International Journal of Computing and Digital Systems

An Intrusion Detection System with Home Installation Networks

[1]Thomas Mundt, [1]Andreas Dähn, and [1]Stephan Sass

[1] *Institute of Computer Science, University of Rostock, Germany*

E-mail address: thomas.mundt@uni-rostock.de, andreas.daehn@uni-rostock.de, stephan.sass@uni-rostock.de

Received 2 Sep 2013, Revised 11 Oct 2013, Accepted 13 Nov 2013, Published 1 Jan 2014

Abstract: Modern buildings are often equipped with universal bus systems. The purpose of these bus systems is to control the functions of houses such as lighting, climate control, and heating. In this paper we present a case study on how to build a home intrusion detection system based on data delivered by a house installation network. For this purpose we use anomaly detection in a similar way as in traditional network observing intrusion detection systems. We also present an example implementation utilizing outlier detection. Finally, we derive further research questions from the example implementation.

Keywords: Home and building automation, Presence detection sensors, Intrusion detection, Outlier detection.

I. Introduction

Intrusion detection systems (IDS) are routinely used in large computer networks. Their main purposes are to alert administrators when intrusive actions are detected and to collect information about attacks that might be recognized after the attack has been performed. IDS listen transparently to network traffic using sensors and determine a variety of measurable parameters, such as requests per second or amount of data per 15 minutes etc.

Installation bus systems are increasingly often used to monitor and control lighting and climate in commercial and private houses. These bus systems are designed for easy integration of controller elements such as light switches, presence sensors, and temperature sensors with lights and climate systems (actors). For this purpose sensor events such as pressing a light button or passing a sensor are signaled over a network. Those sensor events represent the current situation in the building (the current state of the building) quite precisely. We will show that those sensor events can be used to detect intrusive actions inside the building. It seems natural to borrow the ideas of network based IDS to building based IDS.

Without limiting generality we focus on KNX as most often installed bus system for home automation. A KNX system can include many sensors and sensor types. We use available sensor data in three steps: to learn about installed sensors that can potentially provide useful information about intrusion, to learn about the normal behavior inside a building, and to detect anomalies, which might indicate an intrusion.

After this introduction, the paper continues with a brief introduction on intrusion detection systems and home installation networks in the two parts of section II.

Section III describes the concept behind a home intrusion detection system in detail. Basic outlier detection is used for this. An implementation is presented in section IV. Section VI finally concludes the paper and gives an outlook for further work.

II. Background

In this section, we take a look at intrusion detection and home installation networks. We show how ideas can be transferred from network based IDS to building based IDS and which parts of a building based IDS have to be redeveloped.

A. Intrusion detection systems

Intrusion Detection Systems (IDS) are today a part of most network security appliances. In case of an intrusion (which can be defined as “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource” [1]), IDS can alert operators or even try to repel the attack. To identify malicious behavior, IDS utilize a behavior model of the network, which represents it as a feature set. Each feature can describe interval-based values (such as the number of incoming packets within the last second of monitoring), instant values (such as the size of a specific packet) or complex observations (such as system calls). The behavior model is used to detect intrusions as differences between model and real observations. A more detailed introduction to the theoretical basis of IDS can be found in [2].

Typical problems IDS have to deal with are false positives and false negatives. In case of a false positive, normal behavior is misinterpreted as intrusion; in case of a false negative, an intrusion is not detected. As there is no ideal behavior model and false negatives usually come with worse consequences than false positives, IDS tend to produce alerts caused by false positives. Some recent research tries to address this problem using the operator’s feedback on alerts [3].

With emerging new network technologies, IDS were adapted to them (e.g. to the requirements of wireless networks [4][5] or web services [6]). Learning from such adoptions of IDS to new technologies, we try to close the gap between network intrusion and building intrusion.

In the following subsections, we take a look at the methods, which are used to decide whether an observed behavior is an intrusion – or not. The decision is either made using pattern detection or various methods of anomaly detection.

1) Pattern detection: In pattern detection, algorithms compare the observed behavior to well-known sets of intrusion-related behavior. If the comparison result indicates a match for intrusion-related behavior, operators are alerted.

