KNX security devices

Friedrich Praus*, Wolfgang Granzer+ and Peter Balog*

*University of Applied Sciences Technikum Wien
Department of Embedded Systems
{praus, balog}@technikum-wien.at

+ NETxAutomation Software GmbH
wolfgang.granzer@netxautomation.com

As recently shown in [1], thousands of KNX installations are being connected to the Internet unprotectedly. Fortunately, today security awareness is rising and security mechanisms are available, allowing to secure KNX installations. KNX Data Security (AN158) and KNXnet/IP Security (AN159) secure the building automation network (BAN) by providing secure communication as well as verifying the identity of the involved network nodes (i.e. authentication). This prevents security threats like unauthorized interception (e.g. network sniffing), modification (e.g. man-in-the-middle attacks) and fabrication (e.g. replay attacks). Also, security guidelines are available [2, 3].

Nevertheless, network as well as device attacks exist, that cannot be prevented using such methods. For BANs typical representatives are interruption attacks, which have the objective of making a service or data unavailable. These attacks are also referred to as Denial of Service (DoS) attacks. In order to interrupt the communication in a BAN, the adversary is trying to waste network and system resources to prevent the sensor actuator and controller (SAC) from performing its expected function. DoS attacks are always hard to handle, which is especially true for SACs where nodes are subject to stiff resource limitations. Besides, attacks evolving from programming errors/bugs (i.e. buffer overflows) in software running on KNX devices can still occur, even if security measures are deployed. Such attacks or threats can only be detected or prevented by a system having a global view of exchanged process data [1].

The paper is structured as follows: After a discussion on the security of KNX in the year 2016 in Section 1, the process to provide security is discussed in Section 2. Section 3 presents two device concepts that have the aim to prevent or at least to detect the attack scenarios mentioned above. A so called *secure KNX firewall* acts as an interconnection device which is able to prevent attacks by physically separating two or more different BANs. Process data exchange on the network interfaces is being analyzed and depending on the deployed security policy data telegrams are filtered. It allows to drop physically addressed telegrams, prohibit process data exchange between unknown user applications, and evaluate security attributes allowing e.g. telegrams from specified devices containing specific data. A network based *KNX Intrusion Detection System (IDS)* on the other hand is able to detect abnormal network communication as well as abnormal behavior of devices. After having detected such an abnormal situation, it must be determined whether it has to be considered as an attack. Typical attack scenarios that are detected by such an IDS are process data exchange between user applications that violates a statically defined policy or communication relationship, unauthorized management communication i.e. physical destination addresses, or high bus load.

The configuration of both device types is based on security policy which can be derived from the ETS project. A test environment demonstrating compatibility to standard (existing) KNX installations without influencing the normal operation regarding its performance is also presented.

The main contents of the paper are based on the dissertation of the author [4].

Country	BACnet
US, United States	8989
CA, Canada	2296
FI, Finland	282
AU, Australia	271
ES, Spain	231
FR, France	148
SE, Sweden	138
GB, United Kingdom	131
DE, Germany	118
KR, Korea, Republic of	110
NO, Norway	103
IT, Italy	101
CZ, Czech Republic	98
TW, Taiwan	97
NL, Netherlands	89
NZ, New Zealand	47
HK, Hong Kong	45
JP, Japan	44
AT, Austria	42
CH, Switzerland	39
worldwide	13964

Country	KNX
DE, Germany	627
NL, Netherlands	522
ES, Spain	332
FR, France	244
AT, Austria	220
CH, Switzerland	204
IT, Italy	173
NO, Norway	129
SE, Sweden	120
BE, Belgium	119
IL, Israel	109
PL, Poland	67
GB, United Kingdom	56
GR, Greece	42
CZ, Czech Republic	30
RU, Russian Federation	24
VN, Vietnam	23
TR, Turkey	21
LT, Lithuania	20
PT, Portugal	20
worldwide	3295

Table 1: *Scan 2014* Results (Top 20 Countries)

1 Security of KNX in 2016

In order to research whether and how many Building Automation Systems (BASs) based on KNX are openly connected to the Internet and what security measures are currently implemented, [1] performed a worldwide scan of IPv4 addresses. It has been assumed, that KNX installations are connected to the Internet using KNXnet/IP InterConnection Devices (ICDs). KNX devices directly connected to an Internet Protocol (IP) backbone ("native" KNX IP devices) are not considered, since such devices hardly exist. Besides, Internet Protocol version 4 (IPv4) is used and no distinction between dynamic or static IP address ranges is made. Additionally, BACnet/IP devices have been scanned similarly. The results are shown in Table 1. A total of 17.259 BAS installations has been detected. BACnet is being widely used in the US and Canada whereas KNX is very popular in Europe. The installations ranged from business parks and towers, high schools, shopping plazas, water pollution control stations, fire stations, churches to smart homes with control of private saunas. Security measures have not been implemented, allowing unauthenticated and unauthorized full access.

In 2016 security awareness is rising. KNX Data Security (AN158) and KNXnet/IP Security (AN159) have been released, allowing secure communication and verifying the identity of the involved network nodes (i.e. authentication). They prevent unauthorized interception (e.g. network sniffing), modification (e.g. man-in-the-middle attacks) and fabrication (e.g. replay attacks). Besides, dedicated security guidelines [2, 3] have been published and a website targeting security `http://KNXsecure.knx.org` is available.

Nevertheless, the question arises whether security is enabled in real world installations. Thus, the scan of the year 2014 has been repeated. It lasted from 8th August 2016 to 25th October 2016. A total of 22.081 BAS installations have been discovered. The detailed results are shown in Table 2. An increase of unprotected installations of $\sim$28% is noticeable. Regarding KNX, there even is an increase of about $\sim$87%, for BACnet $\sim$14%.

