Zero Trust Access Control with Context-Aware and Behavior-Based Continuous Authentication for Smart Homes

Giovanni R. da Silva[1], **Daniel F. Macedo**[2], **Aldri L. dos Santos**[1,2]

[1]NR2 - Federal University of Paraná, Brazil

[2]Dept. of Computer Science - Federal University of Minas Gerais, Brazil

{grsilva,aldri}@inf.ufpr.br, damacedo@dcc.ufmg.br

Abstract. *Generally, approaches to build the security of Smart Home Systems (SHS) require big amount of data to implement Access Control and Intrusion Detection Systems, being a vulnerability to inhabitants privacy. Additionally, most works rely on cloud computing or resources in the cloud to perform security tasks, what can be exploited by attackers. This work presents ZASH (Zero-Aware Smart Home System), an Access Control for SHS. ZASH uses Continuous Authentication with Zero Trust, supported by real-time context and activity information, enabled by edge computing and Markov Chain, to prevent and mitigate impersonation attacks that aim to invade inhabitants privacy. An experimental evaluation demonstrated the system capability to dynamically adapt to new inhabitants behaviors withal blocking impersonation attacks.*

1. Introduction

The Internet of Things (IoT) is an established paradigm, where many things are interconnected providing services to the people [Castro et al. 2019]. As things are being connected, many systems assumed social characteristics intrinsic to human problems. Thus, Cyber-Physical Social Systems (CPSS) are in emergence [Dong and Ansari 2020], evolving towards the Internet of Everything (IoE), which includes people, data and things in an unified process [de Matos et al. 2017]. IoE aims to connect anything that has Internet-connection, from people, to TVs, cars and sensors. CPSS and IoE are intimately linked by human factors that bring human knowledge, mental capacity, and sociocultural elements using things to mix the physical world with cyberspace.

Particularly Smart Homes settings have been growing rapidly and gaining popularity as there are more internet users, more importance for green energy, for security, and more smart gadgets available to more people [MarketsAndMarkets 2020]. Further, recently due to pandemic people mandatorily need to stay at home, increasing its importance in daily life. However, people still fear vulnerabilities that can expose intimate information and the physical security of themselves and their families. [Lee 2020] exposed that resistance of smart home adoption comes from vulnerabilities in technology, with fragile security systems, nonexistent or weak laws on digital crimes, unreliable service providers and the users themselves, due to misuse or inexperience.

Since Smart Home Systems (SHS) comprehend CPSS, physical security requirements in Smart Homes are as high as the informational. Impersonation attacks are common in SHS, allowing attackers to steal and use authentic identities, compromising the confidentiality, integrity, authenticity and non-repudiation [Mocrii et al. 2018,

Junior et al. 2020]. These attacks often involve social engineering, when victims are deceived and their credentials can be stolen, and eavesdropping, when attacker steal information on the fly [Humayed et al. 2017]. Then, attacker can use someone's credential to gain access to SHS and perform malicious actions to further monitor house activity or invade the property physically, violating the data and physical privacy of the resident.

Prior works on SHS security focus on Access Control (AC), Continuous Authentication (CA) and Intrusion Detection System (IDS). Many of them uses context-based decisions [Ashibani et al. 2019, Ghosh et al. 2019, Sikder et al. 2019] and behavior-based decisions [Ashibani and Mahmoud 2019, Ghosh et al. 2019, Amraoui et al. 2020]. Although few of them are suitable to endure impersonation attacks [Ashibani et al. 2019, Ghosh et al. 2019, Sikder et al. 2019], they present some issues as dependence of cloud computing and big amount of data to infer information about environment or user activities, leading to privacy breaches. Another issue is the dependence of end-user's devices as they have general purpose, being more exposed to exploitation. Furthermore, it can lead to vulnerability due to misuse, once inhabitants interact with SHS by many means, like smart assistants and devices themselves.

This work proposes ZASH (Zero-Aware Smart Home System), a system to provide AC for a SHS using CA with Zero Trust (ZT) in order to continuously verify users authenticity, powered by edge computing to dismiss unreliable service providers and capable of processing request originated from any means. The system is designed with user levels (e.g., admin, adult, child, visitor), device classes (e.g., critical, non-critical) and actions (e.g., view, control, manage) in order to mitigate possible undetected impersonation attack by contributing for devices isolation and actions differentiation. The CA process is constituted by three phases, being the first the verification among user, devices and actions using ontologies. The second phase is the verification of context information to check if they achieve expected trust for a requested action with a specific user level on a device class. The final phase consists of verifying whether the requested action makes sense considering a Markov Chain built on all previous activities. We have implemented ZASH and made an experimental evaluation to demonstrate the system capability to dynamically adapt to new inhabitants behaviors withal blocking impersonation attacks.

The remaining of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the ZASH system and its operation. Section 4 describes the evaluation methodology and results. Section 5 presents conclusion and future works.

2. Related Work

This section reviews techniques to prevent attacker to break into the system, detecting when the attacker has success in the invasion and mitigating presumable damage from this action. Access to a Smart Home must be restricted to its inhabitants, being authentication an important phase to identify who is requesting actions and information from the system. The Smart Home environment is dynamic regarding both devices and users. CA related works aim at identify users interacting with the system throughout the whole session rather than relying on a unique authentication in the session start, as described in [Nakayama et al. 2019, Al-Naji and Zagrouba 2020]. [Kuyucu et al. 2019] presents the state-of-the-art of strategies to protect security and privacy in Smart Homes.

