# Expert access control system for access management in a smart home system*

Vira Titova[1,*,†], Yurii Klots[1,†], Viktor Cheshun[1,†], Serhii Mostovyi[1,†], and Abdel-Badeeh M. Salem[2,†]

[1] *Khmelnytskyi National University, 11, Instytuts'ka str., Khmelnytskyi, 29016, Ukraine*
[2] *Ain Shams University, El-Khalyfa El-Mamoun Street Abbasya, Cairo, Egypt*

**Abstract**

Smart home systems continue to evolve toward higher levels of autonomy, where devices independently exchange data and make local decisions without relying solely on centralized controllers. While this improves usability and flexibility, it simultaneously increases the attack surface by enabling compromised devices to influence system behavior from within. Existing security mechanisms primarily focus on protecting privacy and the confidentiality of sensor data, leaving internal interactions between devices insufficiently protected. As a result, insider threats – particularly those arising from malicious or compromised nodes capable of executing unauthorized operations – remain largely unaddressed.

This article examines the challenge of securing device-to-device interactions in dynamic smart home environments by employing a context-based expert access control system. The study includes an analysis of modern smart home architectures, device communication models, and their built-in security mechanisms, with a special focus on the limitations of standard IoT protocols used for automation. Based on this analysis, a set of unresolved risks was identified, including the inability of existing models to detect conflicts between device operations and the current system context.

To overcome these limitations, the authors developed a context-based access control model explicitly tailored to smart home infrastructures. The model incorporates dynamic role assignment, trust-level evaluation, contextual constraints, and an extended rule base capable of identifying conflicting operations before they are executed. These rules form the foundation of an expert system designed to detect compromised devices, prevent unauthorized actions, and enforce safe system states.

Experimental validation was performed using a simulated smart home environment generated with discrete-event modeling. A series of threat scenarios was executed, including attempts to alter configuration parameters, perform unexpected network operations, and generate abnormal sensor readings. The results demonstrated a high level of accuracy in identifying context violations and detecting potentially malicious device behavior, confirming the effectiveness of the proposed approach. The developed expert system provides a reliable mechanism for strengthening internal security in smart home networks and mitigating insider threats..

**Keywords**

Smart Home, Z-Wave, ZigBee, Security Threats, Context-based Access Control Model, Expert System.[1]

## 1. Introduction

Automated building management systems, commonly referred to as "smart home" technology, are increasingly utilized to address various challenges faced during building operations. This technology employs modern automation systems and a range of peripheral devices to enhance security, conserve resources, and improve overall living conditions. A key feature of smart home technology is the active interaction among various automated subsystems, which enables the system to recognize and respond effectively to different situations.

Numerous communication protocols are available in the field of building automation; however, there are currently no universally accepted standards for networking the devices that comprise a smart home system [1,2]. Utilizing local area network (LAN) technologies is often ineffective due to

* *AdvAIT-2025: 2nd International Workshop on Advanced Applied Information Technologies: AI & DSS, December 05, 2025, Khmelnytskyi, Ukraine, Zilina, Slovakia*
[1,*] Corresponding author.
[†] These authors contributed equally.
titovav@khmnu.edu.ua (V.Titova); klots@khmnu.edu.ua (Y. Klots); cheshunvn@khmnu.edu.ua (V. Cheshun); serhii.mostovyi@khmnu.edu.ua (S. Mostovyi); abmsalem@yahoo.com (Abdel-Badeeh M. Salem)
0000-0001-8668-4834 (V. Titova); 0000-0002-3914-0989 (Y. Klots); 0000-0002-3935-2068 (V. Cheshun); 0000-0002-9505-3206 (S. Mostovyi); 0000-0003-0268-6539 (Abdel-Badeeh M. Salem)

their inherent redundancy. Technologies used in smart home systems must meet specific criteria, including low power consumption, high reliability, secure transmission, low cost, and ease of physical deployment. It is also important to note that many applications do not require high data transfer rates.

When comparing wired and wireless networks, the ease of physically deploying network devices is a decisive factor. A family of competitive wired network technologies known as Power Line Communication (PLC) relies on the fact that most premises are electrified. This allows for the establishment of both wired and wireless networks using technologies such as X10, INSTEON, HomePlug, and LonWorks for communication via electrical wiring, as well as Bluetooth, Z-Wave, and ZigBee for wireless communication.

