physical access to the BAS we devised a specific attack scenario. In our attack, the attacker is able to reset to default operating condition each installed actuator; the final result is to turn on all lighting (or in general, any KNX- managed load present in the building). In practice, the end user is no longer able to control these functions until a complete reprogramming of the system takes place. The pre-requirement of our scenario is that the attacker can deliver an executable payload to any machine on the same LAN as the KNX system. This can happen through any common vector, such as e-mail attachments, spear phishing, drive-by downloads, or infection of portable USB devices. The pattern of attack, as shown in Figure 6, exploits the main vulnerabilities of KNX system, i.e., lack of authentication and lack of encryption in the transmission channel, in order to perform a combination of denial of service and man-in-the-middle attack. The workflow of the attack can be summarized as follows:

- the initial compromise of a machine on the LAN happens through any vector, and a malware is deployed on the machine;
- if the KNX system has not been programmed with a password (see Section 3.3), the malware reprograms all of the actuators, deactivating the controlled systems and setting a password, which effectively prevents the installer to reprogram the system, forcing it to be dismantled and sent back to the factory for resetting;
- if the system is protected by a password:
  - the malware simulates a system malfunction, e.g. by turning on and off the lights (since, as claimed in Wolfgang Granzer and Wolfgang Kastner and Fritz Praus (2010), operations are not authenticated);
  - as a result of the malfunction, the system will be re-programmed using the password;
  - as the password is sent in clear text, the malware can intercept it over the wire;
  - after locating the password, the attack proceeds as if the system were not protected.

KNXnet/IP Router Lookup. First, the malware creates a list of candidate KNXnet/IP router addresses by probing the whole local subnet for hosts that respond to `KNXNetworkLinkIP` messages. The malware then invokes the `KNXDeviceScanning` method of the *Discovery* class, which identifies the devices connected to the KNX bus side of that router.

Password Protection Check. The malware sends a packet as shown in Listing 2 by casting the byte format request to cEMI. At the same time, the malware starts a sniffing process to intercept the responses of the device. The `A_Authorize_Request(key)` contains the default password `ffffffff`, which is used if the system is unprotected.

An `A_Authorize_Response (key,level)` is send back from the KNX device containing the level of granted access; if the response contains the hex

[11] http://calimerong.org/

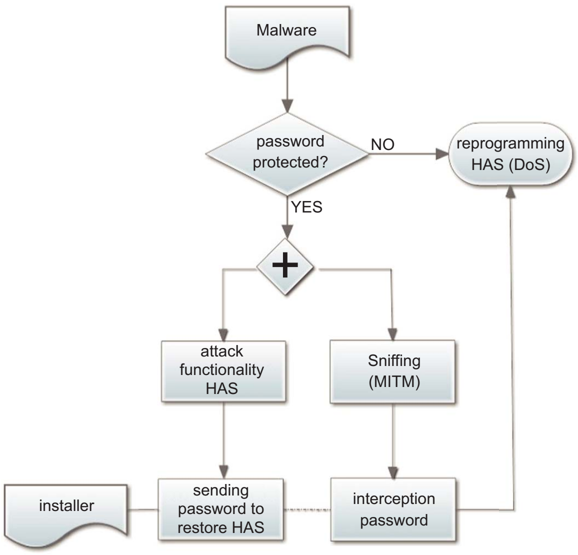


Figure 6: *Attack workflow.*

code `d200` (Listing 3) then the level of access allows to write the memory of the device, otherwise the response contains the hex code `d203`, which means that the password does not allow write access on the memory. The `A_Authorize_Response` is captured by the malware using the sniffing process.

Password Sniffing. We implemented a sniffing module in C# that intercepts TCP/IP packets with KNX payload and reads the APCI field (see Figure 5). Whenever an `A_Authorize_Request` instruction is found, the malware decodes the password from the data segment and attempts a **Password Check** to make sure that it is valid. To entice the user into resetting/reprogramming the system, the malware can simulate a system malfunction (e.g., by turning on and off the loads at random). These operations are not authenticated. Confronted with a malfunctioning system, one of the troubleshooting steps is to reset it using the original password, which is sent in clear text from the ETS4 to the actuators and thus can be sniffed.

Device Reprogramming. After sniffing the password, or if no password is set, the malware can set a randomized password which locks legitimate users out. More importantly, it can reprogram the devices such that, from now on, they will execute arbitrary functions decided by the attacker.

4.3. Attack Evaluation

We deployed the malware in our testbed, and verified that the cycle of operations works both with and without the password being set. As a result, the KNX-controlled equipments can be made unusable or otherwise reprogrammed.

```
//A_Authorize_Request(key)

byte[]b6 = (byte)0x11, (byte)0x00, (byte)0xb0, (byte)0x60,
(byte)0x00, (byte)0x00, (byte)0x11, (byte)0x02,
(byte)0x06, (byte)0x4b, (byte)0xd1, (byte)0x00,
(byte)0xff, (byte)0xff, (byte)0xff, (byte)0xff;
```

Listing 2: `A_Authorize_Request(key)`

```
KNX body: 040207002900b060110200000247d200
Service Type: Tunnelling_Request
Source Address: 1.1.2
Receiver Address: 0.0.0
TypeTransportLayer: Numbered Data Packet (NDP)
Sequence number: 1
APCI Name: Escape
Extended APCI:
```

Listing 3: Packet analysis of a memory write access granted response.

