THESIS

TRUST BASED ACCESS CONTROL AND ITS ADMINISTRATION FOR SMART IOT DEVICES

Submitted by

Zarin Tasnim Promi

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2024

Master's Committee:

Advisor: Indrajit Ray

Indrakshi Ray
Leo R Vijayasarathy

ABSTRACT

TRUST BASED ACCESS CONTROL AND ITS ADMINISTRATION FOR SMART IOT DEVICES

In today's interconnected world, the security of Internet of Things (IoT) devices is paramount, given the types of smart devices ranging from household appliances to industrial machinery. The continuous, long-term operation of IoT networks increases vulnerability to attacks, and the limited capabilities of IoT devices render standard security measures less effective. Traditional cryptographic methods used for establishing trust through identification and authentication face challenges in IoT contexts due to their computational demands and scalability concerns. Additionally, administration for these intricate networks can become extensive, and the presence of malicious or unskilled human operators can further increase security risks. To combat these issues, adopting a "Zero Trust - Never Trust, Always Verify" strategy is vital in IoT environments. Our approach involves creating an access control model based on device trust, which continuously evaluates the trustworthiness of connected devices and dynamically modifies their access rights according to their trust levels. This enables adaptive and fine-grained access control in IoT settings. Furthermore, we propose a trust-based administrative framework that enables configuration policy, enhancing security and administration efficiency in IoT networks. Similarly to the access control model, this approach will continuously monitor the operator behavior and adjust their operational privileges based on their actions.

ACKNOWLEDGEMENTS

I am sincerely grateful to Dr. Indrajit Ray, my adviser, for all of his insight and advice. This thesis would not have been completed without his guidance. I would like to thank Dr. Indrakshi Ray and Dr. Leo Vijayasarathy for agreeing to serve on my thesis committee.

This work was partially supported by the US National Science Foundation under Grant No. 1822118 and 2226232, the member partners of the NSF IUCRC Center for Cyber Security Analytics and Automation – Statnett, AMI, NewPush, Cyber Risk Research, NIST and ARL – the State of Colorado (grant SB 18-086) and the authors' institutions. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or other organizations or agencies.

DEDICATION

to Mona

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) devices have become a common feature in both home and organizational networks, offering a diverse array of services. According to the Statista survey [1], the number of connected devices to the IoT will be more than 50 billion by 2024 and 75 billion by 2025. Most IoT devices are designed for plug-and-play functionality, allowing any user to easily integrate them into existing networks. However, insecure or improperly configured IoT devices pose substantial security risks, as evidenced by their history of being vulnerable to attacks [2].

Even though access control models are well researched in computer science, their application in the Internet of Things (IoT) presents multiple challenges. Traditional security and access control techniques often prove inadequate in the diverse IoT ecosystem due to the wide variety of devices and their respective protocols. For example, traditional Public Key Infrastructure (PKI) based access control techniques are too computationally expensive for most IoT devices. Similarly, Role-Based Access Control (RBAC) systems can become difficult to manage by large-scale IoT networks. Moreover, the administration of these systems is often complex and labor-intensive, requiring substantial resources to oversee and maintain security. Therefore, a major challenge within the IoT ecosystem is to develop a fine-grained access control and administration model that can effectively address these issues.

1.2 Limitations of Prior Work

Prior solutions have one or both of the following limitations:

- Traditional Access control models have various shortcomings when applied to IoT networks. As a result most of the IoT frameworks enforce coarse-grained access control policies [3]. For instance, Nest Thermostat1 grants access to all the capabilities of a smart device or to

none. On top of this, existing access control models for IoT ecosystem differ widely due to the nature of devices, protocols, interfaces, etc, lacking a unified model which can provide dynamic access control irrespective of system structure and device type.

- Research into administrative models is still in an early stage, given the recent wave of digitalization. This is particularly true for IoT systems, where the complexity and novelty of the ecosystem make studies even difficult. Although well-established access control models are being adapted into administrative frameworks, these adaptations often overlook the specific needs of IoT systems. While there has been some research on the administration of Role-Based Access Control (RBAC) in IoT, other access control models remain largely unexplored. This gap highlights a critical need for further exploration into how these models can be effectively administered within the unique context of IoT environments.