Z-Wave, a proprietary protocol stack developed and supported by the Z-Wave Alliance, is considered one of the most promising protocols for smart home systems. Critical components of these automation systems, such as locks, currently utilize AES-128 encryption. However, this encryption is an extension of the standard, meaning that older devices may not support it. Moreover, some devices have vulnerabilities in their key exchange protocol implementations, which can allow unauthorized access using the default key "00000000000000000000000000000000h" [3].

Several solutions exist to facilitate the management of smart home systems based on the Z-Wave protocol. One such solution certified by the Z-Wave Alliance is Z-Way, which provides a web interface and API for interacting with the system. Unfortunately, authentication and data encryption mechanisms are not included. Consequently, if an attacker compromises the local network, they can take arbitrary actions within the smart home system.

ZigBee is an open wireless communication standard designed for automation systems. The standard includes upper-layer network protocol specifications for the application and network layers, while the lower-layer services – medium access control and physical layers – are governed by the IEEE 802.15.4 standard. The application layer defines the ZigBee device object, support layer, and the application development interface (ADI), which specifies standard data types, service discovery descriptors, and packet formats, enabling rapid development of simple, attribute-based profiles. Application objects are software modules that control ZigBee devices at the endpoints.

The application support sublayer provides data to applications and manages the connection of devices to the ZigBee network, also storing device data. ZigBee networks can operate in modes that permit unencrypted communication; however, the standard security level does not guarantee secure distribution of network keys.

To mitigate replay attacks, a mechanism for monotonically increasing counters is integrated, but implementing this can induce network issues, necessitating manual counter resets. Without this option enabled, replay attacks can easily occur [4]. Tools like the KillerBee framework have been developed to analyze ZigBee networks and demonstrate such vulnerabilities in practice.

Overall, smart home systems encounter several security threats common to many computer networks [5]. An examination of various attacks and the vulnerabilities leading to their success can be found in Table 1.

Based on the threats examined, creating methods for protecting smart home systems is currently a pressing scientific task, which is the subject of this article.

## 2. Review of existing solutions

A smart home includes many interconnected devices, such as sensors, cameras, controllers, and other smart components, that provide comfort and security. However, these devices also create vulnerabilities that can be exploited to compromise the system's security. Based on the threats listed above, let us examine the shortcomings of existing smart home security methods.

Many smart home devices can connect via common channels such as Wi-Fi or Bluetooth. Existing security methods are often insufficiently effective at scanning connected devices for viruses or malware, allowing infected devices to penetrate the system and threaten the security of other smart home components [6,7].

Internal smart home networks can be vulnerable to attack through weak passwords or a lack of security updates. Traditional security methods, such as antivirus software or firewalls, are ineffective against sophisticated attacks, especially when multiple devices interact [8,9].

**Table 1**
Security threats to smart home systems

| Attack type | Vulnerability | Possible consequences |
|---|---|---|
| Attacks on the central node | Connecting a smart home network to the internet. Lack of (ineffective) network perimeter security mechanisms | Disruption or failure of the central server, and consequently the entire system. Violation of the confidentiality, integrity, and availability of information |
| The impact of viruses and Trojans on system operation | Connecting a smart home network to the internet. Lack of (ineffective) network perimeter security mechanisms | System software failures, resulting in system hardware malfunction or failure. Violation of the confidentiality, integrity, and availability of information |
| Interception of information transmitted via wired and wireless communication channels | An attacker can access wired channels or the network's radio signal interception zone. Lack of (or ineffective) traffic protection mechanisms | Violation of the confidentiality of information transmitted over the channel. Possible system control takeover |
| A malicious user with administrator rights gains access to the central node by stealing passwords and other access control credentials | Lack of (or ineffective) authentication and identification mechanisms | Violation of the confidentiality, integrity, and availability of information located within the network |
| Unauthorized users access the network | Lack of (or ineffective) authentication and identification mechanisms | Violation of the confidentiality, integrity, and availability of information located within the network |
| User errors | Lack (ineffectiveness) of system protection mechanisms against user misuse | Violation of confidentiality, integrity, and availability of information. System failures are possible due to improper use of equipment |
| System hardware failure | Low equipment reliability, low personnel qualifications | Violation of confidentiality, integrity, and availability of information |
| Software errors | Use of unlicensed software, low qualifications of personnel, and a lack of access to information (inefficiency) in testing purchased software | Violation of confidentiality, integrity, and availability of information |

Existing systems may use simple data processing algorithms that cannot always account for possible sensor failures. For example, if a temperature sensor produces incorrect data, the system may fail to recognize the problem and continue executing erroneous commands, leading to undesirable consequences (e.g., overheating or cooling) [10,11].