A typical drawback of intrusion detection with pattern detection is the need for (static) a priori knowledge – at least the pattern have to be known before they can be used in testing.

Because of the problems caused by this restriction, most IDS today use statistical approaches to detect anomalies, which will be described in the following subsection.

2) Anomaly and outlier detection: In contrast to pattern detection, the anomaly detection approach relies on statistical evaluation of monitored data. To differentiate intrusive from normal behavior, classification algorithms are used.

The statistical evaluation of observations uses classification algorithms to decide to which of a given number of classes the observed behavior belongs. The used algorithms are mostly from the field of supervised machine learning. The user has to provide training data for the behavior model before the algorithm can be used to differentiate suspicious from harmless behavior.

Roughly and overall, intrusion detection can be seen as application of live outlier detection: Observations caused by normal behavior can be explained with means of the behavior model – everything that cannot be explained is treated as anomaly and triggers an alert.

In recent research, both approaches are combined to hybrid approaches [7], leading to systems that can identify known intrusions by pattern matching and unknown intrusions by statistical evaluation. In contrast to computer networks a building does not change its structure very often. We can further assume that typical sensor events will reoccur in daily and weekly patterns.

B. Home installation networks - KNX as example

For the purpose of explaining home installation networks we chose KNX as expressive example. KNX is a standardized network protocol for controlling the electrical installations in commercial and residential buildings. The KNX standard is maintained by the KNX Association and was approved by several standardization organizations, such as ISO (ISO/IEC14543- 3) and ANSI (ANSI/ASHRAE 135). An overview about the capabilities of KNX can be found in [8] and [9].

KNX supports different medias. Twisted pair wiring is most frequently used. Other medias are powerline, radio, infrared, and Ethernet. Twisted pair cables allow a throughput of 9600bit/s. Media is accessed in a CDMA/CA scheme [10] [11]. The maximum allowed segment length is 1000m (physical) or 4000m (logical with line repeaters). Devices have to be wired without building loops. Exempt this, topology is irrelevant and can be any type of tree.

KNX devices send single telegrams as data carrying objects. The structure of a telegram is shown in Figure 1.

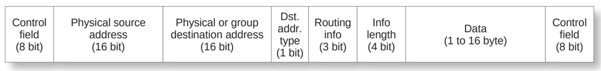

Figure 1. Structure of a KNX data telegram.

Each telegram contains a 16-bit physical address as source address and 16-bit physical or group addresses as destination address. A flag indicates group addresses. Physical addresses consist of area (4 bit), line (4 bit), and device (8 bit). A physical address will be written like 3.2.12 (area 3, line 2, device 12). In most cases a line represents a physical segment of the network.

In some rare cases repeaters couple up to 4 segments. Lines can contain up to 256 devices (64 devices per segment due to the limitation of power supply). Up to 15 lines can be assembled into an area. One additional line (main line) then works as cross-link. Couplers are also counted as devices in each line. A network can have up to 15 areas. Destinations can be addressed by physical addresses or group addresses. Group addresses are structured as main group (4 bit), middle group (3 bit), and sub group (8 bit) and are written as 1/1/74 for example. Devices can be configured to listen to certain group addresses. Telegrams are sent by sensors (such as light switches), and received by actuators (such as relays). The KNX standard defines several data types ranging from simple bits (e.g. light on, light off) to complex structures such as “Atmospheric Pressure with Status and Command”.

Sensor devices are configured to send telegrams to either physical or group addresses. The current configuration can be queried from sensors.

Figure 2 shows a part of a KNX network with physical device addresses and several group members of one specific group.