In our work, we attempt to address the identified shortcomings of the previous work and provide a fine-grained access control model and its administration.

1.3 Overview of Proposed Approach

In this work we outline a trust model and trust based access control- TrustBAC model. The trust model has multiple components which continuously monitor devices to evaluate their characteristics. These quantifiable characteristics determine the reliability of the device. The TrustBAC model dynamically adjust authorizations based on this trustworthiness of devices. Trust values are continuously updated based on devices's current actions. Moreover, when access control using the TrustBAC model, a cost trade-off is considered. We discuss the cost of consequences and cost of effectuating permissions and outline the formal definition cost model on a set of accesses. We further implement a software suite which is based on this TrustBAC model. This micro-service based software system emulates all the components from TrustBAC. This tool enforces dynamic access control when deployed in an IoT network. Finally we propose a administration framework for the TrustBAC model. This administration model is based on the same notion of trustworthiness.

System operators are allowed to configure system policies depending on their trust level. This trust level is assessed from several admin attributes such as designation, type, audit trail, etc.

1.4 Key Contributions

Our key contributions can be summarized as:

- We outline a trust based access control (TrustBAC) model where trust is computed from quantifiable characteristics. This trust is used to enforce dynamic fine-grained access control.
- We have developed a software system which simulates the TrustBAC model.
- We introduce the notion of trade off in terms of cost of consequence and cost of effectuating access permissions in the context of the TrustBAC model.
- We propose a trust based administration model. This model possess the similar intuitions as the TrustBAC model and enables configuration policy, enhancing security and administration efficiency in IoT networks.

1.5 Thesis Organization

In chapter 2, we discuss multiple related works categorized by access control and administration. Chapter 3 contains a few key technical backgrounds such as the device fingerprinting process and trust indicators in detail. In chapter 4 we outline the trust based access control(TrustBAC) model. In chapter 5, we describe the administration in TrustBAC.

Chapter 2

Related Work

We discuss several related works in two sections. In the first section, we detail research works on access control and in the second section we describe administration of access control models.

2.1 Access Control

In traditional scenario, cryptographic mechanisms are used to ensure security in devices. The OCF Security Specification for the IoT ecosystem [4], states that if a device is authenticated while onboarding can be trusted throughout the session. This might not be true in general, a device can get malicious any time after the onboarding process. Moreover, IoT devices have limited computational capabilities. Cryptographic techniques are computationally expensive, making it almost impossible to implement them in IoT devices. On top of this, public key certificates are expensive to buy and manufacturers often choose not to buy them. Considering all these factors, we can say that cryptographic techniques are not an option to ensure security in IoT ecosystem.

By approximately the 1990s, with the development of the Internet and the increasing large-scale applications of information systems in enterprises, the traditional models of access control (DAC, MAC, and its extension models) encountered difficulties addressing complex application layer access requirements. To solve this role-based Access Control (RBAC) was first introduced by Ferraiolo et al. in [5], [6] to address the limitations of the discretionary access control model (DAC). Sandhu et al. [7] present four reference models that offer a systematic method to understand the RBAC model. Their framework distinguishes between the administration of RBAC and its application in controlling access, and categorizes how RBAC is implemented across different systems. Subsequent modifications led to the proposal of the NIST standard for RBAC in [8]. This standard outlines an RBAC reference model that defines the scope of features included in the standard and introduces terminologies to aid in specification. It further details the system and administrative functional specifications, which define the functional requirements for administra-

tive operations and system-level functionalities. RBAC is a well-adopted access control model in enterprise settings [9].

Attribute-Based Access Control (ABAC) has recently gained significant attention due to its flexibility [10], [11], [12], [13], [14]. Moreover, it has proven to have the ability to represent different access control models [12], as well as application in Internet of Things domains(IoT) [15].