All of the discovered installations are connected entirely unprotected. But what are the threats to KNX installations, even if security measures would have been deployed?

First, an adversary may attack the network medium to access the exchanged data and thus interfere with the data when they are transmitted (network attacks; cf. Figure 1). According to [5], an adversary may try to intercept,

Country	BACnet
US, United States	10374
CA, Canada	2538
FR, France	307
ES, Spain	293
AU, Australia	274
FI, Finland	244
GB, United Kingdom	190
DE, Germany	158
SE, Sweden	153
IT, Italy	149
PL, Poland	104
TW, Taiwan	96
NZ, New Zealand	81
CZ, Czech Republic	80
NO, Norway	69
KR, Korea, Republic of	49
SK, Slovakia	48
NL, Netherlands	47
AT, Austria	43
HK, Hong Kong	40
worldwide	15918

Country	KNX
DE, Germany	809
NL, Netherlands	733
ES, Spain	665
IT, Italy	513
FR, France	464
CH, Switzerland	405
AT, Austria	312
NO, Norway	283
IL, Israel	233
SE, Sweden	233
BE, Belgium	228
PL, Poland	127
GB, United Kingdom	127
GR, Greece	114
CZ, Czech Republic	113
RU, Russian Federation	92
VN, Vietnam	77
TR, Turkey	46
PT, Portugal	44
HU, Hungary	42
worldwide	6163

Table 2: *Scan 2016* Results (Top 20 Countries)

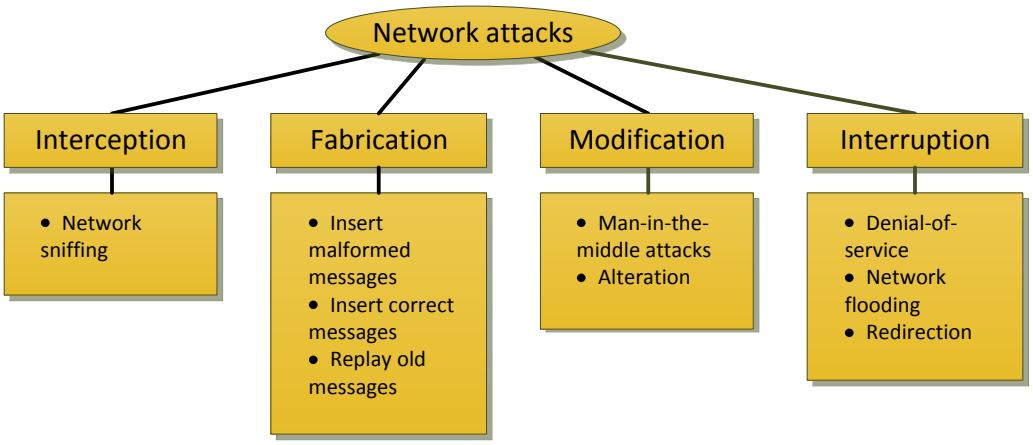

Figure 1: Network Attacks

manipulate, fabricate, or interrupt the transmitted data. Access to the network medium can be achieved in two ways.

- Medium access: The adversary gains physical access to the network medium. This can be accomplished more easily when open communication technologies (e.g. IP, Radio Frequency (RF) or powerline networks) are used.
- Device access: The adversary can use the network interface of another compromised device (e.g. a Web gateway).

Second, an adversary may attack a device to access control functions (device attacks; cf. Figure 2). These attacks can be classified based on the means used to launch them [6, 7].

- Software attacks: An adversary may use regular communication channels to exploit weaknesses in a device's software.
- Side-channel attacks: An adversary may observe external (device) parameters which are measurable during operation to collect information about internals.
- Physical or invasive attacks: An adversary may use physical intrusion or manipulation to interfere with a device.

Interception, fabrication and modification attacks can be prevented by the current KNX security measures. Interruption and and software attacks are covered in the next sections of this paper.

2 Security Process

This paper focuses on the research question, how user applications for smart homes and buildings can be secured. The following hypothesis will be discussed and proven throughout the work:

Today's software for smart home and building devices lacks adequate security mechanisms enabling adversaries to successfully attack those devices. Existing protection techniques from other domains (e.g. the Information Technology (IT) domain) are insufficient and not applicable. Thus, a secure architecture needs to be established. This architecture needs to cover KNX specific constraints, provide a security policy and a secure software environment. Besides, mechanisms for attack detection are needed.

Figure 3 briefly describes the security process and involved stakeholders over the building lifetime. A building owner defines the requirements and instructs a planner to plan the building. The system integrator selects and commissions the devices and an installer deploys them in the building. The building operator finally maintains the building. Basically, security is essential in all steps, for all stakeholders and during the complete lifetime. This paper describes the process of providing secure UAs (blue markings in Figure 3): Starting from secure UA

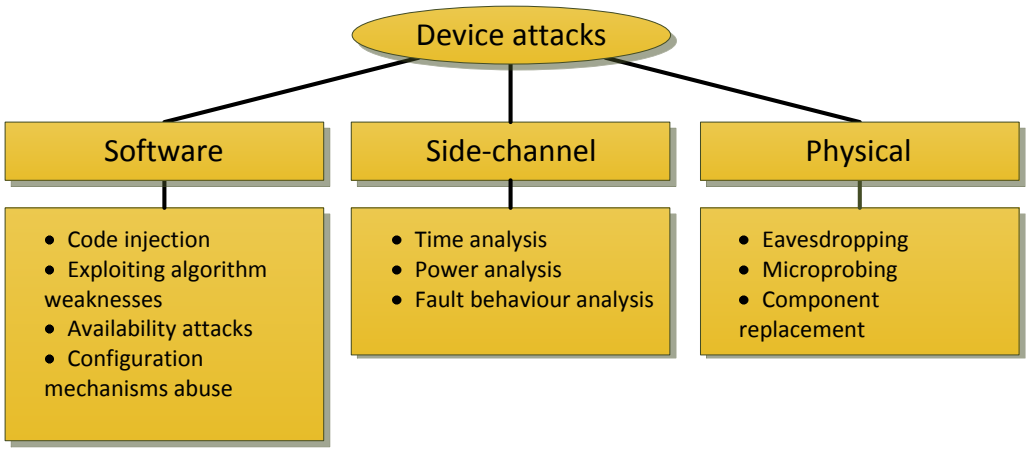

Figure 2: Device Attacks

development, committees are supported in standardizing mechanisms, frameworks and generic policies, while UA manufacturers are supported in creating UAs. System integrators and building operators are supported in adapting security policies during UA operation.