4.4. Physical attack feasibility

Our attack can be carried out, alternatively, by gaining physical access to the KNX bus, as opposed to gaining logical access to a machine on the LAN. Although this may seem a stronger requirement, it is as simple as detaching a wall switch plate and connecting a KNXnet/IP router to the exposed bus. Multiple KNXNet/IP routers can work independently on the same bus: The actuators and sensors cannot tell legitimate or malicious routers apart. If a password is set, it can be bypassed as explained above. Moreover, even if a password is set, malicious instructions can be injected onto the bus. This can be easily automated by incorporating the routines of our attack in a custom KNXnet/IP router.

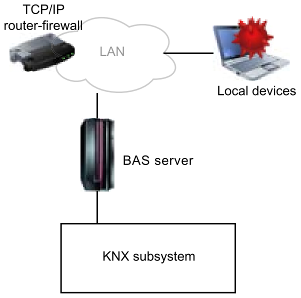


Figure 7: *Stopgap deployment example.*

5. COUNTERMEASURES

Existing work proposed to secure the IP side of the communication, as in Fritz Praus and Wolfgang Kastner (2009), or to implement a cryptography layer inside the actuators, as in Salvatore Cavalieri and Giovanni Cutuli (2009). These are not always viable solutions as they require substantial redeployment of existing hardware; even for newer hardware, a modification of the protocol would be ill accepted by manufacturers due to increased production costs of components.

A potential stopgap countermeasure against remote attack would be to redesign the deployment topology of an existing IP-enabled KNX network, similarly to typical DMZ approaches as shown in Figure 7. For instance, a hardened (web) application could be exposed to the Internet, e.g. to allow remote control of the system, connected back-to-back to the KNX router which would then be shielded from the local LAN, without any need to tamper with the standard itself. In this condition, a malware on the LAN would be unable to attack the KNX system, without exploiting the server or application, thus increasing the attack cost.

Another measure to increase attack difficulty can be to set an installation password, and then, in case of malfunctions or of any reprogramming, to isolate the KNX network from the LAN, using a dedicated, non-infected management device to reprogram the KNX devices.

6. CONCLUSIONS

We analyzed the attack surface of a real KNX system and showed the actual impact of the lack of authentication and encryption solutions in the protocol. To this end, we implemented a proof-of-concept malware able to remotely attacks a real-world home BAS and disable its functionalities entirely.

We also discussed how, although some solutions can theoretically solve the issues, real-world deployments can realistically only be protected through stopgap measures that do not address the underlying weakness of the system.

Our conclusion is that, unless drastic improvements are made to the protocol and to the devices on the market, KNX cannot be secured against attacks led with physical access to the system, and can only be made marginally resilient to remote, IP-based attacks. This means that the protocol, as of now, is unsuitable to handle security-critical systems such as alarms, and its deployment to handle safety-critical systems such as lighting, power or environmental control should be carefully evaluated, taking into account the aforementioned vulnerabilities, and at very least adopting airgapping or other stopgap measures such as the ones we proposed.

REFERENCES

Cavalieri, S. and Cutuli, G. (2009) Implementing encryption and authentication in KNX using Diffie-Hellman and AES algorithms. In: *IEEE Industrial Electronics Conference*, 2459–2464.

EIB (1999) Volume 2: Guide for development. In: *EIBA handbook series, release 3.0*.

Granzer, W. and Kastner, W. (2011) Security analysis of open building automation systems. In: *Computer safety, reliability, and security*. Berlin, Germany: Springer.

Granzer, W. Kastner, W., and Praus, F. (2010) Security in building automation systems. *IEEE Trans. Ind. Electron.*, 57 (11), 3622–3630.

Korzeniowski, P. (2012) *Commercial building automation products: technologies and global markets*. Wellesley, MA, USA: BCC Research.

Lechner, D., Granzer, W., and Kastner, W. (2008) Security for KNXnet/IP. In: *Konnex Scientific Conference*.

Lee, J. et al. (2011) Power consumption scheduling for peak load reduction in smart grid homes. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*, 584–588.

Praus, F. and Kastner, W. (2009) Securing IP backbones in building automation networks. In: *IEEE International Conference on Industrial Informatics*, 410–415.

Tung, C. (2012) Growing trend of network-based, smart green buildings towards automatic energy-saving performance: A study based on Advantech's energy-saving system. *Int. J. Autom. Smart Technol.*, 2 (2). 2.