[Ashibani et al. 2019] proposed a holistic approach in devices security for Smart Home. CA uses context information from multiple sources. The results demonstrated low latency overhead, effects on access decision-making by authentication-assigned weights and thresholds, and the capacity to handle multiple simultaneous requests without bottlenecking access to smart devices. One of the drawbacks consists in dependency on an external information (Google Calendar), which may decay reliability in the system, since this data can be altered or suppressed by an attacker. Moreover, it remains in dependency of end-user's devices (e.g., smartphone, tablet), considering the interaction inside a Smart Home can happen using other ways, like voice assistants and with the devices themselves. It decreases the likelihood of correct usage from users by enforcing them to use only the smartphone, making the Smart Home vulnerable to misuse. Besides, relying on the smartphone can lead to violations, since it has a general purpose and is carried in external environments, for work and leisure, being more exposed to exploitation.

[Ashibani and Mahmoud 2019] submitted a behavior-based CA model for Smart Home. The goal is to identify users from their apps usage in smartphones to assure the person trying to control the Smart Home is authentic. This is a suitable option for CA because app access patterns are continuously provided. However, users cannot depend on a smartphone to control their Smart Home. Moreover, it depends on track app access logs creating historical data that can be vulnerable to an attacker, exposing data privacy. [Amraoui et al. 2020] proposed another behavior-based CA model for controlling a Smart Home focused in making implicit re-authentication without user interaction. The dependency of high amount of data for reasonable accuracy is a drawback, as it exposes privacy by collecting and storing users behavior. The dependency of cloud computing is also a vulnerability that could be exploited by an attacker isolating the home from the Internet. Furthermore, it does not support different levels of security for users or devices, compromising the services differentiation and resources isolation. Thus, an impersonation attack would gain unrestricted access regardless the exploited user. In [Ghosh et al. 2019] it is presented a context-aware behavior-based authorization framework for Smart Homes. Among the work contributions is a modelling of expected belief on device access request based on user's past request patterns and on context of the request. It tries to block insider attacks by setting a threshold of confidence to execute a request. However, the dependence in historical data, as the other works focused in Anomaly Detection, pose an issue of cold start, as system would need to be trained before operating [Bakar et al. 2016].

[Sikder et al. 2019] exposed a context-aware Intrusion Detection System (IDS) that relies on a correlation between user activities and devices. It uses context of sensors, devices and controller devices. Although the work prove to cope with different home layouts and inhabitants number adaptively, it needs big amount of data to feed the anomaly detector module for training. Additionally, it depends on cloud computing, which brings a vulnerability for a Smart Home since an attacker could eavesdrop the network to steal data or cut the Internet means to isolate the house. [Dimitrakos et al. 2020] proposed UCON+ as an extended Usage Control (UCON) method to include trust evaluation to perform continuous authorization. UCON is a framework created to focus on Digital Rights Management (DRM) and is suitable to monitor continuously the execution rights of a subject over a resource. UCON+ take into account a Zero Trust Architecture (ZTA), which requires continuous verification of permissions and authorization policies. Zero Trust (ZT) is the term for an evolving set of cybersecurity paradigms that

move defenses from static, network-based perimeters to focus on users, assets, and resources [Scott W. Rose, Oliver Borchert, Stuart Mitchell 2020]. It assumes any entity can become malicious, regardless of its past authenticity and reliability, being the reason why trust should be evaluated continuously. Although UCON+ is highly customizable, it is not suitable for inexperienced users, as a SHS AC should be.

Prior works focused against insider attacks rely on continuous authentication either context-based or behavior-based with big amount of data and depending on cloud computing. Furthermore, they ignore other access ways commonly used nowadays, like voice assistants, house devices or the device itself. A robust solution is a hybrid context-aware behavior-based continuous authentication to prevent and mitigate impersonation attacks, considering many access ways and using edge computing. It should not depend on external network, be restricted to the local network and minimize message exchange by storing device states as they interact with the system, thus being less exposed to interceptions and preventing privacy violation. Furthermore, the mix of ontologies, instant context, and user behavior aims to improve accuracy with minimal performance overhead.

3. The ZASH System for Continuous Authentication in Smart Home

This section presents an overview about the environment characteristics, the concerns of the users and the solution for it, called ZASH (Zero-Aware Smart Home System). The Smart Homes counts on infrastructure network, enabled by Ethernet, Wi-Fi, Bluetooth, Zigbee, etc., and the connection between devices are point-to-point with TCP/IP protocol. In spite of Smart Home trend became popular in the last years, people are still highly concerned of their privacy, both related to data and physical world. SHS are not as secure as they should be regarding invaders, specially during impersonation attacks. Actual works struggle to prevent, detect and mitigate these attacks. Thus, this issue among others is avoiding a broader adoption of Smart Homes. The objective of this work is to focus on the prevention and mitigation of impersonation attacks in a SHS.

Impersonation attacks in SHS consists in several steps and consequences. First, the attacker steal any of inhabitants credentials, usually the one with more privileges, using social engineering or eavesdropping attack. The social engineering can range from simple techniques as baiting and phishing to more advanced ones as reverse social engineering and advanced persistent threat [Krombholz et al. 2015]. Attackers can alternatively benefit from a man-in-the-middle attack to access the SHS. After that, they can monitor the house activity from inside to plan a physical invasion, steal more privileged information and even control the home devices to compromise inhabitants safety.