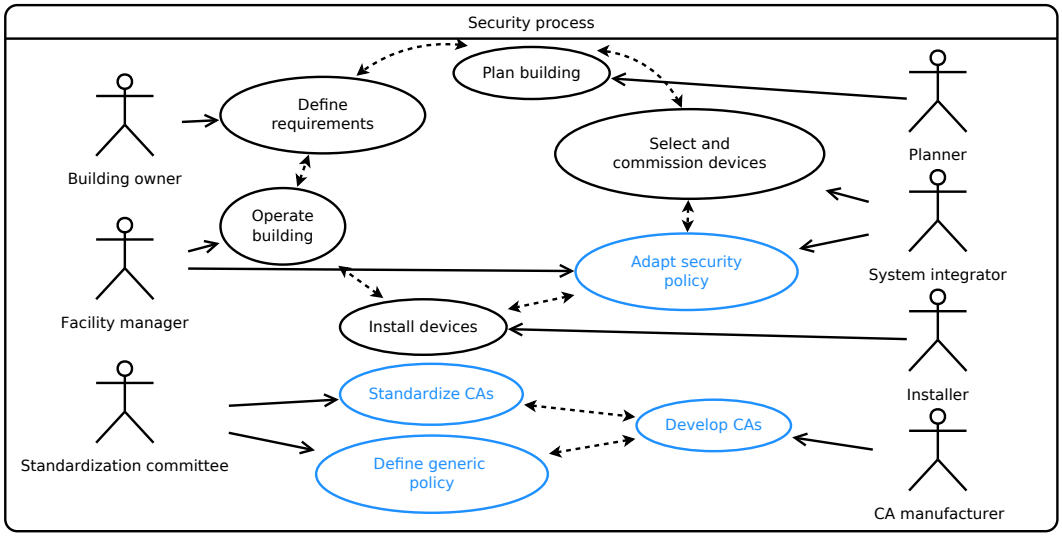

Figure 3: Security Process and Involved Stakeholders

To be able to describe the security concept presented in this paper in a better way, the following use case ⊗ is defined according to VDI 3813-3 and will be used throughout the remaining sections. Figure 4 shows a minimalistic light actuator with delayed on/off behavior being used for a stairwell light.

Use case ⊗ *Stairwell light: When pressing the "on" button* `E` *of the light sensor, the attached light control immediately triggers the light actuator to switch on lamp* `L`*. Upon pressing the "off" button* `E` *of the light sensor, the attached light control waits the time configured with the off delay parameter* `PAR_OFFD` *and then triggers the light actuator to switch off the lamp* `L`*. The output* `L_STA` *of the light actuator is used as feedback to the actuate light sensor, to be able to display the current status using output* `A`*.*

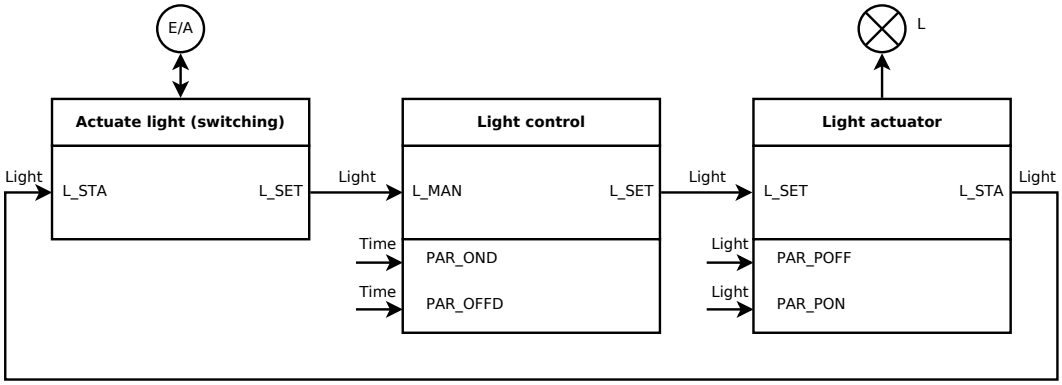

Figure 4: Use Case ⊗: Distributed Control Application: VDI 3813-3: D-1-2 Lighting Control Manual with Time-Controlled Switching Off (Stairwell Light)

There are four requirements for secure KNX installations

1. Generic application model
2. Software security policy
3. Secure software environment
4. Attack prevention and detection

2.1 Generic application model

The basic idea of a generic application model for KNX is to separate generic information from an installation dependent one. An application model thus can be used to abstract an existing KNX installation and represent this particular installation in a generic form. All configuration and management tasks and definition of a security policy can now be performed directly on the abstracted representation.

In this model, the nomenclature is based on the one used in IEC 61499 [8], but modifications were made with respect to the building automation domain vocabulary. It is specified using a formal way accompanied by a textual representation.

Figure 5 shows an example of the model for Use Case ⊗).