When devices fail, commands may be sent to the system in an incorrect format or with errors. Current security methods are typically unable to effectively diagnose such errors and prevent their impact on the system, which can lead to unstable or even dangerous operation [12,13].

Configuring smart home devices often depends on predefined settings and user preferences. Configuration errors can lead to improper operation, causing inconvenience or even a security threat. Modern security systems are not always able to adequately manage complex and changing configurations [14,15].

Errors in configuring device operating modes (for example, incorrectly configured heating or security systems) can lead to malfunctions. This creates additional security risks, as it is impossible to predict or control the system's behavior in the event of errors [8,16].

Given all these threats, the need for more comprehensive and flexible security systems that can effectively prevent and respond to potential threats is clear [17]. In this context, access control models (ACMs) are becoming integral to any smart home system. Access control models can ensure access rights are differentiated between different devices and users, preventing unauthorized actions and minimizing the risks associated with malicious code, device failures, and configuration errors. The use of ACMs helps prevent many types of attacks, restricts access to only necessary devices, and ensures protection at the smart home infrastructure level.

Simple access control models (e.g., basic login and password authentication mechanisms) have several significant shortcomings that can be critical to smart home security. Limited authentication capabilities. Simple access control systems cannot effectively identify a device or user based on multiple factors, such as user behavior, interaction context, or current state [9,18].

Lack of flexibility in responding to threats. Simple access control systems often operate according to rigidly defined rules. They cannot adapt to changes in context, such as changes in device behavior depending on time of day, operating mode, or other factors [19,20].

Vulnerability to brute-force attacks. Simple models based on passwords or PINs are susceptible to brute-force or phishing attacks, allowing attackers to bypass security [21,22] easily.

Ineffectiveness in distributed systems. In a smart home system, devices may be located at different points on the network, and a simple access control mechanism cannot effectively manage access between multiple system components, especially when using different communication protocols and the vulnerabilities associated with them [23, 24].

The context-based access control model provides a much more flexible and robust security solution than simpler methods. In the smart home context, the following factors can be taken into account:

1. User and device context. Models can track not only user identity but also current behavior, such as user location, access time, and task context. For example, access to a heating control system can be permitted only during certain hours or only when the user is in the home.
2. Dynamic adaptation. Context-based systems can change their decisions based on the current system state, for example, blocking access if a sensor detects anomalies in device operation or if an unauthorized connection attempt is detected.
3. Distributed access control. In a smart home, many devices operate autonomously, and context-based models allow access control to be integrated at the device level, not just at the central server or user level.

Thus, the context-based access control model provides higher protection and flexibility, which is critical for maintaining security in complex and dynamic environments such as the smart home.

## 3. Context-based Access Control Model

Context-based access control models differ from traditional models that rely on static information about the protected system. Instead, they utilize information about the system's state when the controlled operation is executed. This information is referred to as "context."

Context represents the aggregate state of system devices during an operation. A device's state is defined by its parameters, which can be read by third-party devices. Therefore, context consists of various elements describing measurable system parameters. Examples include the current time, device location, and tasks being performed.

Context may also encompass historical data on individual parameters. In this case, it is essential to store information about the relationships between these parameters to track their combined changes over time. The presence of historical data does not contradict the definition of context as the state of the system at the time of an operation, as the current state results from certain previously occurring events reflected in the context's change history.

One crucial task in developing context-based access control models is the collection and analysis of context. For effective access control purposes, it is important to prepare the raw data gathered from sensors. The following data preparation stages can be identified:

1. Data Normalization. This process involves converting data from various devices into a unified format suitable for processing by the access control system.
2. Data Filtering. This stage focuses on identifying data that is relevant for analysis.
3. Data Correlation. At this initial data preparation stage, various dependent values are correlated.

Access control in context models is conducted based on extended rules derived from context. These rules refine access rights according to known context parameters. An example of a language used for context-based access control rules is the access control policy language, which is similar to the predicate logic employed in expert systems.

In this work, the context model builds upon the role model by dynamically assigning roles based on context.