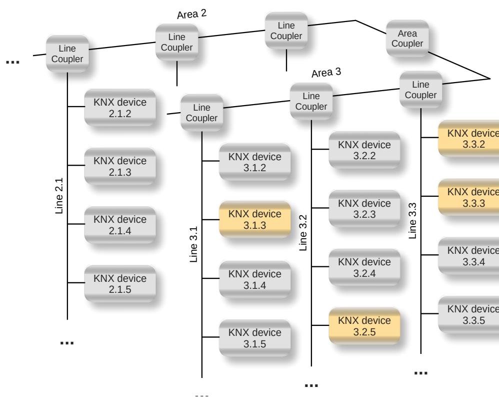

Figure 2. Example for the structure of a KNX network showing two areas with several lines and several devices per line.

In most scenarios line couplers and area couplers filter telegrams according to configured rules in order to limit collision domains and, hence, to reduce network utilization. For the same reason Ethernet is often used as media for main lines between lines and between areas. In this case KNX telegrams are tunneled via the KNXnet/IP [12] protocol. Depending on the filters set in couplers telegrams will be distributed throughout the network and can be sampled at several locations. KNX does neither implement authentication nor privacy measures.

Software libraries are available for all popular programming languages. For our purposes we used Calimero, an open source framework from the University of Vienna [13] [14].

1) Domain knowledge: Although the final implementation should detect intruders without having knowledge about the meaning of sensor events and KNX telegrams, we need this information for evaluating our approach. In order to understand KNX traffic we need to gather additional information. In the terminology of KNX we need to find the position and purpose of sensors. We considered four approaches to collect the needed information:

- Visible inspecting KNX devices and looking for addresses.
- Triggering KNX sensors or waiting for visible events and correlate with sensor readings on the bus.
- Using social engineering to get the building's wiring plan from the electricians that installed the network.
- Automatically learning from KNX traffic about the meaning of telegrams. Active querying the type of KNX device (a function implemented in KNX) would deliver additional information.

Obvious sensors are light switches and switches to control shading blinds. For evaluating our model we needed to find the room number and KNX physical address for those sensors. Other sensors subject to visible inspection are motion detection sensors on the ceiling within the floors.

2) Typical sensors: Typical sensors in a building are selected by necessity for automation. Available sensors can be classified in those manually used (such as switches), those triggered during other tasks (such as presence sensors, sensors which notice open doors and windows), and sensors which provide a stream of values without external triggers, e.g. temperature, wind, or outdoor light intensity sensors. Presence detection sensors are typically built as infrared sensors and mounted on walls or below ceilings so that a person walking by - including intruders - will trigger them - see Figure 3. They can be mounted more or less visibly. Switches are commonly used for manual control or function selection for e.g. air conditioning, lighting, and sunshades. The third sensor category is of less interest regarding home installation IDS.

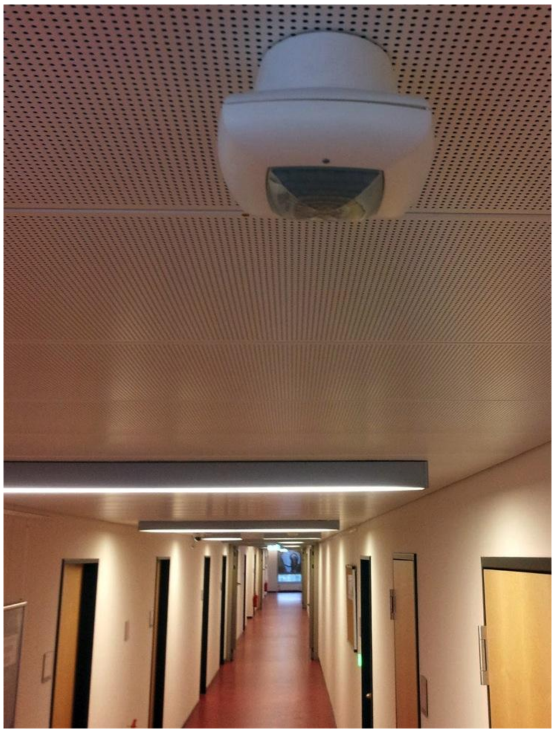

Figure 3. Siemens UP258/21 presence sensor in the 3rd floor of the office building under investigation. Sensors of this type are installed approx. every 8 meters in corridors and in some rooms.