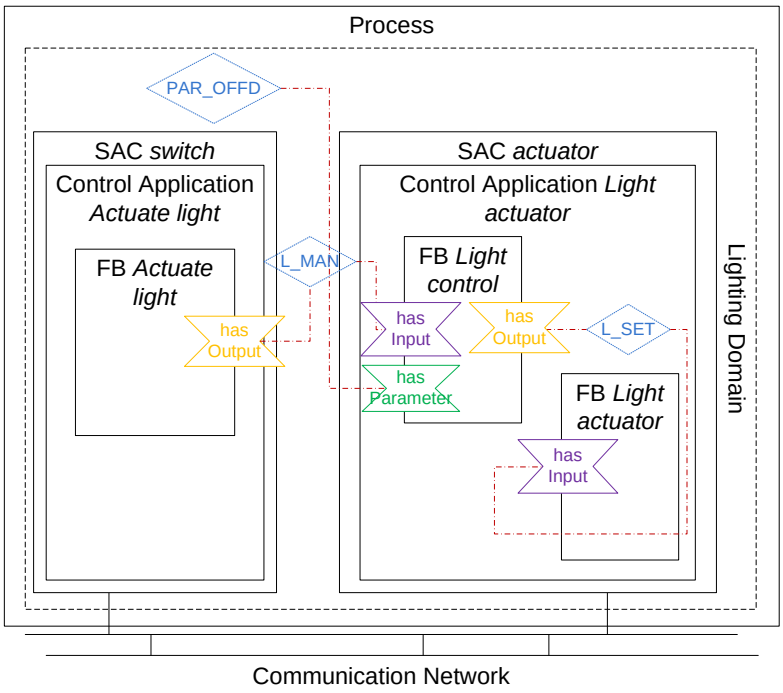

Figure 5: Generic Application Model (Example for Use Case ⊗)

A specification using a formal way and being defined in machine readable form is shown below:

$$
\begin{aligned}
\text{MDP } PAR\_OFFD: &\quad < p, \ldots > \\
\text{PDP } L\_MAN: &\quad < p, \ldots > \\
\text{PDP } L\_SET: &\quad < p, \ldots > \\
\text{FB } \textit{Actuate light}: &\quad \{\text{DP } L\_MAN\} \\
\text{FB } \textit{Light control}: &\quad \{\text{DP } PAR\_OFFD, \text{ DP } L\_MAN, \text{ DP } L\_SET\} \\
\text{FB } \textit{Light actuator}: &\quad \{\text{DP } L\_SET\} \\
\text{CA } \textit{Actuate light}: &\quad \{\text{FB } \textit{Actuate light}\} \\
\text{CA } \textit{Light actuator}: &\quad \{\text{FB } \textit{Light control}, \text{ FB } \textit{Light actuator}\} \\
\text{SAC } \textit{switch}: &\quad \{\text{CA } \textit{Actuate light}\} \\
\text{SAC } \textit{actuator}: &\quad \{\text{CA } \textit{Light actuator}\} \\
LIGHTING\ DOMAIN: &\quad \{\text{SAC } \textit{switch}, \text{ SAC } \textit{actuator}\} \\
PROCESS: &\quad \{\textit{Lighting}\}
\end{aligned}
\tag{D.1}
$$

In fact, such a generic application model is available in the KNX specification, section interworking (function blocks, application domains, . . .).

2.2 Software security policy

The second step to be able to establish adequate countermeasures against the discussed security threats, is to define a security policy. This global policy states, whether the condition of a BAS is security critical and violates some defined constraints or not. For executing this policy it can be split down to involved present values p_i of DPs DP_i, where security requirements for the conditions derived from the policy can be defined, formulated and finally evaluated.

Figure 6 shows an example a software security policy again for use case ⊗. The formal specification is illustrated in Definitions D.2–D.4.

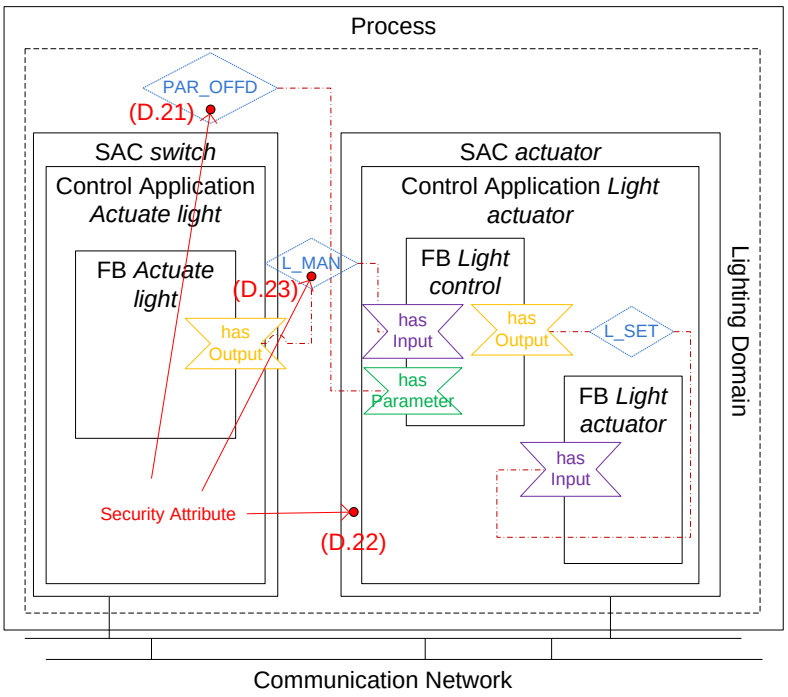

Figure 6: Software Security Policy for Use Case ⊗

$$s_{DP\ PAR\_OFFD} :< p_{PAR\_OFFD}, cond1_{p_{PAR\_OFFD}}, cond2_{p_{PAR\_OFFD}}, AND > \quad \text{(D.2)}$$