The access control policy establish in advance, taking into account the conditions and requirements of the protected system. Particular attention give to developing rules for assigning a user's trust level based on context values. The access-granting process can be represented as a diagram, as shown in Figure 1.

Figure 1 illustrates the main components involved in the access process. A typical scenario is depicted, where a user seeks access to a service provided by one device within a "smart environment."

Communication between the user and the device occurs through specialized equipment supporting the "smart environment," which connects devices into a single network and collects context information. The environment includes an access broker, designed to perform user access verification operations.

A service user submits a request to operate, which requires a specific access level. Before executing the request, the broker requests information about current access rights from the access control service, providing it with the current context.

The access control service, which stores the system's security policy, determines access rights using the following algorithm:

1. The digital signature of the user's context is verified. If the signature is valid, the process continues. Otherwise, the access control service denies access.
2. The user's trust levels are calculated based on the context elements. At this stage, the context is analyzed, and these values will not be used further.

3. The user's ability to be assigned roles is checked based on the trust levels of the context elements, resulting in a list of user roles.
4. Based on this list of roles, access rights to the requested service are verified. The operation result is then returned to the broker.

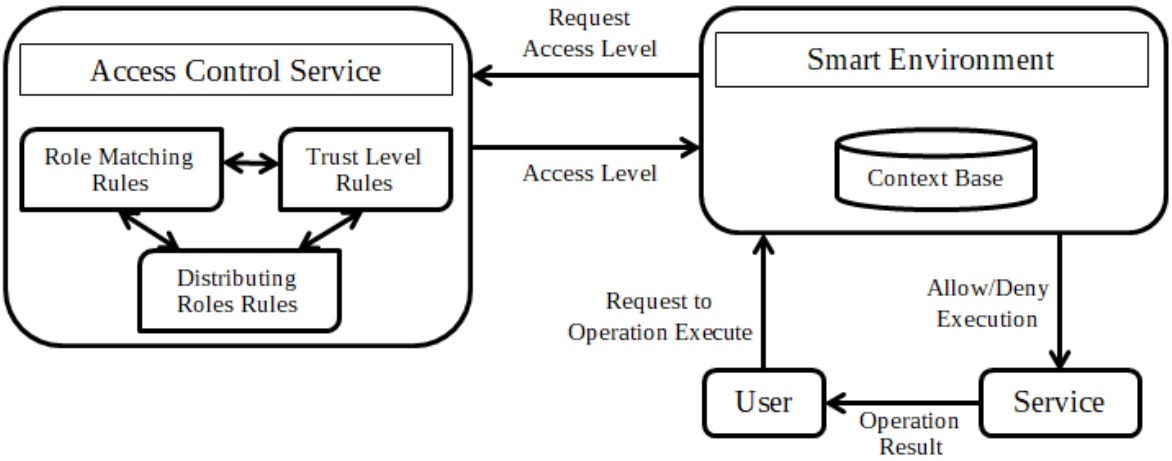


**Figure 1:** Generalized architecture of a context-based access control mechanism.

Then, if access is granted, the requested operation is performed. The service receives information about the user's access level and reports the operation's result.

## 4. Expert Access Control System for Access Management in a Smart Home System

The ease of use of an expert access control system is influenced by the approach to defining access control rules. In our proposed model, rules are presented in two groups: constraints on system parameters based on conditions, and permissible operations of subjects on objects. The rules describing permissible operations of subjects on objects are as follows:

$$Execute = (S, O, A),$$

where $S$ – is the subject, $O$ – is the object, and $A$ – is the operation. Operation is part of the set of operations that can be performed on an object.

The following functions are defined to obtain this and other information related to the object:

$$Operations(O) = \{operations\ allowed\ on\ object\ O\}$$

$$Description(O, A) = \{change\ of\ parameters\ O\ after\ operation\ A\}$$

The first function is reference and serves to verify the correctness of rules. The second function is used in context analysis to identify conflicts that arise during potential operations. The proposed expert system assumes that access rules are determined by the devices added to the system. This eliminates the need for manual configuration of the access control system, which is important for applications such as smart home security. To achieve this, devices are divided into predefined classes with specific operations they can perform. Each device is then given a list of access rules for device classes.

When a new device is connected to the smart home system, the access control policy is updated by merging the rule lists of the system's devices.