III. An Intrusion Detection System for Buildings

In a straightforward approach for developing IDS for buildings we would look at all sensors observing the building. We would be able to determine sensor types and their positions in a documentation describing the installation. In a second step we would determine sensors that most likely indicate an intrusion. Such sensors might be located near to entry doors and exits or observe floors. A third step would be either learning about normal sensor events or defining static rules for alerts manually.

In various cases we review a documentation of the installation network was not readily available. Even in cases it would be available, configuring an IDS manually would be very inconvenient. All information would have to be gathered and analyzed, rules would have to be set up or training would have to be performed.

Instead, we want to build an IDS in a way that does neither require manual training nor manually introducing rules for pat- tern matching. This requires learning about normal behavior of the building and people inside the building. The IDS shall be initially connected with the building's infrastructure and learn about sensors by itself without adding further data. For this, the IDS need to distinguish between sensors that are not directly influenced by human behavior and those that are.

An example for the first category would be temperature sensors that deliver sensor readings every couple of minutes. Obviously, those sensors are not delivering any substantial information about an ongoing intrusion attempt. Contrary, examples for the latter category are presence detection sensors, especially presence detection sensors that are regularly triggered by humans.

Sensors that would be triggered very seldom might indicate an intrusion better, but are usually hard to identify in a scenario with unsupervised learning. Examples for such a sensor are presence detection sensor or light switches in rarely used rooms such as storage rooms. In those cases, manually added rules might be appropriate, but this is not in the focus of this paper. When anomaly detection is being performed in such cases, every motion detected by such sensors would be recognized as anomaly since it is a rare event.

Consequently, building and using an Intrusion detection system for buildings with unsupervised learning of normal behavior and anomaly detection requires the following steps:

- Connect the IDS to the house installation network and record all available sensor events.
- Analyze data and searching for sensors tracing human behavior. Grouping sensor events according to their suitability to measure anomalies in human behavior.
- Learn normal behavior over time.
- Detect anomalies.
- Raise alert, when an anomaly has been detected.

A. Requirements

As IDS utilize different categorizing algorithms (see Section II-A), there will never be an absolutely correct detection. Additionally, anomaly detection algorithms require training data representing normal behavior. Hence, the goals for the proposed IDS for buildings are:

- No false negatives, meaning alerts should be raised whenever an intrusion takes place.
- Low rate of false positives, meaning alerts should not be raised without an ongoing intrusion.
- Short training periods, meaning the system will be ready to detect intrusion within a short time.
- Alerts should have a meaning indicating the cause of the alert.

B. Limitations

We assume that an intrusion is a rare event. Hence, most of the time sensors will deliver data that represents normal behavior. We further assume that sensor data representing an intrusion is statistically different to sensor data representing normal behavior. This is not always true. An intruder could hide in large groups of people or imitate legit visitors, causing normal sensor data.

C. Connecting and gathering data

As the structure and topology of most installation networks allows recording all messages at any point of the network, installing a recording device is relatively simple. In case of a typical KNX installation a two-wire cable has to be connected to a KNX-IP-gateway. All data can be stored in a database for later examination. For the purpose of learning data representing a period of normal behavior has to be collected. For our purpose two to four weeks are sufficient as initial set for unsupervised learning. More data will become available during the execution of the IDS later.

D. Searching for suitable sensors

By filtering all sensors that do not possibly represent human behavior we dramatically limit the number of events and reduce noise. When looking for sensors that indicate human behavior instead of delivering the actual outside temperature we have to find a suitable discriminant.

Pragmatically, two indicators are suitable, periodicity and time-precision of repeating events. Sensors directly reporting human actions will generate events with long periodic patterns, such as daily or weekly re-occurrence, since human behavior follows patterns with similar periods. Most technically caused events will have short periodic patterns and will occur with more exact periodic time intervals.