$$eval(p_{PAR\_OFFD}) \rightarrow cond1_{p_{PAR\_OFFD}}\ AND\ cond2_{p_{PAR\_OFFD}}$$

$$cond1_{p_{PAR\_OFFD}} : cond(p_{PAR\_OFFD}, maximum, \leq)$$

$$cond2_{p_{PAR\_OFFD}} : cond(p_{PAR\_OFFD}, minimum, \geq)$$

$$s_{SAC\ actuator} :< systemcall\_switchoutput, \quad \text{(D.3)}$$
$$check\_accesstime(systemcall\_switchoutput, maximum\_frequency, \leq) >$$

$$s_{Lighting\ DOMAIN} :< p_{L\_MAN}, cond_{p_{L\_MAN}} > \quad \text{(D.4)}$$

$$cond_{p_{L\_MAN}} : cond(p_{L\_MAN}, \{0, 1\}, =)$$

2.3 Secure software environment

The idea is to separate the system software (including network stacks) running on the device from the UA as well as the node configuration. This model encapsulates the system software entities in a way which is inspired by the object-oriented paradigm. It also makes use of a generic application model, which describes the application behavior and provides relevant configuration and security parameters.

As outlined in Figure 7, the architecture of a secure SAC consists of three major components, each imposing an additional security barrier to the overall security and limiting possible security threats:

- A system software provides controlled access to system resources. It encapsulates hardware specific details, the network protocol stack, the process interface as well as any further system components and offers clean interfaces for the enhanced application layer.
- An enhanced application layer stores the application objects, their DP mappings as well as the security policy for the UA.
- A Sandbox executes the UA in a controlled way and is also designed to support its rapid development. It interfaces the system software via the enhanced application layer and provides a clear abstraction of the underlying hard- and software by providing an object-oriented access (using e.g. the Java programming language). The application designer can thus focus on the application development. This also allows portability of applications between devices offering the same Sandbox (SB).

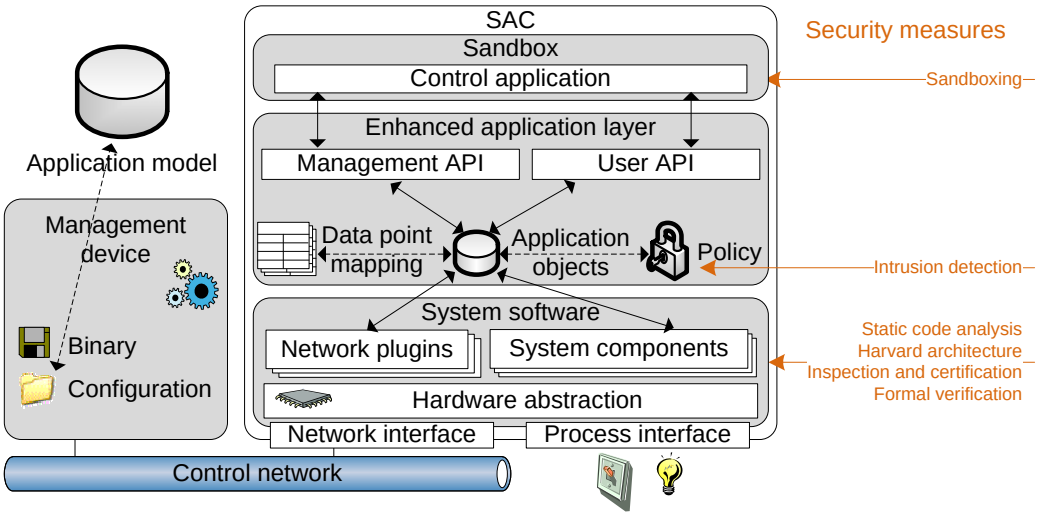

Figure 7: Architecture of Sensor Actuator and Controllers

2.4 Attack prevention and detection

Providing secure devices and preventing software attacks can be partly achieved using the secure CA architecture presented in the sections before. Nevertheless, network as well as device attacks exist, that cannot be prevented using such methods. For Building Automation Networks (BANs) typical representatives are interruption attacks, which have the objective of making a service or data unavailable [5].

Hence, an attack prevention and detection system has to

- prevent attacks using the secure CA architecture and additionally report those violations
- detect attacks and additionally report those violations
- provide a reasonably small detection latency, be operable without influencing the normal operation of a BAS and be update-able with respect to new attack scenarios as well as being flexible with respect to changed system behavior

- be secured with respect to attacks directed towards its operation

3 KNX Security Devices

To evaluate the presented concept and proof its feasibility, KNX security devices demonstrating attack detection and prevention in existing installations will be described. On the basis of use case ⊗, prototypes have been implemented for the different device classes to evaluate and test the stability with respect to memory consumption, performance, and security. A test environment (cf. Figure 8) has been established for SACs, which interact directly with the physical environment, ICDs which link different networks and network segments, and Management Devices (MDs) which are used to configure, maintain and diagnose a BAS. It consists of a KNXnet/IP based backbone with an attached desktop computer running the Engineering Tool Software (ETS) and a network based Intrusion Detection System (IDS) to provide additional security. Using secure ICDs with firewalls, two KNX lines are connected to the backbone. A secure SAC switch and a standard KNX SAC are located on these different lines.