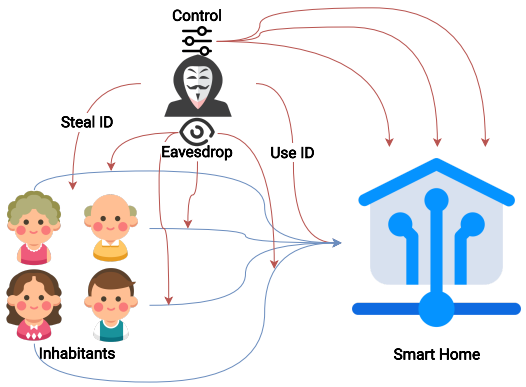


Figure 1. Impersonation attack in SHS

A SHS typically embodies several heterogeneous devices, denoted as $D = \{d_1, \ldots, d_n\}$, where n is the number of devices. These devices are connected to

the SHS in a Virtual Local Area Network (VLAN) and can be a smart object, like a smart TV; an actuator, like a smart curtain; or just a sensor, like for temperature. ZASH specifically will count on User Levels (UL), denoted as $UL = \{UL_1, \ldots, UL_m\}$ (e.g., visitor, child, adult, admin — from the least to the most privileged), where m is the number of UL. The Device Classes (DC), denoted as $DC = \{DC_1, \ldots, DC_k\}$ (e.g., critical, like smart locks, and non-critical, like smart light bulbs), where k is the number of DC. Furthermore, there are classes of Actions (A) that an user can perform on a device, denoted as $A = \{A_1, \ldots, A_p\}$ (e.g., manage, control, view), where p is the number of A.

The ZASH flow is detailed in Figure 2 and consists in inhabitants performing actions using intermediaries (e.g., smartphone, tablet, voice assistant) or interacting directly with device to control its states. The user request is received in the objects and then forwarded to the local server, which decides to authorize or not. ZASH can also require a proof of identity from users. The decision flows back to intermediaries (if used) and finally for devices to execute or not the action.

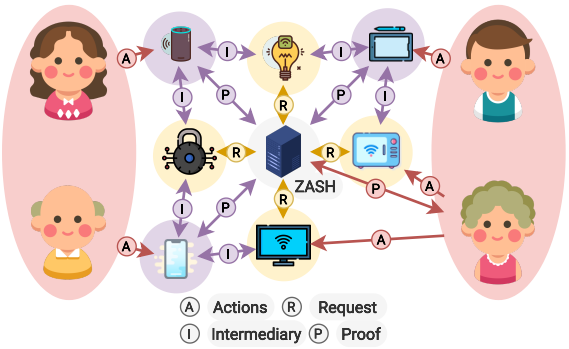


Figure 2. The ZASH flow

3.1. Architecture

The ZASH architecture contains three modules, named *Behavior, Collection* and *Decision*. The Behavior module includes Configuration Assembler, used by root user to configure system behavior, and Notification Dispatcher, responsible for notifying users to trigger further actions from them. The Collection module comprises the Device Communicator, which includes all devices configuration from the SHS and is responsible for communicating with them, and Data Provider, that stores latest device states and process it to support decision. Finally, the core Decision module consists in the Activity Manager, that process activities in the SHS, the Context Manager, responsible for building instant context information, the Ontology Manager, which includes the rules for the authorization, and the Authorization Controller, in charge of using all provided information to accept or reject a request.

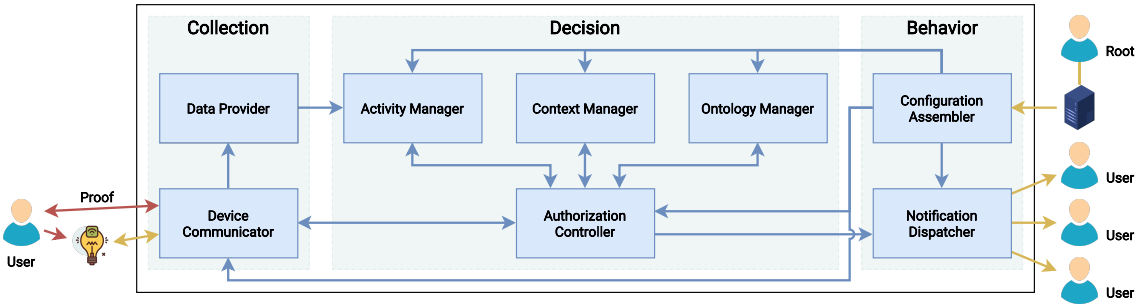
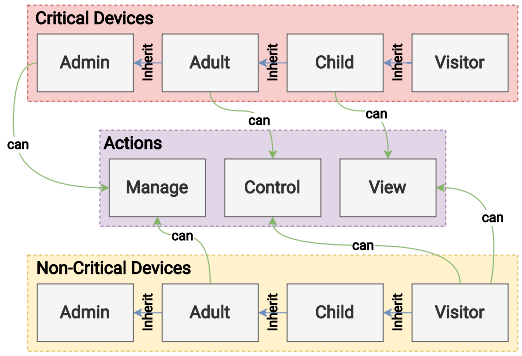


Figure 3. The ZASH architecture

Ontology Manager: ZASH imposes barriers to attackers using stolen credentials by counting on different UL, DC and A to guarantee the isolation of devices and differentiation of actions. Root can use Configuration Assembler to manage devices from Device

Communicator by adding new, altering and deleting existing. Each device included in the system must be classified with a DC and this will imply on its security level. The Ontology Manager intersect A on a DC from a specific UL to verify ontologies defined by root through the Configuration Assembler. Ontology based modelling is the best technique to represent knowledge through formalisms [de Matos et al. 2017]. An instance for the rules of valid ontologies is represented in Figure 4, where each UL_i for a specific DC_j can perform a set of A, named Capabilities (Cap_{ij}) (Equation 1), and UL_i inherits all Cap_{i-1j} from the lower UL_{i-1} (Equation 2). The Ontology Manager is mainly responsible to deal with easily detectable attacks and to keep the system safety and security, together with Context Manager, while the Activity Manager is starting.