There are several sensor types in a typical installation network which can be evaluated: *Presence detecting sensors*, such as the infrared sensor shown in Figure 3 detect the presence of humans - but also of animals such as dogs, which is a suitable application of the IDS we present here. Despite, in this paper, we ignore animal-based scenarios, as they are special cases of the general scenario. The main application of such presence detecting sensors is commonly to automate the lighting. *Temperature sensors*, which are part of the climate control. Temperature sensors may be installed inside as well as outside. When an air condition is deployed, there are probably also door- and window-contacts installed to prevent the air condition working when windows or doors are wide open. In rooms without automatic lighting control, light switches are the easiest way to control the lighting and, hence, are usually connected to the installation bus as well.

As mentioned before, domain knowledge indicating types of sensors, location of sensors, meaning of telegrams sent by sensors, and purpose of sensors should not be used for the system. The reasons are:

- The system should be deployable by everybody without special knowledge.
- The system should adapt itself when new sensors are installed.
- The system should be deployable when no domain knowledge is available or where it is only available at high efforts.
- The system should not need to update a database of sensor types available on the market.

When using a pragmatic approach, sensor data has to be mined for periodic events with high deviations. Short periods with a very precise timing (events occur exactly as scheduled) are typical signs for artificial causes usually not indicating human behavior. As regular events the deviation over time should be relatively low. Figure 4 shows events generated by such a sensor. Sensor events caused by human activities will in contrast to that usually follow daily and weekly patterns. The deviation of the number of events over time will be much higher - see Figure 5.

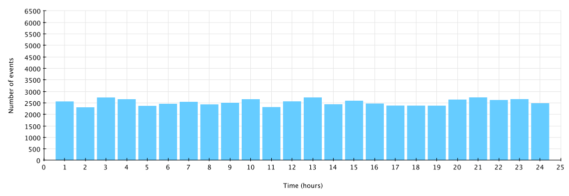

Figure 4. Number of events per hour generated by a temperature sensor. Standard deviation is below 150. This indicates a sensor not monitoring human behaviour.

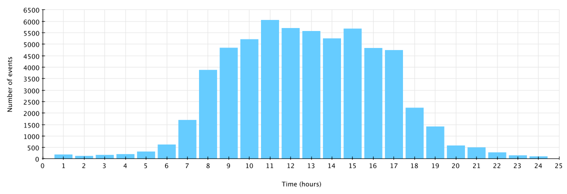

Figure 5. Number of events per hour generated by a temperature sensor. Standard deviation is below 150. This indicates a sensor not monitoring human behaviour.

E. *Learning normal behavior*

During the training sequence of the IDS, a database with all sensor events is built. In an unsupervised scenario, which we aim to, there are no labels attached to each data record. During the training period all behavior will be considered as normal. It is important to ensure the recorded behavior to be truly normal; otherwise intrusions may remain undetected as the behavior model for normal behavior covers them.

Besides this, the duration of the recording needs to be discussed: A short period may not contain every behavior (e.g. cleaning personnel who visits rooms only once a month and might consequently cause long periods in training data). This might lead to false positives or training data, which does not contain workdays as well as non-workdays: The use of most building differs quite significantly between those situations. Generally, periodic events cannot be learned if the training period is chosen to short.

This interest conflicts with the interest of having the IDS in a productive state as fast as possible. Thus, a compromise between sufficient collection of training data and offset between installation and productive use has to be found. Learning is a permanent process in order to adapt to slightly changing behavior.

F. *Detecting anomalies*

Anomalies can be detected once enough training data have been collected to differentiate anomalies from normal behavior. Data is filtered for sensors events, which indicate human behavior as explained before. Once enough data have been collected, the IDS evaluates current activities by continuously collecting the relevant sensor events. Significant anomalies will trigger an operator alert.

A variety of parameters can be compared with normal behavior. Examples are:

- Number of events per time. Suitable time slots are in the range of a few minutes. A sliding window could be used when appropriate. Longer time slots would prolong the time before an alert might be raised. Shorter time slots could increase the influence of noise in the data.
- Sequence of events. Certain events might significantly correlate with other subsequent events, such as presence detection sensors, which observe a corridor in a row. A number of typical event

sequences can be learned. Such sequences may include simple sequences consisting of door sensors and light switches but also quite complex sequences covering presence detectors, window and temperature sensors and the climate control. The more complex the event sequence, the more training data will be necessary as humans tend to vary such sequences (e.g. a warm summer evening: switch off the lights, open the window, switch off the air condition - or open the window, switch off the air condition and then darken the room?).