Bezawada et al. [16] describe the implementation of Attribute-Based Access Control (ABAC) for home Internet of Things (IoT) environments to address specific security challenges caused by the heterogeneous nature and dynamic interactions of IoT devices. It discusses the applicability of the NIST Next Generation Access Control (NGAC) specification for defining and enforcing ABAC. The authors provided a comprehensive framework that includes device-type agnostic policies, the management of attributes, and the enforcement architecture necessary for maintaining security in dynamically changing home IoT settings, thereby enhancing the overall security posture without compromising the operational efficiency and usability of IoT networks.

There has been a great deal of research that uses trust as a tool of access control. Ray et al [17] proposed a significant advancement in role-based access control (RBAC) systems, particularly for open and decentralized environments. Traditional RBAC models often struggle in dynamic contexts where user identities and roles cannot be predetermined. TrustBAC addresses these limitations by integrating trust levels into the RBAC framework, thereby extending the model to evaluate access based on trust metrics rather than static role assignments. The model considers three trust factors, user's past behavior, knowledge about the user and recommendation provided by others about the user. This collectively improves the robustness of access control decisions. This approach allows for dynamic assignment of trust levels to roles, which are then linked to specific permissions.

Dimitrakos et al [18] offers a policy language and architecture for authorization mechanism in consumer IoT settings by incorporating trust-level evaluations. This model integrates Attribute-Based Access Control (ABAC) with a Trust-Level-Evaluation Engine (TLEE). This enables dynamic adjustments of trust assessments and continuous policy re-evaluation based on mutable at-

tributes and environmental changes. This approach provides adaptability and responsiveness to varying trust conditions, strengthening security in highly dynamic IoT environments. However, it introduces challenges related to the complexity of real-time trust management and potential privacy concerns due to continuous data monitoring. This may lead to performance bottlenecks and increased data protection risks.

Mahalle et al. [19] introduce a novel framework to manage trust within the dynamic decentralized IoT environment. The authors propose the Fuzzy Trust-Based Access Control (FTBAC) model. This model utilizes fuzzy logic to quantify and manage trust among IoT devices based on three key parameters: Experience, Knowledge, and Recommendation. The model addresses the challenge of traditional access control systems that fail to accommodate the unpredictable nature of IoT device interactions. FTBAC scheme simulation results show that it can be used to calculate fuzzy trust values for any number of devices which makes it more suitable for scalable IoT. However, the reliance on fuzzy logic may introduce difficulties in implementation and might require fine-tuning to balance between responsiveness and computational overhead effectively.

Ray et al. [20] formulate a trust model designed for the Internet of Things (IoT), which is crucial for implementing fine-grained access control in dynamic environments. The authors propose a trust framework that is not based on reputation. This model uses measurable properties of IoT devices to establish trust levels. The trust model works by calculating a trust score for each device based on a set of predefined indicators/properties and the trust history. This score decides the access permissions granted to a device, ensuring that only devices that meet a certain trust threshold can interact with sensitive or critical components of the system. By focusing on measurable behaviors and characteristics, the model aims to provide a more robust and adaptable security framework suitable for the dynamic nature of IoT environments.

2.2 Administration of Access Control

In RBAC, the administration of the user role, the permission role and the assignment of the role role is a critical and challenging task. This is often referred as administrative RBAC or ARBAC.

The assignment and revocation of user roles, role permissions, and other related tasks are managed by the RBAC administration. Many approaches have been proposed for ARBAC [21], [22], [23], [24], [25]. In URA97 [21], user-role assignment is determined based on prerequisite roles of the target user. Similarly, in URA99 [22], it is determined based on the current membership of the target users in the mobile and / or immobile roles. Uni-ARBAC [25] model for administering user-role and permission-role relations by combining many of the existing administrative principles and novel concepts. Uni-ARBAC assigns tasks to roles and aggregates permissions into tasks rather than granting separate permissions. For administrative purposes, Uni-ARBAC decouples users and tasks from roles following the decoupling principle of ARBAC02 [26]. However, it also introduces administrative overhead and complexity, as it requires continuous monitoring and adjustment of roles and permissions based on evolving organizational policies and responsibilities.