$$Cap_{ij} \subseteq A,\ \forall i = 1..4, \forall j = 1..2 \quad (1)$$

$$Cap_i \supseteq Cap_{i-1},\ \forall i = 2..4 \quad (2)$$

Figure 4. ZASH Ontologies

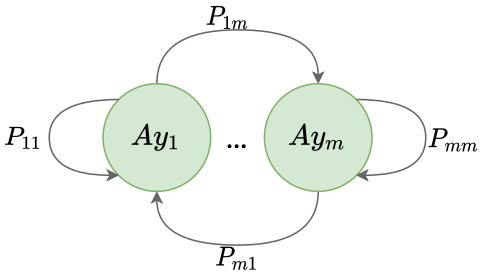
Context Manager: The system works with Security Level (s), determined as $s \in \mathbb{N} \mid s \leq 100$. Root configures initial s of each DC and UL. They also set additional s for each of the possible A. The Context Manager combines s of both DC (DC_s) with A (A_s) and UL (UL_s) with A (A_s) to check which one has the highest value. This value is used to verify whether instant context information achieves the required s. Instant context counts on factors, as instantiated in Equation 3 (e.g., request time, access way, localization, age, group). Each of these context factors receives a s that adds up to be compared. For example, the time can be classified in common or uncommon, if the request time occurrence probability is below a threshold. The access way can be requested (the device itself), house (e.g., voice assistant, tablet) or personal (smartphone). The localization can be divided in internal or external in reference to the house. The age can be linked to each user and can be adult, teen, kid. Group can be ranked in together or alone. [Ashibani et al. 2019] used a similar technique to compare security level access to devices threshold with confidence level collected from context and demonstrated low performance overhead, flexible access control and high scalability. The sum (X_s) of the context factors (C) is represented in the Equation 4 and the subsequent logic to accept request is expressed in Equation 5. ZASH prompts for a proof of identity if the context trust is below expected security level in order to assure user authenticity. This proof need to be a *Something You Are* challenge, which is less prone to falsification and can be fingerprint, face recognition, voice recognition, etc., as used in [Dimitrakos et al. 2020].

$$C = \{f_1, \ \dots, f_5\} \mid f_i \leq 30, \forall i = 1..5 \quad (3)$$

$$X_s = \sum_{f \in C} f,\ X_s \in \mathbb{N} \mid X_s \leq 100 \quad (4)$$

$$X_s \geq max(\{DC_s + A_s,\ UL_s + A_s\}) \quad (5)$$

Activity Manager: Activities in a SHS are sequential and follow a specific pattern. Activity Manager relies on a Markov Chain fed with every successful user request changing the balance of the transition matrix over time. Each of the UL has its respective Markov Chain, since they tend to present different patterns. It is associated with UL only in order to preserve individual users. The Activity Manager starts inactive in the authorization module as it depends on some requests to be processed to create a reliable transition matrix. The cold start problem is mitigated by the other modules. [Sikder et al. 2020] proves that Markov Chain is the best technique to detect anomaly requests in a context-aware environment and [Sikder et al. 2019] uses the technique achieving high accuracy in a SHS with low performance overhead, making it suitable for edge computing. Each activity (Ay_i) carries information about every device state (DS_i) in that moment, which is restricted to binary 0 or 1 data. Ay_i is represented in Equation 6, where n is the number of devices in the SHS. The transition model is represented in Figure 5, where m is the number of all possible states Ay (Equation 7) and P_{ij} is the probability of going to state j at time $t+1$ from state i at time t as shown in Equation 8. Considering each DS_i is binary, the value of $m = 2^n$. Similarly to the Context Manager, ZASH requires a proof of identity if the transition probability between two states is below a threshold.



$$Ay_i = \{DS_1, \ \dots, DS_n\}, \ n \in \mathbb{N} \tag{6}$$

$$Ay = \{Ay_1, \ \dots, Ay_m\}, \ m \in \mathbb{N} \tag{7}$$

$$P_{ij} = Pr(Ay_j \mid Ay_i), \ \forall 1 \leq i, j \leq m \tag{8}$$

Figure 5. ZASH Markov Chain

Other Components: Authorization Controller receives action request from Device Communicator and checks for Ontology Manager, Context Manager and Activity Manager in this sequence. The request must pass the three requirements in order to be accepted and if it fails in one of them, the subsequent ones are not even checked. If the same user has more than a defined number of rejected requests in a defined interval, Authorization Controller blocks they and send it to Notification Dispatcher, which will disseminate this information for all users. Device Communicator is responsible for requesting user proof of identity when needed. It also receives request from devices and sends changed state to the Data Provider and request to the Authorization Controller. All devices must be modified to redirect actions to ZASH and wait for approval. Data Provider keeps all current and last device states to feed Activity Manager. Finally, root user configures ZASH through Configuration Assembler by managing devices, setting ontologies and specifying thresholds for user blockage and notification with the number of rejected requests and the verified interval. This module also contains all users and root user can manage by adding new ones, altering their UL or deleting them.

3.2. Operation

ZASH operates in a local dedicated machine, communicates only inside a VLAN protected by firewall, which includes only devices and the local server, and is not con-

nected to the Internet. It can be only configured through a device inside VLAN with a recently provided proof of identity from a root user. Root user is not the same as admin UL, as it does not participate in the ZASH authorization schema and is used to log in the dedicated machine only. Algorithm 1 describes the steps followed by each device when an action is performed on active devices or state is changed in passive devices. The Device forms $Request$ information and sends to ZASH (*l*.3-4), then it receives answer from ZASH and perform action if authorized (*l*.6-7). ZASH starts operating after the initial configuration by the root using Configuration Assembler, which assigns UL for each user, DC for each device and ontologies rules. Root also configures s for each C, DC, UL and A. The number of rejected requests allowed by user before blocking and the considered time interval need to be set by root too. The build interval is also configured by the root to tell ZASH when the building of the Markov Chain and the time probabilities are reliable to start operating. As it can be a cumbersome task, alternatively the system should come with at least two predefined profiles, named hard, with more strict rules as default, and soft, with less strict parameters. After the initial setup, ZASH is ready to receive requests, which are automatically redirected from devices using any suitable technology like Zigbee or Wi-Fi to the central gateway, where the system works.