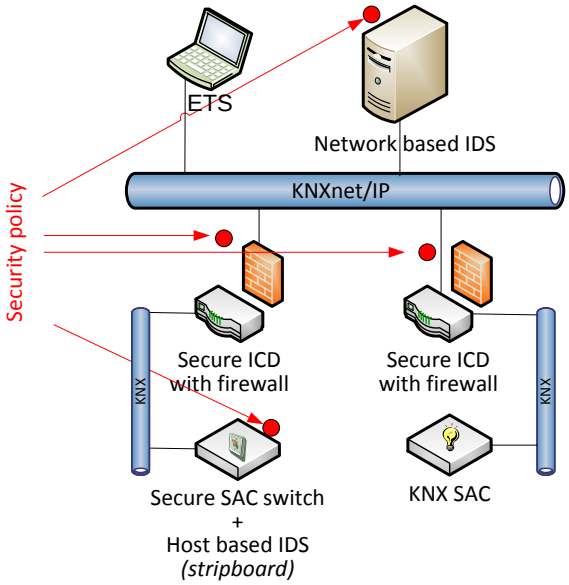

Figure 8: Test Environment

3.1 KNX Firewall

First, a generic firewall architecture is presented, which is then implemented for a KNX based BAS [9]. The developed prototype is able to prevent attacks by physically separating two or more different BANs. Process data exchange on the network interfaces is being analyzed and depending on the deployed security policy data telegrams are filtered.

On the one hand, it is possible to check the communication relationship between UAs – i.e. which device is allowed to talk to which other device. The required information is gained out of the binding information, which has been generated when linking DPs. On the other hand, the data being exchanged – i.e. the present values – are monitored and checked against the security attributes gained out of the security policy.

The left side of Figure 9 shows the typical filter rules and filter chains of a firewall, which are used to implement a security policy. Using the rule `DENY`, telegrams are silently dropped by the firewall. Neither sender nor receiver are informed, that delivery of telegrams has been filtered. Using the rule `REJECT`, telegrams are also dropped with the difference that the sender is informed. The rule `ALLOW` permits telegrams to pass the firewall. Filter chains accumulate filter rules, which are evaluated step by step. The filter chain `INPUT` is used for filtering process data

exchange targeting the application layer of the firewall. The filter chain `OUTPUT` is applied, when the application layer of the firewall transmits a telegram. In fact, the use of the `INPUT` and `OUTPUT` chain is only feasible, if a firewall is implemented directly on a SAC. If an ICD without UA is deployed, only the `FORWARD` chain is required. It is responsible for filtering telegrams being transmitted from one interface to the other. Two approaches are possible, when implementing a firewall policy. The first approach, being more secure but also harder to configure and maintain, is to `DENY` or `REJECT` all telegrams per default and to provide `ALLOW` rules for each telegram, that should be allowed to pass the firewall. Second, the default rule is set to `ALLOW` and `DENY` or `REJECT` rules are specified for specific telegrams that need to be blocked.

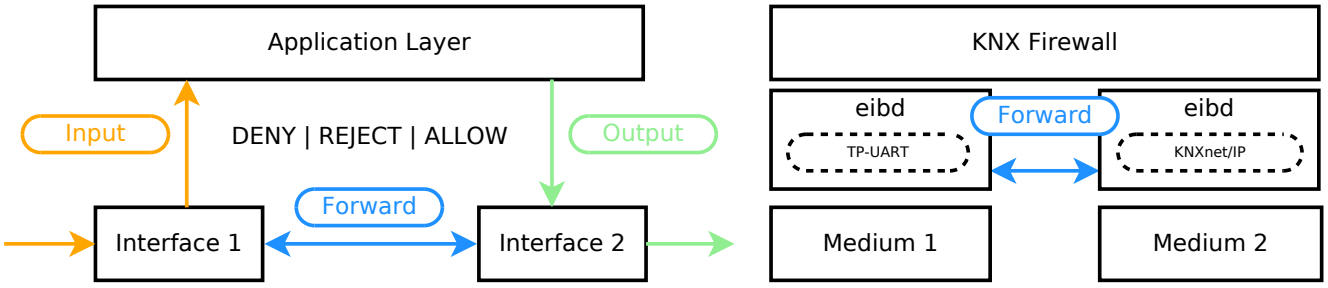

Figure 9: Secure Interconnection Device and Implementation of KNX Firewall

To validate the approach, a KNX based prototype ICD has been implemented for use case ⊗ (cf. right side of Figure 9). It provides connection to two KNX lines using two instances of the eibd[1]. The first interface is connected to TP-1 using a Twisted Pair Universal Asynchronous Transceiver (TP-UART) connection, whereas the second interface is connected to an IP backbone using a KNXnet/IP router. A `FORWARD` chain between those lines is configured, no `INPUT` or `OUTPUT` chain is defined. To generate firewall rules, the following KNX characteristics need to be considered, when formulating `ALLOW`, `DENY` or `REJECT` rules. The communication relationships between UAs can easily be gained out of the installation configuration being available in the ETS. Physical destination addresses are only used for management communication to e.g. (re-) configure devices, read mask version or memory areas. A `REJECT` rule generates a `NACK`, and an `ALLOW` rule generates an `IACK` on TP-1 lines. The Application Protocol Data Unit (APDU) type can be used to further filter process data exchange. The `A_GroupValue_Write`, for instance, is used to set a PDP.