AARBAC [27] provide a method for incorporating attribute-based controls into Administrative RBAC (ARBAC) to increase its versatility. Their work focuses on the management of permission- and user-role assignments in RBAC systems, which are often handled through explicit role-driven techniques. In order to design more dynamic access control policies, two proposed models—AURA (Attribute-Based User-Role Assignment) and ARPA (Attribute-Based Permission-Role Assignment)—use permissions and the attributes of entities, such as normal and administrator users. Compared to conventional ARBAC approaches, which rely on preset property sets, this attribute-centric approach enables a more granular administration of permissions and user responsibilities.

Chapter 3

Technical Background

In this chapter we discuss the technical knowledge required to understand the following chapters. We discuss two key elements for trust based frameworks : device unique identification and trust indicators among other elements and intuitions that operate within the model.

3.1 Device Unique Identification

Numerous security issues can be solved by properly identifying and authenticating devices, allowing administrators to effectively monitor and apply the right access controls to each device. Strong identification requires each device to have a unique and unforgeable identity linked to it. Fingerprinting can link a device to its natural behavior within the network. A malicious device differs from this set behavior patten. For example, a malicious device may scan the network to acquire information about other devices. However, fingerprinting IoT devices is challenging due to the robust variety of devices, protocols, and interfaces.

However, IoT device fingerprinting research is still in developing stage as the IoT industry is still blooming. General device fingerprinting explored in [28], [29], [30] describes several techniques ranging from packet header features to physical features such as clock-skews. The techniques of finger printing on wireless devices have been addressed in [31], [32], [33], [34], [35]. These studies investigated how different implementations of a common protocol such as SIP can help identify device types among similar devices. However, given the vast array of protocols used by IoT devices, conducting such detailed analysis for each protocol. Fingerprinting of devices based on physical layer characteristics has been extensively studied, as indicated by [36], [37], [38], [39], [40]. These studies primarily analyze the physical attributes of general wireless devices to establish unique fingerprints. However, the effectiveness and applicability of these techniques for IoT devices remain uncertain and are currently an open area of research.

Radhakrishnan et al. [38] outlines a method for identifying device types such as smartphones, laptops, and tablet PCs, based on the inter-arrival times of packets to pinpoint features specific to applications like Skype. However, the lower traffic generation of most IoT devices means that collecting such data would be both time-consuming and labor-intensive.

Miettinen et al. [41] introduce IoT Sentinel, a framework designed to detect device fingerprints and improve the security of IoT networks. This framework utilizes supervised machine learning techniques to fingerprint devices at the time of their initial network registration. This method leverages packet header characteristics to pinpoint specific device types. A notable drawback of this approach is its vulnerability to packet spoofing, as packet headers can be easily altered. Similarly, Siby et al. present IoTScanner [42], a system designed to passively monitor network traffic at the link layer, analyzing it through the use of frame header data within specified observation windows. The primary focus of this architecture is to identify distinct devices and their activities based on the patterns of traffic observed during these windows. However, a limitation of this method is that two devices of the same type might be mistakenly identified as different types due to variations in the traffic they generate within the observation period.

IoTSense developed by Bezawada et al. [43] , which is specifically aimed at monitoring the active state of devices, analyzing their behavior on the network using similar machine learning approaches. While IoTSense is effective at recognizing devices of known types, it struggles with identifying new, unfamiliar device types that connect to the network. It often incorrectly classifies these unknown devices as one of the device types it has previously learned.

A more recent works by Maxwel et al. [44] leverages a sliding-window packet analysis and synthetic data generation using Adversarial Autoencoders (AAE) to enhance device-type fingerprinting from small datasets. This approach addresses the challenges of accurately identifying IoT devices with limited data by applying machine learning to short network trace samples. The technique improves data coverage and fingerprint accuracy through the strategic use of synthetic data to supplement training datasets. Furthermore, the model incorporates distributed deep learning,

enabling collaborative training across multiple organizations without direct data exchange, thus preserving data confidentiality.