Algorithm 1 Device awaits ZASH authorization

```
1: ZashIP ← XXX.XXX.X.X
2: procedure ListenStateChange                          ▷ Triggered when device detects state change
3:     Request ← Device, User, Context, Action
4:     send Request to ZashIP                           ▷ sends request for authorization
5: procedure ListenAnswer(Request)
6:     if Request.Authorized is True then
7:         perform Request.Action
```

Algorithm 2 Device communication and data provision

```
1: DeviceCommunicator
2: function ExplicitAuthentication(User)
3:     Proof ← FindProof(Request.User.Id, Request.Context.AccessWay)   ▷ Find proof for user with access way
4:     if Proof not found then
5:         Proof = InputProof()                         ▷ Wait for user proof of identity
6:         if Proof not matches User then
7:             return False                             ▷ Deny action
8:         else
9:             store Proof                              ▷ Store proof to be used for the next T time
10:    return True                                      ▷ Grant action
11: function ListenRequest(Request, CurrentDate)
12:    ClearProofs(CurrentDate)                         ▷ Clear stored proofs obtained more than T time ago
13:    DataProvider.UpdateCurrentState(Request)         ▷ Update current state with requested action
14:    Result ← True
15:    if Device.IsActive then
16:        Result ← AuthorizationControl.AuthorizeRequest(Request, CurrentDate, ExplicitAuthentication)
17:    if Result is True then
18:        Request.Authorized ← True
19:        DataProvider.UpdateLastState()               ▷ Update last state to be former current state if granted
20:    send Request to Request.Device.IP                ▷ Send answer back to device
21: DataProvider
22: procedure UpdateCurrentState(Request)
23:    CurrentState ← Copy(LastState)
24:    CurrentState_{Request.Device.Id} ← InvertState(CurrentState_{Request.Device.Id})
25: procedure UpdateLastState
26:    LastState ← CurrentState
```

The system assumes that any user can interact with any device in the SHS by any way (e.g., directly, smartphone, tablet), but the requested action will be authorized

or not by ZASH. Algorithm 2 describes that ZASH receives request from device, where Device Communicator updates the current device state in Data Provider (*l*.13) and then asks Authorization Controller (*l*.16) for active devices. Finally, it sets current state as last state if change was granted (*l*.19) and returns the result to device in order to execute or ignore requested action (*l*.20). Device Communicator is also responsible for managing proof of identity by collecting from the user when needed (*l*.5), storing the proofs (*l*.9) and clearing those over T time (*l*.12). The proof is stored for the user and access way, so user is not bothered to prove identity every time using the same access way.

Algorithm 3 Authorization control for user requesting action on device

```
1:  AuthorizationControl
2:  function AUTHORIZEREQUEST(Request, CurrentDate, ExplicitAuthentication)        ▷ Triggered every request
3:      ClearUsers(CurrentDate)                                  ▷ Clear rejects out of interval from all users
4:      if Request.User.IsBlocked then
5:          return False                                                              ▷ Deny action
6:      if not VerifyOntology(Request) or
7:          not VerifyContext(Request, CurrentDate, ExplicitAuthentication) or
8:          not VerifyActivity(Request, CurrentDate, ExplicitAuthentication) then
9:          Request.User.Rejected ← Request.User.Rejected ∪ {CurrentDate}          ▷ Register rejection
10:         if |Request.User.Rejected| > ConfigurationAssembler.BlockThreshold then
11:             Request.User.IsBlocked ← True                    ▷ Block user if above threshold
12:             AlertAllUsers(Request.User)                      ▷ Alert all users about blockage
13:         return False                                                              ▷ Deny action
14:     return True                                                                   ▷ Grant action
15: OntologyManager
16: function VERIFYONTOLOGY(Request)                                  ▷ Check ontologies for request
17:     Cap ← FindCap(Request.User.UserLevel, Request.Device.DeviceClass)   ▷ Find capabilities for UL in DC
18:     if Request.Action in Cap then
19:         return True                                                               ▷ Grant action
20:     else
21:         return False                                                              ▷ Deny action
22: ContextManager
23: function VERIFYCONTEXT(Request, CurrentDate, ExplicitAuthentication)   ▷ Check context for request
24:     CalculateTime(Request, CurrentDate)          ▷ Calculate if request time is common or uncommon
25:     CheckBuilding(CurrentDate)                   ▷ Check if time probability building is over
26:     if IsTimeBuilding then
27:         Request.Context.Time ← COMMOM            ▷ Time is always the highest trust while still building
28:                                                  ▷ Calculate security level for A in DC
29:     ExpectedDevice ← Request.Device.DeviceClass.SecurityLevel + Request.Action.SecurityLevel
30:                                                  ▷ Calculate security level for A in UL
31:     ExpectedUser ← Request.User.UserLevel.SecurityLevel + Request.Action.SecurityLevel
32:     CalculatedTrust ← CalculateTrust(Request.Context, Request.User)   ▷ Calculate context security level
33:     if min(CalculatedTrust, 100) < min(max(ExpectedDevice, ExpectedUser), 100) then
34:         if not ExplicitAuthentication(Request.User) then                  ▷ Require proof of identity
35:             return False                                                          ▷ Deny action
36:     return True                                                               ▷ Grant action
37: ActivityManager
38: function VERIFYACTIVITY(Request, CurrentDate, ExplicitAuthentication)   ▷ Check activities for request
39:     CheckBuilding(CurrentDate)                   ▷ Check if markov chain building is over
40:     CurrentState ← DataProvider.CurrentState
41:     LastState← DataProvider.LastState
42:     if not IsMarkovBuilding then                                  ▷ Always pass if still building
43:         if MarkovChain.GetProbability(CurrentState, LastState) < P then   ▷ Check if transition above threshold
44:             if not ExplicitAuthentication(Request.User) then          ▷ Require proof of identity
45:                 return False                                              ▷ Deny action
46:     MarkovChain.BuildTransition(CurrentState, LastState)          ▷ Build the transition matrix
47:     return True                                                               ▷ Grant action
```