- The data originating from sensors detecting presence (light switches, infrared sensors) could be used to deter- mine situations preceding certain actions causing further events or the absence of events, for example when a person moves to the sleeping room (domain knowledge would be required), activity in other rooms is unlikely to appear without presence sensors noting a person going there. When domain knowledge is not available, which is required for the system, sequences of events will remain in the data.

Without domain knowledge modeling human behavior and deducting expected sensor patterns in a generative approach is not suitable. Instead, a discriminative approach needs to be applied. We present a more concrete solution utilizing general outlier detection techniques.

IV. Implementing an example

To validate our assumptions, we used data from the building automation system of the Institute of Computer Science building at the University of Rostock for the 3rd floor, which has been collected over 2 months (April and May 2012).

Our IDS example implementation consists of two separate evaluation steps: Evaluation of activity data for each single sensor and evaluation of activity sequences, meaning sequences of sensor events caused at different sensors which usually are installed along walking paths. Sequences of sensor events are considered because they carry more significant information than punctual events regarding underlying intentions, which might indicate an intrusion.

For instance, a sequence of events might indicate a person walking through the building while a single event simply indicates presence. The evaluation of *single sensor data is done* by comparison of the current sensor data with what is expected from a FFT of the training data. This test covers all recurring events. It can make sense to divide the training data: During working time there will probably be other dominant frequencies than during the night.

The next evaluation step covers *series of sensor events*, which are reasonable to be triggered sequentially (e.g. sensors 3.6.11, 3.5.54, 3.5.58, and 3.5.57 in Figure 6) in contrast to sensors which might correlate without cause (e.g. sensors 3.6.6, 3.5.14, and 3.6.12 or 3.6.20 in Fig. 6).

A variety of methods to detect abnormal behavior has been developed in the context of intrusion detection systems - see section II-A2. For evaluation purposes we have chosen three easy methods:

- Detection of single point anomalies. The detection is based on the recognition of an unusual number of sensor events for particular sensors. An alert is raised when the number of events exceeds a dynamically defined thresh- old which depends on former observations (learning) within a given time slot. This can be repeated for all sensors, which have to be determined as useful according to the criteria explained in section III-D.
- Context based anomaly detection. Context is defined by the time of the day and the day of the week together. The system has learned typical behavior (number of sensor events) over several weeks. Daily and weekly patterns are recognized during the training period and later being considered for intrusion detection during the active phase.
- Collective anomaly detection [15]. The number of events of all sensors will be accumulated over sections of time, for instance 15 minutes, and compared with a reference value.

V. Tests and results

A. Test setup

To test the intrusion detection system, we first provided the data of two months without intrusion for learning purposes to the IDS. In order to generate a repeatable test under controlled environment we sampled data of some test runs through the area we used for testing (see floor plan in Figure 6). We copied these data records and added them to a recorded real data stream (list of sensor events) at different times and in different order (a virtual person would consequently run in different directions). Hence, we generated virtual traces of persons in the data set.

We inserted the virtual walk through in several situations representing:

1. A person walking through the building.
2. An abnormal activity, which would represent persons, leaving the building in a hurry.
3. Several persons walking in either the same or opposing directions.

For pragmatic reasons we had to define a certain situation as intrusion or not. In reality, a situation could be judged as intrusion after an attack took place. There are clear indications of an intrusion, but these indications cannot be expressed as a discriminate mathematical formula.

We configured the following *test cases*. In tests 1 and 2 we gradually increased the number of virtual traces per fifteen minutes until the IDS raised an alert. We did not overlap two or more traces in these cases, hence, only one virtual person runs through the building at a time. We performed this at several virtual times of the day and several times of the week (see table I for details).

For test 3 we created records representing the sensor events that would be triggered by a group of persons walking through the floor and applied it as sketched before for a single virtual person.