For our work, we adopt the fingerprinting process described in Ray et al. [20]. They utilized behavioral fingerprinting to uniquely identify IoT devices based on network traffic patterns, without relying on expensive cryptographic methods. They proposed an unsupervised learning model based exclusively on network traffic. This fingerprinting is done when a device is bootstrapped on the network. The learning model looks into 4 types of common messages: DNS Query, DNS Response, SSLCert, Client Hello. The effectiveness of this model is assessed using measures of probability and confidence associated with each device identification. For trust based access control, uniquely identifying a device is mandatory. Moreover, in trust level computation, how strongly we can identify a device plays a crucial factor. Our intuition is that the degree of trust associated with a device is dependent on how confidently we can identify a device. For this reason, we incorporate the confidence parameter from the device fingerprinting process in trust level computations. Further details on this are elaborated in the subsequent section 4.2.

3.2 Trust Indicators

A trust indicator is defined as some measurable characteristic of an entity, in our case devices and human operators, which can significantly influence the trust level of that entity. They can be of two types:

- Internal Trust Indicator:

  An internal trust indicator is property that is intrinsic to the entity or property of that entity that emerges from within the IoT ecosystem.

- External Trust Indicator:

  Similarly, an external trust indicator is property that is extrinsic to the entity or property of that entity that emerges from within the IoT ecosystem.

Trust indicators need to be defined and classified according to the system's specifications and functionality. We provide a list of several device trust indicators in the following, organized by their functionality rather than their significance. This list is curated from [20], [45] and [46]. System designers have the flexibility to determine the relative importance of these categories and the elements within the categories.

Trust indicators are scored on a point basis, evaluated, for example, on a Likert scale. When a trust indicator positively contributes to, enhances, affects, or supports the security of the IoT system, a positive point is assigned. If it negatively impacts the system, a negative point is assigned. The trust indicators listed here are not all-inclusive; they have been identified through an preliminary analysis of the smart home IoT environment and distributed cable systems. For other systems, or even variations of similar systems, system architects should conduct tailored studies to find specific trust indicators relevant to their needs. A comprehensive explanation of the identified indicators is detailed below:

- **Device Category:** An evaluation decision is made based on what impact a compromise of a device in that category would have on the system. Device subcategories may include without limitations:

    1. End-point networking: Examples include IoT hubs, switches, routers, Wi-Fi access points, etc.
    2. Infrastructure networking: CMTS, routers and switches, servers for network operations, etc.
    3. Productivity: Examples include tablets, smartphones, smart printers, etc.
    4. Entertainment: Examples include smart TV, speakers, music systems, media hubs etc.
    5. Physical safety and security: Examples include motion detectors, security cameras, fire alarms, smoke detector, doorbell etc.
    6. Home comfort and convenience: Examples include thermostat, bulb, electric switch etc.

7. Utility: Examples include refrigerator, washer, dryer etc.
8. Medical and health care devices: Examples include smart watches, wearable health monitors, automated insulin pumps, etc.
9. Voice assistants and controllers: Examples include devices such as Echo show, Google Home etc.

- **Device capability:** The device capability is an important indicator owing to the fact that the higher the device's computational capabilities the more damage can probably be done by exploiting the devices. The device capability may be measured with respect to:
  1. Processor type, e.g., micro-controller, general purpose, etc.
  2. Available RAM
  3. Storage capability
- **Device certification:** Device certification is done through a trusted CA and is not expired. Evaluation depends on the trust put in the CA. Any time a new certificate is issued, it will trigger re-evaluation of trust indicators and trust value.
- **Network connection type:** This indicator may consider if a device is wired or wireless, and the wireless type such as Wi-Fi, Bluetooth, Zigbee, etc.
- **Physical location of device:** This may consider whether the device is:
  1. Unprotected access: No physical barrier to physically accessing the device.
  2. Partially protected: Limited physical barrier to device access.
  3. Highly protected: inside a confined area and access limited to authorized personnel. For example, devices inside houses are only accessible to homeowners.
- **Vulnerability density:** The notion of vulnerability density presents the number of vulnerabilities contained in the device firm-ware. The vulnerability density can be annotated

as

$$V_d = \frac{V}{S}$$

Where, S represents the software size, V represents the number of total vulnerabilities in the device software. The list of vulnerabilities can be sourced from well known vulnerability databases such as NVD/CVE.