Algorithm 3 details the authorization control that, when more than a defined number of rejected requests occurs for the same user in a defined time interval, ZASH blocks the user and notifies all users in the system through their registered personal devices (*l*.10-12). It can be only unblocked through the local dedicated machine by the root user. The

Authorization Controller verifies firstly with the Ontology Manager (*l*.6), which in turn checks for the rules set by the root user (*l*.16-21). Secondly, it verifies with the Context Manager (*l*.7) to certify the request has the expected context trust based on the combination of security level from $DC_s + A_s$ and $UL_s + A_s$ (*l*.23-36). Finally, it feeds the Activity Manager (*l*.8) in order to build the Markov Chain transition matrix and then check if the change from last state to current state is above a threshold P (*l*.38-49). If requested action fails in the Context Manager (*l*.34) or in the Activity Manager (*l*.44), Device Communicator will ask for a proof of identity from the user in order to validate the action. A valid proof will permit the system to learn that the action was correct and the time and states transition probabilities will be recalculated considering the new valid possibility.

4. Experiment

This section presents the evaluation methodology for analyzing the efficacy of the ZASH system. We implemented and executed the ZASH system with *Python 3.9.5* in *Zorin OS 15.3* using *SIMADL (Simulated Activities of Daily Living Dataset)* [Alshammari et al. 2018] and can be found at GitHub[1]. The technology was chosen for its portability, ease of file manipulation and ease of algorithms writing, providing quick development and evaluation. The graphics were designed using *Draw.io*. The dataset, found at GitHub[2], was chosen for simulating activities of daily life and presenting good amount of data in a well structured form with 29 devices, as presented in Table 1. The variant *d6_2m_0tm.csv* (167,211 lines of records after cleaning duplicates, 60 days, 1 line per second) was used for presenting the biggest amount of records. Devices states are represented with 0 or 1, being, for example, a door with state 0 closed and 1 opened.

Table 1. Configured devices

Device	Device Class	Room	Type
Wardrobe	Non-critical	Bedroom	Active
TV	Non-critical	Living Room	Active
Oven	Critical	Kitchen	Active
Office Light	Critical	Office	Active
Office Door Lock	Critical	Office	Active
Office Door	Non-critical	Office	Active
Office Carpet	Non-critical	Office	Passive
Office Sensor	Non-critical	Office	Passive
Main Door Lock	Critical	House	Active
Main Door	Non-critical	House	Active
Living Light	Non-critical	Living Room	Active
Living Carpet	Non-critical	Living Room	Passive
Kitchen Light	Non-critical	Kitchen	Active
Kitchen Door Lock	Critical	Kitchen	Active
Kitchen Door	Non-critical	Kitchen	Active
Kitchen Carpet	Non-critical	Kitchen	Passive
Hallway Light	Non-critical	House	Active
Fridge Sensor	Critical	Kitchen	Active
Couch Sensor	Non-critical	Living Room	Passive
Bedroom Light	Non-critical	Bedroom	Active
Bedroom Door Lock	Critical	Bedroom	Active
Bedroom Door	Non-critical	Bedroom	Active
Bedroom Carpet	Non-critical	Bedroom	Passive
Bed Table Lamp	Non-critical	Bedroom	Active
Bed Sensor	Non-critical	Bedroom	Passive
Bathroom Light	Non-critical	Bathroom	Active
Bathroom Door Lock	Critical	Bathroom	Active
Bathroom Door	Non-critical	Bathroom	Active
Bathroom Carpet	Non-critical	Bathroom	Passive

Figure 6. Home design [Alshammari et al. 2018]

Table 2. Configured users

	User 1	User 2	User 3	User 4	User 5
UL	Admin	Adult	Child	Child	Visitor
Age	Adult	Adult	Teen	Kid	Adult

[1] https://github.com/giovannirosa/zash
[2] https://openshs.github.io/datasets/

Table 3. Configured security layers

	User Level				Action			Device Class	
Categories	Admin	Adult	Child	Visitor	Manage	Control	View	Critical	Non-critical
Security Level	70	50	30	0	40	20	0	30	0

Table 4. Configured context factors

	Time		Localization		Age			Group		Access Way		
Cat.	Common	Uncommon	Internal	External	Adult	Teen	Kid	Together	Alone	Requested	House	Personal
SL	20	10	30	10	30	20	10	10	0	30	20	10

There are 8 passive and 21 active devices in the dataset, being a state change in any active one considered as an user request. The execution flows sequentially through the records, simulating 2 months. The configured users can be found at Table 2. The home design, exhibited in Figure 6, includes bedroom, living room, kitchen, bathroom, office and hallway. The ontologies were configured as shown in Figure 4. The configured security layers were as seen in Table 3 and the context factors were set as seen in Table 4. The values were inspired by [Ashibani et al. 2019] and adjusted by experimentation to balance security and user experience with less proofs asked. Three threats were considered in the evaluation: 1) valid users trying to access devices with no permission (violation to ontology rules); 2) valid users trying to access devices in abnormal conditions (violation to context trust); and 3) illegitimate users trying to access devices with stolen credentials (violation to activity pattern). The applied dataset influences the results as the activities might be different, leading to distinct probabilities in the Markov Chain.

4.1. Metrics

The metrics, shown in Table 5, assess the main characteristics proposed by ZASH. Ontology Fail (OF) measure the percentage of requests that failed in Ontology Manager and required proof of identity to verify authenticity. Similarly, Context Fail (CF) and Activity Fail (AF) evaluates the same as OF, but for the Context Manager and the Activity Manager, respectively. The Requests Granted (RG) count the ratio of granted requests and the Requests Denied (RD) count the ratio of denied requests.