Parameter constraints are specified in one of the following forms:

*Constraint* (*context_condition, parameter, value_expression*)

*Constraint* (*context_condition, conditional_expression*)

The first form lets you specify a hard-coded value that a parameter must take when a condition is met. This rule form also allows you to define new context elements based on system devices' definitions.

The second form is used to impose arbitrary constraints on context elements. Context element constraints are fragments of the description of the safe state of the smart home system. The primary objective of context analysis in the developed expert access control system is to detect inconsistencies between potentially possible operations in the smart home system and established context restrictions.

Therefore, the context analysis algorithm is used to determine the list of current restrictions *Constraint* and rules of type $Execute = (S, O, A)$ that conflict with the current restrictions.

An important feature is that rules of the *Constraint* (*context_condition, parameter, value_expression*) type allow the context to be supplemented with new calculated parameters. This capability allows for creating complex rule chains that can be supplemented by rules from new devices introduced into the system.

This same feature must be taken into account when conducting context analysis. To describe the algorithm with this in mind, the concept of an “involved rule” *Constraint* is introduced. At the beginning of the algorithm, a list of rules *Constraint* to be processed is compiled. Initially, the list consists of all rules present in the system. An “involved rule” is a rule from the list of rules to be executed in the current context. This rule is applied immediately upon activation. Further actions depend on the rule type.

If this rule has the form *Constraint* (*context_condition, parameter, value_expression*) then a new parameter is added to the context as a result of the calculation *value_expression*.

If such a parameter exists in the context, the new value is added to the list of possible values. The new value is also added to the list of context constraints. Contradictions are not identified at this stage since they do not affect the integrity level assignment.

If a rule of the form *Constraint* (*context_condition, conditional_expression*) is invoked on the context constraint list, then the condition is added. After a rule is invoked, it is removed from the list of rules to be processed. This process is repeated until, at the next iteration, no more invoked rules are left in the list of rules to be processed.

Further processing of the compiled list of context constraints involves identifying rules that violate these constraints. To do this, for each rule *Execute*=(*S, O, A*) used in the system, the resulting parameter changes are checked using the function *Description*(*O, A*). All rules whose execution would violate the conditions imposed on the context are added to the list of rules that cause conflicts. The result of the algorithm is a list of context constraints and access rules that conflict with them.

Below are examples of some rules.

Unauthorized Access Attempt:

*Constraint* (
*context_condition: time_between*(00:00, 05:00) AND d*evice_class == "controller",*
*conditional_expression: source_ip NOT IN trusted_ips* )

This rule prevents controller-level access attempts from unknown IP addresses during vulnerable hours (e.g., midnight to 5 a.m.).

Repeated Failed Login Attempts:

*Constraint* (
*context_condition: failed_login_attempts* > 5 *AND device_class == "admin_panel",*
*parameter: alert_level,*
*value_expression: "high"*)

Triggers a high alert level when too many failed login attempts are detected on the admin panel, indicating a potential brute-force attack.

Unexpected Parameter Change:

*Constraint* (
*context_condition: last_config_change_source != "authorized_admin",*
*conditional_expression: config_modified == true*)

Flags unauthorized configuration changes possibly indicating compromise of the central node.

Unknown Device Performing Restricted Operations:

*Constraint* (*context_condition: device_class == "light_bulb" AND Operation == "network_scan",*
*conditional_expression: false*)

Devices in the "light_bulb" class are not expected to perform network scans. This rule blocks unexpected operations, hinting at Trojan behavior.

Unexpected Data Transmission:

*Constraint* (
*context_condition: device_class IN ["thermostat", "camera"] AND data_sent_outside == true,*
*parameter: suspicious_activity,*
*value_expression: true*)

If typical local-only devices send data to unknown external servers, it could indicate a Trojan.

Unauthorized Parameter Creation:

Constraint (
*context_condition: new_parameter_created == true AND source_device NOT IN trusted_devices,*
*conditional_expression: false* )

Blocks unknown devices from injecting new parameters or modifying system context—this is a behavior often used by Trojans to insert hidden rules.

Multiple Failed Configuration Changes:

*Constraint* (
*context_condition = (User.failed_config_changes > 2), parameter = "ConfigLock", value_expression = "true"*)

If the user fails to apply configuration changes more than twice, lock further configuration attempts.