- **Connection requirement:** Factors considered are:

  1. Device to device
  2. Device to cloud
  3. Application to cloud
  4. Application to device, etc.

- **Connection security:** This indicator may consider:

  1. Authenticated connection (Y/N)
  2. Encrypted connection (Y/N)
  3. Encryption type (e.g., TLS, DTLS, etc.)
  4. Encryption strength

- **Device manufacturer reputation** Manufacturer reputation can be an important indicator for trustworthiness. This notion of trust of a manufacturer may be derived from their public image, business dealings or years of experience. In real world, IoT devices from more reputable manufacturer tend to get purchased more than manufacturer having poor reputation or than a new manufacturer in market. For example, consumers will prefer the same IoT device from established brands like Amazon/Apple/Google over one from some "ABC" company. However, since this is largely based on public perception, it is important to approach such views with caution.

- **Connection behavior:** This indicator may consider one or two of the following factors about the connection:

    1. Device-initiated or outside-initiated
    2. Average frequency of connection with outside
    3. Average duration of connection
    4. Average bandwidth consumed
    5. Packet length deviation (expected vs observed)
    6. Frame rate for traffic

3.3 Definitions and Parameters

Key definitions and parameters is curated from related previous works [20], [45] and [46] as these notions are also valid for our work.

- **Access type-based trust:**

  Access type-based trust is represented by the expression $CM_i \overset{c}{\Rightarrow} D_j$, indicating that the trustor CM_i trusts the trustee D_j for a specific access type c. This model tailors trust according to the particular actions D_j is permitted to perform on CM_i's resources, which are defined by the access type. Trust levels are dynamically adjusted based on the context of each access request, ensuring that trust is granted precisely.

- **Degree of trust:**

  The degree of trust refers to the specific level to which a trustor CM_i trusts a trustee D_j. The degree of trust is evaluated periodically or on a trust trigger event. It can be represented by the expression: $V_t(CM_i \overset{c}{\Rightarrow} D_j) \in [0, 1]$. Here V_t is the degree or level of trust that the trustor has on the trustee for the access type c calculated at time t. A value 0 means that the trustor has no trust on the trustee. This may happen when there is a lack of information to calculate the trust level or the trust evaluation process returned 0. On the other hand, a value 1 means the trust evaluator completely trusts the entity.

- **Trust history:**

  The trust management system corresponding to a trustor maintains a historical record of trust values for the trustees with whom it has previously interacted. Additionally, this history contains a log of the trust indicator values that contributed to the trust levels over time. The trust history is important for monitoring the evolution of a device's trustworthiness and plays a significant role in assessing its current trustworthiness. Utilizing trust history to determine present trust levels serves two primary purposes: (a) In cases where trust indicators cannot be directly assessed due to various constraints, the trust history provides a basis for a reasonable evaluation of trustworthiness. (b) The trust history helps prevent sudden and arbitrary changes in the trust status of a device, ensuring that a device that was previously trusted does not become distrusted overnight.

- **Trust trigger:**

  A trust trigger is defined as an event that causes a change in the value of a trust indicator. Not every trust indicator is linked to a trigger; some indicators may be affected by multiple triggers or share a common trigger with others.

- **Trust indicator vector:**

  Every device is linked to an ordered set of size m consisting of trust indicators p_t = $[k_1, k_2, \ldots, k_j, \ldots, k_m]$. Each trust indicator k_j has a value $|k_j|$. The Device Monitoring System can evaluate these indicators; however, the measurement of some trust indicators is probabilistic and comes with a certain degree of confidence, whereas others can be measured with precision.

Chapter 4

Trust Based Access Control

In this chapter we discuss the trust-based access control model TrustBAC. Conventional access control models such as role-based access control are often inadequate for dynamic IoT networks as the identity of devices is not known in advance. In addition, the limited capabilities of IoT devices make it impossible to implement strong perimeter-centric security controls and defenses in IoT networks. Hence, adopting a ”Zero Trust - Never Trust, Always Verify” [47] strategy is vital in IoT environments. TrustBAC is such an access control model which continuously evaluates the trustworthiness of connected devices and dynamically modifies their access rights according to their trust levels. We outline the trust framework architecture that uses quantifiable properties of IoT devices to calculate device trust level, and the access control model that uses the underlying trust model. This chapter builds on ideas presented in [20], significantly enhanced by our original contributions.