Table 5. Evaluation metrics

Description	Equation
Ontology Fail (OF) is the percentage of requests that failed in ontology evaluation (OF_{req}) out of total requests (T_{req}).	$OF = \frac{OF_{req}}{T_{req}} \times 100$
Context Fail (CF) is the percentage of requests that failed in context evaluation (CF_{req}) out of total requests (T_{req}).	$CF = \frac{CF_{req}}{T_{req}} \times 100$
Activity Fail (AF) is the percentage of requests that failed in activity evaluation (AF_{req}) out of total requests (T_{req}).	$AF = \frac{AF_{req}}{T_{req}} \times 100$
Requests Granted (RG) is the percentage of granted requests (G_{req}) out of total requests (T_{req}).	$RG = \frac{G_{req}}{T_{req}} \times 100$
Requests Denied (RD) is the percentage of denied requests (D_{req}) out of total requests (T_{req}).	$RD = \frac{D_{req}}{T_{req}} \times 100$

4.2. Preliminary Results

The simulation provided three cases to validate the ZASH correctness, as exhibited in Figure 7. The first one (threat 1) is the request that happened at 2016-02-01

08:01:48, when a child user requested to control the oven (critical device) using a voice assistant (house access way). ZASH denied the action, because in the configured ontologies, shown in Figure 4, a child does not have the capability to control a critical device. The second case (threat 2) occurred at 2016-03-28 08:04:10, when an admin user requested to manage the main door lock (critical device) interacting directly with it. The expected trust is 100 because the admin UL_s (70) add to the manage A_s (40), considering the maximum expected trust is 100. However, the calculated trust from the instant context is 90, being requested access way (30), external localization (10), common time (20), alone (0) and adult (30). Therefore ZASH asks for a proof of identity from the user and since it is valid, the action is granted. The third case (threat 3) is an impersonation attack that happened at 2016-03-05 19:30:26, where the attacker stole a personal device from an adult user and tried to control the main door lock (critical device) from an external location. The expected trust was 70 from adult UL_s (50) plus control A_s (20). The calculated trust was also 70 from personal access way (10), external localization (10), common time (20), alone (0) and adult (30). Although the Context Manager granted the action, the Activity Manager denied it, because the P_{ij} equal to 6.82% was below the threshold 10%. The results presented that every impersonation attack is prevented by ZASH, assuming the attacker cannot provide a valid proof of identity belonging to the stolen user.

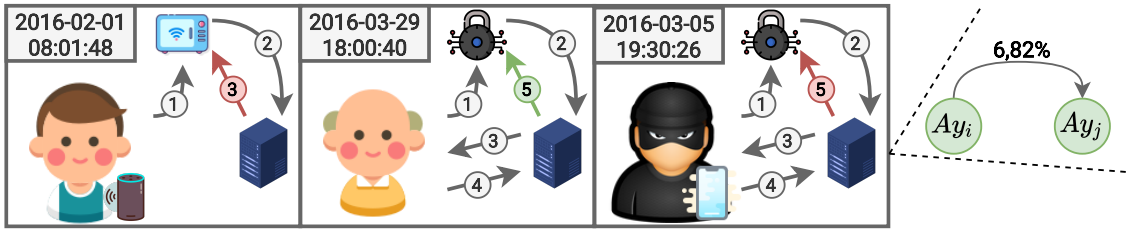


Figure 7. Ensuring ontologies rules, activities consistency and context trust

Table 6. Collected metrics from variations in context and user

Variations	OF	CF	AF	Blocked	RG	RD	Proofs Asked
-	0%	1.77%	0.28%	No	100%	0%	37
External	0%	100%	0.28%	No	100%	0%	216
Adult	0%	0%	0.28%	No	100%	0%	6
Child/Visitor	0.16%	0%	0%	2016-02-01 08:05:52	0.28%	99.72%	0

The simulation was also evaluated as a whole by using the metrics. Table 6 displays the collected metrics from variations in context and user. Simulations were executed considering a baseline for all requests with the context as personal access way, internal localization and alone; from the admin UL, being an adult; and the A always as control. It provided an OF of 0% as all requests passed in Ontology Manager as expected, since the admin user should have the highest privileges. It produced a CF of 1.77% due to uncommon request times. All non-blocked variations supplied 0.28% for AF, since it depends on activities only. Still in the baseline, it gave RG of 100% as all the 37 proofs asked were valid. The baseline context was executed with external localization, providing similar results except for the CF, which was 100% as the context trust decayed, and the 216 proofs asked following the need for authenticity verification. Then, the baseline varied to adult UL, giving CF of 0%, considering adult UL requires less trust than admin

UL, and requesting only 6 proofs for the Activity Manager fails. Finally, the baseline ran for child UL and visitor UL, which resulted in the same metrics for both UL, because they were blocked by failing 4 times within 24 hours in Ontology Manager at 2016-02-01 08:05:52. ZASH just authorized 0.28% of requests before blocking user and denying all the other 99.72%. This evaluation assumed attacker could not provide a valid proof of identity. It is a complex task for the attacker to fake the proof since it is categorized as *Something You Are* [Lal et al. 2015]. For now ZASH is evaluating context and behavior, being a multi factor authorization with *Somewhere You Are* from localization, *Something You Have* from access way and *Something You Do* by evaluating activities. However, it could also mix these categories in the proof of identity phase in order to improve security, by also asking for a *Something You Know* challenge, for example.