The following Table I shows results of test runs. A divergence of 20% is defined as an intrusion. Thus, every simulated activity with more than 20% difference to the trained data set will be marked as anomaly. We test whether the anomaly detection works as intended. Caused by the nature of intrusion detection systems it is generally impossible to judge the quality of the system through this kind of tests.

TABLE I. RESULTS OF A CASE STUDY

Time	Day of Week	Number of virtual persons	Alert raised (Single sensor)	Alert raised (Con-text based)	Alert raised (Collec-tive)
0:00	Mo	1	Y	N	N
		2	Y	N	Y
		5	Y	Y	Y
		10	Y	Y	Y
		100	Y	Y	Y
0:00	Fr	1	N	N	N
		2	N	N	Y
		5	Y	N	Y
		10	Y	Y	Y
		100	Y	Y	Y
0:00	Su	1	Y	Y	N
		2	Y	Y	Y
		5	Y	Y	Y
		10	Y	Y	Y
		100	Y	Y	Y
11:30	Mo	1	N	N	Y
		2	N	N	N
		5	N	N	Y
		10	N	Y	Y
		100	Y	Y	Y
11:30	Fu	1	N	N	N
		2	N	N	N
		5	N	N	Y
		10	N	Y	Y
		100	Y	Y	Y
11:30	Su	1	N	N	Y
		2	Y	Y	Y
		5	Y	Y	Y
		10	Y	Y	Y
		100	Y	Y	Y

VI. CONCLUSION AND OUTLOOK

We have shown that it is possible to use modern installation bus sensor data to create an intrusion detection system. For training purposes, a set of data, which is known to be intrusion free.

We tested the system in an experiment at night to provide a ground truth. Our results indicate a working system, while improvements remain possible, especially by changing the way anomalies are being detected and sensor events are evaluated (see following section). By configuring the system to appropriate settings we were able to ensure a minimum number of false negatives without avoiding too many false positives.

A. *Alternative uses of an IDS as spying device*

As the use of an IDS requires a significant amount of sensors (see Section II-B2) delivering live data, the IDS, its collected data, and the incoming sensor events become an interesting target for abuse. Abuse scenarios may cover supervisors observing employees behavior (which may violate local law), but also intruders searching for possible security breaches.

The IDS and its behavior model may increase the possible abuse if the model description can be accessed: It contains detailed information about movement pattern (derived from presence sensors), event sequences, and recurring events with their frequencies.

Consequently, sensor data should be treated with care and be secured against unauthorized access. The problem of abuse by authorized users can't be avoided and has to be solved using social, ethical, or legal measures [16][17].

The implementation of an example clarified the need for more domain knowledge than used in network intrusion detection systems. This need is due to the ambiguous nature of the combination of sensor events and machine learning. Network sensors have a much smaller set of observable activities than sensors within an intelligent building.

Anyhow, the learning algorithm will "learn" something and probably also pass all (prepared) tests. But, unfortunately, those algorithms never clearly expose, which context they *really* learned: Did it learn that activity is not caused by an intruder, when the wash room lights in the 3rd floor are lit? Or is it no intruder because the activity takes place during the weekend working hours?

Thus, domain knowledge has to be used in selection of sensors whose data is used for evaluation purposes and context analysis. One of the most valuable pieces of domain knowledge is the day of week and the time.

B. *Further ideas and improvements*

The algorithms currently used to evaluate the incoming sensor events could be replaced with more powerful algorithms from the field of machine learning, e.g. Hidden Markov Models.

The use of sequences of sensor events instead of single sensor events promises a more accurate identification of underlying human activities. One sensor denotes presence; two or more sensors are able to differentiate activities such as walking or running. It even allows tracking whole movement profiles [18] and thus a much more precise behavior analysis and therefore a more precise identification intrusive behavior.

REFERENCES

[1] R. Heady, G. Luger, A. Maccabe, and M. Servilla, “The Architecture of a Network Level Intrusion Detection System,” tech. rep., Computer Science Department, University of New Mexico, 1990.