Unusual Access Time:

*Constraint* (
*context_condition = (User.access_time NOT BETWEEN 8:00 AND 20:00), conditional_expression = "RaiseAlert"*)

If access is attempted outside normal working hours, raise an alert

Incorrect Password Attempts:

*Constraint* (
*context_condition = (User.login_attempts > 3), parameter = "AccountLock", value_expression = "true"*)

If the user has tried to log in more than 3 times unsuccessfully, the system adds a new parameter.

Temperature Sensor Failure:

*Constraint* (
*context_condition = (Sensor.temperature.value > NormalMaxTemperature AND Sensor.temperature.trend = "not_decreasing"), parameter = "HeaterStatus", value_expression = "OFF"*)

If the temperature exceeds the normal limit and continues not to decrease, then the parameter HeaterStatus with the value "OFF" is added to the context — that is, the heating is forcibly turned off.

Battery Low Warning:

*Constraint* (
*context_condition = (Device.battery_level < 15%), conditional_expression = "SendBatteryWarning"*)

If the battery level drops below 15%, send a battery warning.

Network Module Disconnected:

*Constraint* (
*context_condition = (Device.network_module.status = "Disconnected"), parameter = "NetworkAvailable", value_expression = "false"*)

If the network module is disconnected, update the context to indicate network is unavailable.

Although the expert access control system focuses primarily on context constraints, the complete architecture of the proposed model (Fig. 1) also incorporates two additional groups of rules: Trust Level Rules and Role-Matching Rules. These rules operate at earlier stages of access decision-making and complement the context analysis mechanism.

Trust Level Rules determine the user's or device's trust level based on current and historical context elements (e.g., behavioral anomalies, time of access, deviation from typical activity, device integrity). The trust level represents an intermediate evaluation metric used to refine access rights before role assignment.

Role-Matching Rules define how roles are assigned to subjects according to their computed trust level and contextual compliance. A role is considered applicable to a user or device only if all its preconditions are satisfied (minimum trust, allowed device class, time-of-day restrictions, or security posture requirements). These rules ensure that role assignment dynamically adapts to the current state of the smart home environment.

Below are examples of possible Trust Level Rules and Role-Matching Rules that illustrate their function within the system.

Low Trust at Night for Unknown Devices:

*TrustRule(*
*context_condition: time_between(00:00, 05:00) AND device_class NOT IN trusted_devices,*
*trust_level: "low")*

Unknown devices active during nighttime are assigned a low trust level.

Trust Reduction After Repeated Failed Logins:

*TrustRule(*
*context_condition: failed_login_attempts > 3, trust_adjustment: "-1")*

Each failed login attempt decreases the trust level.

High Trust for Recently Verified Devices:

*TrustRule(*
*context_condition: integrity_check == "passed", trust_level: "high")*

Devices that recently passed integrity validation receive high trust.

Administrative Role Requires High Trust:

*RoleMatch(*
*role: "admin",*
*required_trust: "high",*
*context_condition: device_class == "controller")*

Only trusted controllers can obtain the "admin" role.

Guest Role Allowed Only During Daytime:

*RoleMatch(*
*role: "guest",*
*context_condition: access_time BETWEEN 08:00 AND 20:00)*

A total of 1,500 rule instances were developed for the expert system prototype.

## 5. Experimental studies of the system

A smart home system comprising devices of various classes was simulated. Python was chosen for implementation as a general-purpose language with a proven track record for rapid prototyping. The SimPy discrete-event simulation library was selected for modeling. The library utilizes Python's generator interface, enabling the simulation of multi-agent systems through cooperative multitasking by implementing agents as coroutines.

The smart home system description is stored in a JSON file, which specifies the devices included in the system, their parameters, and the operations Operations(O) allowed on them, along with the values of the Description(O, A) function.

The composition of the simulated system is shown in Figure 2.

The software mockup omits the network layer and focuses solely on the access control gateway, which is responsible for processing and storing context and access control. Using the SimPy library, the simulation involves sending a series of requests between devices.