5. Conclusion

This work presented ZASH to secure a SHS against impersonation attacks supported by Continuous Authentication and Context-Aware Access Control with Zero Trust paradigm. ZASH endures the Smart Home heterogeneity of technologies and also does not rely on any external service or cloud computing to provide the home security. It works in a local server that receives all requests from devices to ensure the ontologies rules, the context trust and the activities consistency. The system relies on multiple security layers with user levels, actions and device classes to mitigate any undetected impersonation attack. ZASH was implemented and evaluated to validate its capacity to prevent and mitigate impersonation attacks by using instant context information and adapting to inhabitants behavior continuously. As future works, we will simulate ZASH in a realistic home network to measure impact on latency and concurrency. The Markov Chain can be turned into a Hidden Markov Chain using the device states as transitions and the user activity as emissions, which could make the model more precise. Additionally, the activities threshold will be investigated to be dynamic depending on the request, similar as in the context. Ultimately, ZASH could include an auxiliary module to adjust its parameters automatically using an optimization algorithm on the fly. The system could be tested in a multi-user scenario and evaluate the user experience to verify its viability and acceptance.

Acknowledgment

We would like to acknowledge the support of the Brazilian Agency CNPq in the Universal Project, grant #436649/2018-7.

References

Al-Naji, F. H. and Zagrouba, R. (2020). A survey on continuous authentication methods in Internet of Things environment. *Computer Communications*, 163(August):109–133.

Alshammari, T., Alshammari, N., Sedky, M., and Howard, C. (2018). SIMADL: Simulated activities of daily living dataset. *Data*, 3(2):1–13.

Amraoui, N., Besrour, A., Ksantini, R., and Zouari, B. (2020). *Implicit and Continuous Authentication of Smart Home Users*, volume 926. Springer International Publishing.

Ashibani, Y., Kauling, D., and Mahmoud, Q. (2019). Design and Implementation of a Contextual-Based Continuous Authentication Framework for Smart Homes. *Applied System Innovation*, 2(1):4.

Ashibani, Y. and Mahmoud, Q. H. (2019). User authentication for smart home networks based on mobile apps usage. *Proceedings of Communications and Networks, ICCCN*, 2019-July:1–6.

Bakar, U. A. B. U. A., Ghayvat, H., Hasanm, S. F., and Mukhopadhyay, S. C. (2016). *Activity and Anomaly Detection in Smart Home: A Survey*, pages 191–220. Springer International Publishing, Cham.

Castro, T. O., Caitité, V. G., Macedo, D. F., and dos Santos, A. L. (2019). CASA-IoT: Scalable and context-aware IoT access control supporting multiple users. *International Journal of Network Management*, 29(5):1–18.

de Matos, E., Amaral, L. A., and Hessel, F. (2017). Context-aware systems: Technologies and challenges in internet of everything environments. In *Beyond the Internet of Things*, pages 1–25. Springer.

Dimitrakos, T., Dilshener, T., Kravtsov, A., La Marra, A., Martinelli, F., Rizos, A., Rosett, A., and Saracino, A. (2020). Trust aware continuous authorization for zero trust in consumer internet of things. *Proceedings - 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020*, pages 1801–1812.

Dong, M. and Ansari, N. (2020). Guest editorial: Special section on cyber-physical social systems—integrating human into computing. *IEEE Trans. on Emerging Topics in Computing*, 8(1):4–5.

Ghosh, N., Chandra, S., Sachidananda, V., and Elovici, Y. (2019). SoftAuthZ: A Context-Aware, Behavior-Based Authorization Framework for Home IoT. *IEEE Internet of Things Journal*, 6(6):10773–10785.

Humayed, A., Lin, J., Li, F., and Luo, B. (2017). Cyber-Physical Systems Security - A Survey. *IEEE Internet of Things Journal*, 4(6):1802–1831.

Junior, C. P., Santos, A., and Nogueira, M. (2020). Detecting fdi attack on dense iot network with distributed filtering collaboration and consensus. In *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6.

Krombholz, K., Hobel, H., Huber, M., and Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, 22:113–122. Special Issue on Security of Information and Networks.

Kuyucu, M. K., Bahtiyar, , and İnce, G. (2019). Security and privacy in the smart home: A survey of issues and mitigation strategies. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 113–118.

Lal, N. A., Prasad, S., and Farik, M. (2015). A Review Of Authentication Methods. *International Journal of Scientific Technology Research*, 4(8):246–249.

Lee, H. (2020). Home IoT resistance: Extended privacy and vulnerability perspective. *Telematics and Informatics*, 49:101377.

MarketsAndMarkets (2020). Smart home market with covid-19 impact analysis by product (lighting control, security access control, hvac control, entertainment, home healthcare), software services (proactive, behavioural), and region - global forecast to 2025. `https://www.marketsandmarkets.com/Market-Reports/smart-homes-and-assisted-living-advanced-technologie-and-global-market-121.html`.

Mocrii, D., Chen, Y., and Musilek, P. (2018). IoT-based smart homes: A review of system architecture, software, communications, privacy and security. *Internet of Things*, 1-2:81–98.

Nakayama, F., Lenz, P., Banou, S., Nogueira, M., Santos, A., Chowdhury, K. R., and Lacuesta, R. (2019). A Continuous User Authentication System Based on Galvanic Coupling Communication for s-Health. *Wireless Communications and Mobile Computing*, 2019.

Scott W. Rose, Oliver Borchert, Stuart Mitchell, S. C. (2020). Zero Trust Architecture - NIST Special Publication 800-207. *Nist*, page 49.

Sikder, A. K., Aksu, H., and Uluagac, A. S. (2020). A context-aware framework for detecting sensor-based threats on smart devices. *IEEE Trans. on Mobile Computing*, 19(2):245–261.

Sikder, A. K., Babun, L., Aksu, H., and Uluagac, A. S. (2019). AEGIS: A Context-aware Security Framework for Smart Home Systems. *ACM Int. Conference Proceeding Series*, pages 28–41.