The following security policy can thus be formulated (cf. Listing 1). The syntax is similar to the one used in the standard Linux firewall:

- Drop physically addressed telegrams (cf. Line 3).
- Prohibit process data exchange between unknown UAs (cf. Line 3).
- Evaluate the security attribute $s_{DOMAIN\ Lighting}$ (cf. Line 16):
  - `-a` append a new rule to chain `FORWARD`
  - `-s` source address is always a physical address
  - `-d` destination address can be physical or group address
  - `-i` in-interface
  - `-o` out-interface
  - `-p` APDU type and value
  - `-j` action being applied on rule match

```
1 [global]
2 # deny all telegrams (including physical addressing)
3 standardFirewallRule=DENY
4
5 # configuration for first interface
6 [KNX-IF-1]
7 connectionurl=ip://auto.tuwien.ac.at:6720
8
```

[1] `https://www.auto.tuwien.ac.at/~mkoegler/index.php/eibd`, Last access: 2015/09/01

```
# configuration for second interface
[KNX-IF-2]
connectionurl=ip://auto.tuwien.ac.at:6721

# rules
[rules]
# allow telgrams from secure SAC switch originating from KNX-IF-1 to KNX-IF-2 with target secure SAC
    actuator, present value must be 0 or 1
-a FORWARD -s 1.2.2 -d 0/0/1 -i KNX-IF-1 -o KNX-IF-2 -p A_GroupValue_Write={0,1} -j ALLOW
```

Listing 1: Configuration of KNX Firewall for Use Case ⊗

Implementation of the policy check can be realized using simple `if` statements (cf. Listing 2).

```
// Check every received telegram
static gboolean CheckReceivedTelegram(struct FirewallRuleChain *chain, eibaddr_t source, eibaddr_t
    destination, gchar in_interface, gchar out_interface, APDU *data) {
  // Check source address
  if (CheckSourceAddress(chain->source_from, source)) {
    // Check destination address
    if (CheckDestinationAddress(chain->destination_to, destination)) {
      // Check in and out interface
      if ((chain->in_interface & in_interface) && (chain->out_interface & out_interface)) {
        // Check APDU type and value
        if (CheckAPDU(chain->data, data) {
          // Rule matched, apply jump
          return (chain->jump);
[...]
```

Listing 2: Implementation of KNX Firewall

The secure ICD with firewall has been implemented on an embedded Linux platform. Regarding performance, no reasonable measurements can be performed. In fact, the processing time of the firewall chain and rules is less than 1 ms and thus not measurable using system time. Also, the speed of the TP-1 medium is so low, that even on full bus load no reasonable telegram delay can be measured using this platform.

3.2 Network Based Intrusion Detection System

This section describes a prototype implementation of an IDS. A network based approach has been chosen. It is thus possible to avoid influencing the normal operation of the BAS, since no change of SACs is required, and also no performance penalty on those nodes is generated. A dedicated IDS can also be equipped with more processing and memory resources. Besides, such an IDS can be deployed to existing installations, since only an access to the BAN is required, which can be achieved easily. The following attack detection functionality has been realized in the KNX based prototype implementation of use case ⊗ (based on [10]):

- Alert on process data exchange between UAs, if it either violates a statically defined policy, or additional abnormal communication relationships are detected.
- Alert on management communication i.e. physical destination addresses.
- Alert on high bus load (the threshold needs to be defined a priori).

Figure 10 shows the software structure of the prototype, which has been implemented on the same embedded Linux platform as the secure ICD. The data gathering component relies on one or more instances of eibd. KNX telegrams from various BANs can thus be gathered by the IDS.

The data processing component has been implemented in a hybrid way. A Signature based Intrusion Detection System (SIDS) approach is used to detect the following issues:

- Violation of communication relationships: The policy (i.e. the security attribute $s_{DOMAIN\ Lighting}$) is provided, which defines the normal communication behavior between UAs. Likewise, security attributes are specified, which define forbidden communication behavior. Alerts are then immediately generated, if e.g. a sending address is not allowed to write to a specific receiving address.
- Management communication
- High bus load

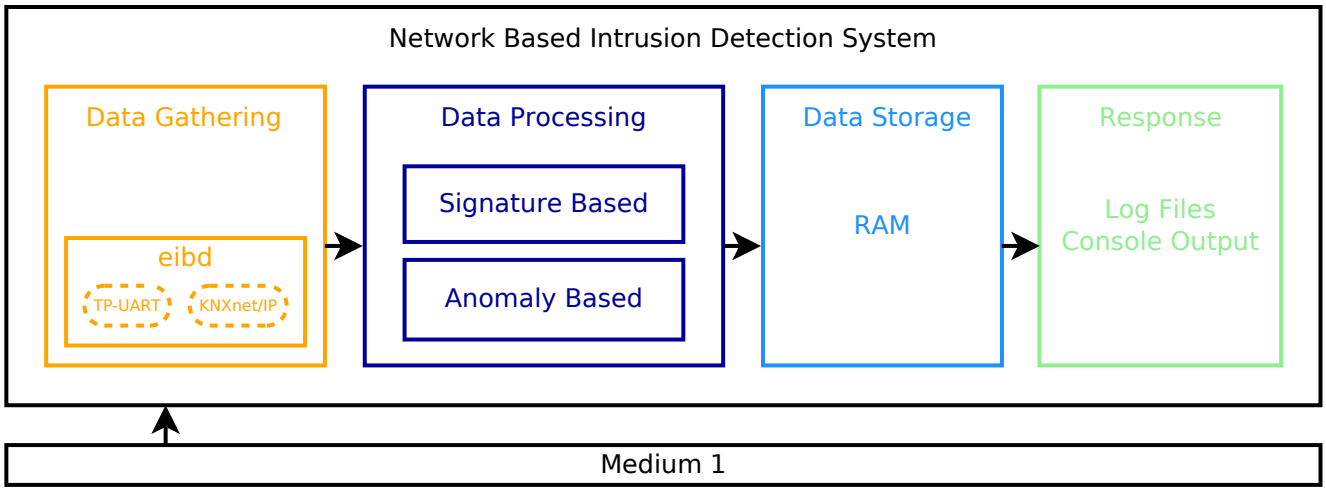

Figure 10: Network Based KNX Intrusion Detection System

Listing 3 shows the configuration containing the static rules for the prototype. Syntax is based on the SNORT IDS and enhanced with KNX specifics. A rule is structured as follows: `<command> <source address> -> <destination address> [APDU] [msg:"<message to display>"]`