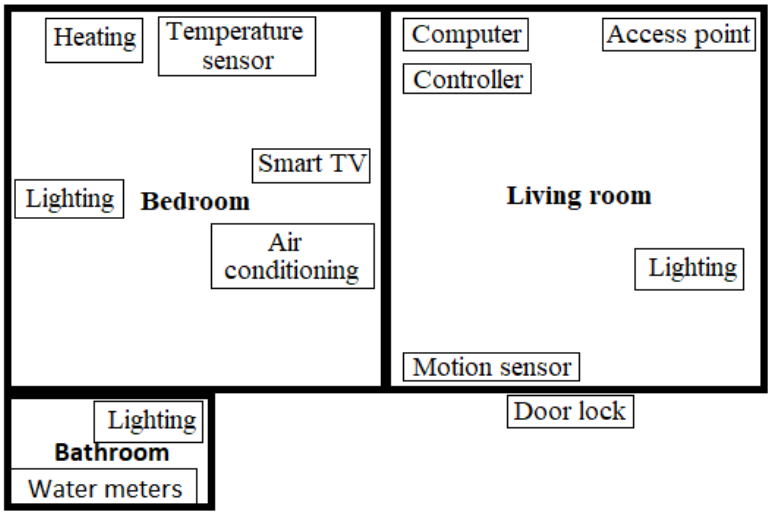


**Figure 2:** The composition of the simulated smart home system.

Its modeling and subsequent testing were conducted to evaluate the effectiveness of the developed smart home security system. Several device compromise and software failure scenarios were simulated.

The analysis demonstrated high threat detection accuracy (Table 2). The expert system correctly classified 391 of 400 dangerous scenarios, with only 2 false negatives. It also correctly identified 98 safe scenarios, with only 9 false positives. Thus, the model's accuracy was 97.8%, demonstrating its ability to detect atypical behavior in routed nodes of smart systems.

**Table 2**
Expert system scenario recognition results

| Number of scenarios | Total | Test | | | |
|---|---|---|---|---|---|
| | | TP | TN | FP | FN |
| Safe | 100 | 98 | - | - | 2 |
| Unsafe | 400 | - | 391 | 9 | - |

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

## 6. Conclusions

Insider threats to smart home systems pose a serious challenge due to the difficulty of prevention. Using an expert system based on a context-based access control model is a promising solution. Using context opens up new access control capabilities that can be fully utilized in smart home systems, as they largely align with the properties of the smart home systems themselves.

For example, the expert system is built based on rules defined by connected devices. This solves the configuration problem that arises due to the differences in the composition of each unique smart home system.

The developed prototype of the expert access control system for access management in a smart home system demonstrated its performance in simulated scenarios of device compromise and software failures.

As a result of this work, areas for further research were identified. One such area is the application of machine learning methods to decision-making regarding the assignment of access levels. This will enable the implementation of more complex access control scenarios.

Another area of focus is the development of a hardware and software prototype for practical verification of the research results, as it was impossible to evaluate the system's performance within the framework of this work.

## Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to grammar and spell check, and improve the text readability. After using the tool, the authors reviewed and edited the content as needed to take full responsibility for the publication's content.

## References


[1] Z. Jebroni, J. A. Afonso, B. Tidhaf, Smart home energy management system based on a hybrid wireless network architecture, EAI Endorsed Transactions on Energy Web (2019) 1–11. doi:10.4108/eai.13-7-2018.161437.

[2] B. Hammi, S. Zeadally, R. Khatoun, J. Nebhen, Survey on smart homes: Vulnerabilities, risks, and countermeasures, Computers & Security (2022) 102677. doi:10.1016/j.cose.2022.102677.

[3] K. Kim, K. Cho, J. Lim, Y. H. Jung, M. S. Sung, S. B. Kim, H. K. Kim, What's your protocol: Vulnerabilities and security threats related to Z-Wave protocol, Pervasive and Mobile Computing (2020) 101211. doi:10.1016/j.pmcj.2020.101211.

[4] A. Zohourian, S. Dadkhah, E. C. P. Neto, H. Mahdikhani, P. K. Danso, H. Molyneaux, A. A. Ghorbani, IoT Zigbee device security: A comprehensive review, Internet of Things (2023) 100791. doi:10.1016/j.iot.2023.100791.

[5] D. Buil-Gil, S. Kemp, S. Kuenzel, L. Coventry, S. Zakhary, D. Tilley, J. Nicholson, The digital harms of smart home devices: A systematic literature review, Computers in Human Behavior (2023) 107770. doi:10.1016/j.chb.2023.107770.

[6] O. Alshamsi, K. Shaalan, U. Butt, Towards securing smart homes: A systematic literature review of malware detection techniques and recommended prevention approach, Information 15(10) (2024) 631. doi:10.3390/info15100631.