[2] Z. Li, A. Das, and J. Zhou in Information Assurance Workshop, 2005. IAW ’05. Proceedings from the Sixth Annual IEEE SMC.

[3] Z. Yu, J. J. P. Tsai, and T. Weigert, “An adaptive automatically tuning Intrusion Detection System,” ACM Trans. Auton. Adapt. Syst., vol. 3, pp. 10:1–10:25, Aug. 2008.

[4] Y. Zhang, W. Lee, and Y.-A. Huang, “Intrusion detection techniques for mobile wireless networks,” Wirel. Netw., vol. 9, pp. 545–556, Sept. 2003.

[5] Abraham, J., “A survey of Intrusion Detection for ad-hoc networks”, Journal of Global Research in Computer Science, 2013, 4, 182-185

[6] M. S. Najjar and M. Abdollahi Azgomi, “A distributed multi-approach intrusion detection system for web services,” in Proceedings of the 3rd international conference on Security of information and networks, SIN ’10, (New York, NY, USA), pp. 238–244, ACM, 2010.

[7] A. S. Aneetha, T. S. Indhu, and S. Bose, “Hybrid network intrusion detection system using expert rule based approach,” in Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, CCSEIT ’12, (New York, NY, USA), pp. 47–51, ACM, 2012.

[8] W. Kastner, G. Neugschwandtner, S. Soucek, and H. Newmann, “Communication systems for building automation and control,” Proceedings of the IEEE, vol. 93, no. 6, pp. 1178–1203, 2005.

[9] H. Merz, T. Hansemann, and C. Hübner, Building automation: communication systems with EIB/KNX, LON und BACnet. Springer Verlag, 2009.

[10] Y. Kyselytsya and T. Weinzierl, “Implementation of the KNX standard,” in Tagungsband KNX Scientific Conference November, 2006.

[11] S. Cavalieri and G. Cutuli, “Proposal and evaluation of deterministic access in KNX standard,” in Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on, pp. 1674–1679, IEEE, 2008.

[12] H. Langels, “KNX/IP - Using IP networks as KNX medium,” in Proceedings of the KNX Scientific Conference 2008, 2008.

[13] B. Malinowsky, G. Neugschwandtner, and W. Kastner, “Calimero: Next generation,” in Proc. KNX Scientific Conference 2007, 2007.

[14] B. Erb, G. Neugschwandtner, W. Kastner, and M. Kögler, “Open-source foundations for pc based KNX/EIB access and management,” in Konnex Scientific Conference, 2005.

[15] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys (CSUR), vol. 41, no. 3, p. 15, 2009.

[16] Martalo, M.; Ferrari, G. & Malavenda, C. S. “Wireless Sensor Networks and Audio Signal Recognition Effective Surveillance for Homeland Security: Balancing Technology and Social Issues”, Chapman & Hall/CRC, 2013

[17] Hongseong, C., “A Design of Advanced Authentication Method for Protection of Privacy in M2M Environment” architecture, 2013, 2

[18] Mundt, T.; Krüger, F. & Wollenberg, T., “Who Refuses to Wash Hands? Privacy Issues in Modern House Installation Networks”, *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2012 Seventh International Conference on,* 2012, 271-277

ACKNOWLEDGMENT

Andreas Dähn is funded by “Landesgraduiertenförderung Mecklenburg-Vorpommern”.

Thomas Mundt works since 2003 as principal engineer at the Group of Information and Communication Services at the University of Rostock. He holds a PhD from the same institution. Thomas teaches and researches in the areas of Authenticated Positioning, Privacy, Security, and Network Protocols. Before joining the University he worked as software architect for several companies.

Andreas Dähn is researcher at the Group of Information and Communication Services at the University of Rostock where he also finished his Diploma in Computer Science. Andreas has been assigned a PhD scholarship from the State of Mecklenburg-West Pomerania. His research interests cover Network Neutrality, Security, and Usability.

Stephan Sass is a Master student at the University of Rostock. Stephan holds a Bachelor degree from the same institution, which he earned by designing and implementing the described Intrusion Detection System for KNX.