Command can either be `alert`, to trigger the response component to e.g. log to a file or `pass` to allow communication. It is followed by the source and destination address, which is a group or physical address in KNX notation or `any` for a wild card. Then an optional specification of the APDU can be provided, to enable evaluating security attributes. Finally, a message can be specified, which is also sent to the response component.

```
[global]
# Configuration for first interface
[KNX-IF-1]
connectionurl=ip://auto.tuwien.ac.at:6720
# Alert on bus load higher than 60%
bus-load-warning=60

# Alarm on management communication i.e. physical destination addresses
alert any -> *.*.* (msg:"Management communication detected")

# Do not alarm on communication from secure SAC switch to secure SAC actuator
pass 1.2.2 -> 0/0/1 A_GroupValue_Write={0,1}
```

Listing 3: Configuration of Network Based Intrusion Detection System for Use Case ⊗

Additionally, an Anomaly based Intrusion Detection System (AIDS) approach is implemented, where the IDS first gathers data from the BAN. The data processing component then uses anomaly based algorithms, to learn the normal communication relationships between UAs and also their behavior regarding telegram rates. Malicious communication between UAs can then be detected and alerted during runtime.

Implementation of the policy check in the data processing component can be realized using simple `if` statements (cf. Listing 4).

```
int checkRule(T_NetworkDataStorage *rule, T_Telegram *telegram) {
  // check source address
  if (checkSourceAddress(rule->source, telegram->source)) {
    // check destination address
    if (checkDestinationAddress(rule->destination, telegram->destination)) {
      // check APDU type and value
      if (checkAPDU(rule->apdu, telegram->data) {
        // rule matched, apply command
        return (rule->command);
[...]
```

Listing 4: Implementation of Intrusion Detection System

The data storage component simply stores relevant data in the memory and does not rely on a database. This allows implementation on simple non-Linux based MicroController Units (MCUs).

The response component is also only implemented partially as a passive system, which simply displays alert messages on the standard output and logs them to files.

Experimental tests with the implementation show similar results regarding performance and memory overhead as the ICD prototype. No reasonably measurements can be performed during normal runtime, and even when stress tests have been executed at maximum bus speed of the TP-1 medium, the CPU load and memory usage are negligible. The requirement to provide a minimal detection latency is fulfilled, since every telegram is analyzed immediately and alerts are generated, if appropriate. Using a network based IDS also ensures, that the standard process data exchange between SACs is not influenced.

4 Conclusion and Future Work

This paper has shown, that despite the currently available security measures in KNX, additional improved KNX security devices are needed. Currently, attacks on KNX based SACs can not be prevented using standard KNX devices. Attacks on KNX networks can likewise only be partly prevented. Standard couplers do not allow context based filtering and also currently do not cover prevention or detection of DoS attacks.

Manufacturers and standardization committees are required to handle these types of attacks in the future. Discussions, whether an extension of the KNX specification is possible to include a security policy or an extension of the ETS can be realized to provide additional security checks need to be carried out.

The question how to deal with already existing installations being connected to the Internet unprotectedly is still open, leaving room for future work.

References

[1] F. Praus and W. Kastner, “Spotting Unsecured KNX Installations,” in *Proc. KNX Scientific Conference*, November 2014, KNX Scientific Award 2014. [Online]. Available: http://www.praus.at/dl.php?url=Spotting_unsecured_KNX_installations-Praus_final.pdf

[2] KNX Association, “KNX Secure Checklist,” Tech. Rep., 2016. [Online]. Available: https://www.knx.org/media/docs/Flyers/KNX-Secure-Checklist/KNX-Secure-Checklist_en.pdf

[3] ——, “KNX Secure Position Paper,” Tech. Rep., 2016. [Online]. Available: https://www.knx.org/media/docs/downloads/Marketing/Flyers/KNX-Secure-Position-Paper/KNX-Secure-Position-Paper_en.pdf

[4] F. Praus, “Secure control applications in smart homes and buildings,” Ph.D. dissertation, Technische Universitaet Wien, November 2015.

[5] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*. Prentice Hall Professional Technical Reference, 2002.

[6] D. Hwang, P. Schaumont, K. Tiri, and I. Verbauwhede, “Securing Embedded Systems,” vol. 4, no. 2, pp. 40–49, mar 2006.

[7] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, “Security in Embedded Systems: Design Challenges,” *Transactions on Embedded Computing Systems*, vol. 3, no. 3, pp. 461–491, 2004. [Online]. Available: http://portal.acm.org/ft_gateway.cfm?id=1015049&type=pdf&coll=GUIDE&dl=GUIDE&CFID=45187622&CFTOKEN=94125376

[8] International Electrotechnical Commission, “Function Blocks – Part 1: Architecture,” IEC 61499-1, dec 2011.

[9] G. Forstner, “Security in Smart Homes - Eine modulare KNX Firewall,” Master’s thesis, AAT-Lab@Department of Embedded Systems, FH Technikum Wien, May 2013.

[10] M. Kreilach, “Ein netzbasiertes Intrusion Detection System für KNX,” Master’s thesis, AAT-Lab@Department of Embedded Systems, FH Technikum Wien, May 2013.

Acronyms

AIDS Anomaly based Intrusion Detection System
APDU Application Protocol Data Unit
BAN Building Automation Network
BAS Building Automation System
CA Control Application
DP Datapoint
ETS Engineering Tool Software
FB Function Block
ICD InterConnection Device
IDS Intrusion Detection System
IP Internet Protocol
IPv4 Internet Protocol version 4
IT Information Technology
MCU MicroController Unit
MD Management Device
MDP Management Datapoint
PDP Physical or Process Datapoint
RF Radio Frequency
SAC Sensor Actuator and Controller
SB Sandbox
SIDS Signature based Intrusion Detection System
TP-UART Twisted Pair Universal Asynchronous Transceiver