[7] S. Sharma, C. Bhatt, A. Tripathi, Attack detection in smart home IoT networks: A survey on challenges, methods and analysis, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 544 (2024) 457–472. doi:10.1007/978-3-031-81168-5_29.

[8] G. Vardakis, G. Hatzivasilis, E. Koutsaki, N. Papadakis, Review of smart-home security using the Internet of Things, Electronics 13(16) (2024) 3343. doi:10.3390/electronics13163343.

[9] A. Huszti, S. Kovács, N. Oláh, Scalable, password-based and threshold authentication for smart homes, International Journal of Information Security 21 (2022) 707–723. doi:10.1007/s10207-022-00578-7.

[10] V. Titova, Y. Klots, V. Cheshun, N. Petliak, A.-B. M. Salem, Detection of network attacks in cyber-physical systems using a rule-based logical neural network, ICyberPhyS 2024, CEUR Workshop Proceedings 3736 (2024) 255–268.

[11] N. E. El Hady, S. Jonas, J. Provost, V. Senner, Sensor failure detection in ambient assisted living using association rule mining, Sensors 20(23) (2020) 6760. doi:10.3390/s20236760.

[12] H. Chen, J. Ma, B. Cui, J. Fu, IoTCID: A dynamic detection technology for command injection vulnerabilities in IoT devices, IJACSA 13(10) (2022). doi:10.14569/IJACSA.2022.0131002.

[13] M. Stetsiuk, Y. Klots, V. Cheshun, A.-B. M. Salem, A statistical method for real-time intrusion detection and response in ZigBee networks, CEUR Workshop Proceedings 4013 (2025) 174–188.

[14] S. M. H. Anik, X. Gao, H. Zhong, X. Wang, N. Meng, Programming of automation configuration in smart home systems: Challenges and opportunities, ACM Transactions on Software Engineering and Methodology (Apr. 2025). doi:10.1145/3731450.

[15] Y. Zhao, B. Yang, F. Teng et al., A review of intelligent configuration and its security for complex networks, Chinese Journal of Electronics 33(4) (2024) 920–947. doi:10.23919/cje.2023.00.001.

[16] M. O. Ozmen, X. Li, A. Chu, Z. B. Celik, B. Hoxha, X. Zhang, Discovering IoT physical channel vulnerabilities, arXiv (2022). doi:10.48550/arXiv.2102.01812.

[17] O. Morozova, A. Tetskyi, A. Nicheporuk, D. Kruvak, V. Tkachov, Smart home system security risk assessment, Computer Systems and Information Technologies (3) (2022) 81–88. doi:10.31891/CSIT-2021-5-11.

[18] S. Ameer, J. Benson, R. Sandhu, An attribute-based approach toward a secured smart-home IoT access control and a comparison with a role-based approach, Information 13(2) (2022) 60. doi:10.3390/info13020060.

[19] B. Li, F. Yang, S. Zhang, Context-aware risk attribute access control, Mathematics 12(16) (2024) 2541. doi:10.3390/math12162541.

[20] V. M. Teslyuk, Kh. V. Beregovska, D. D. Zerbino, T. V. Teslyuk, M. Ya. Seneta, Universal controller for the distributed management in the adaptive smart home systems, Ukrainian Journal of Information Technology 6(2) (2024) 64–73. doi:10.23939/ujit2024.02.064.

[21] I. Alkhwaja, M. Albugami, A. Alkhwaja, M. Alghamdi, H. Abahussain, F. Alfawaz, A. Almurayh, N. Min-Allah, Password cracking with brute force algorithm and dictionary attack using parallel programming, Applied Sciences 13(10) (2023) 5979. doi:10.3390/app13105979.

[22] W. Y. Saputra, S. Sugiarti, H. Junianto, D. Suhartono, Password strength study using the ZX-CVBN algorithm and brute-force time estimation to strengthen cybersecurity, Jurnal Pilar Nusa Mandiri 21(1) (2025) 52–59. doi:10.33480/pilar.v21i1.6119.

[23] L. Golightly, P. Modesti, R. Garcia, V. Chang, Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN, Cyber Security and Applications (2023) 100015. doi:10.1016/j.csa.2023.100015.

[24] O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots, Botnet detection technique for corporate area network, in Proceedings of the the IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS) IEEE, 2013, pp. 363-368.