4.1 Trust Model Architecture

We describe in this section the trust model adapted from the work [20] which can determine the trust value of IoT devices based on their quantifiable characteristics. The trust model continuously monitors for events that can change trust value, and on such an event calculates the degree to which a connected device can be trusted by measuring different properties, also known as trust indicators. The trust model at any given time enforces fine-grained access control relevant to that trust value.

The prerequisite for this trust model is that we identify each device uniquely based on device behavior estimated via network traffic features. This unforgeable identity associated with each device is used to manage their respective current trust value, trust history, and evaluated indicator parameters.

The key components of this model are:

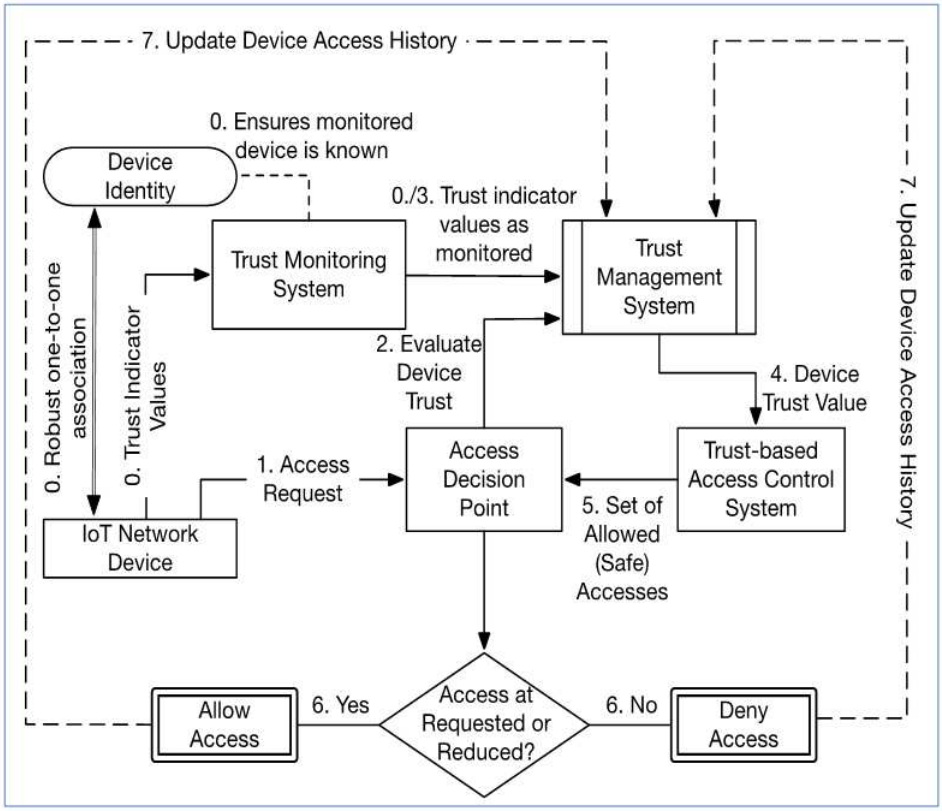

Figure 4.1: Trust Model Architecture.

- **Trust Monitoring System:**

  This component monitors trust triggers, evaluates trust indicator values and updates the Trust Management System. Another responsibility of the monitoring system is to identify any connecting device.

- **Trust Management System(TMS):**

  Trust Management System(TMS) is responsible for managing the trust indicators, evaluates trust level. TMS also resolves any trust query made to it.

- **Trust Based Access Control System(ACS):**

  This component is where the TrustBAC model employed and implemented. This system can return a set of accesses on a particular resource based on a given trust value of the device. We will discuss this component in detail in a later section 4.3.

- **Access Decision Point(ADP):** This component compares the allowed set of accesses with submitted request and allows the access or a set of accesses.

The Trust Monitoring System identifies a device when it connects to the network via suitable fingerprinting process and generates a unique id bound to each device. This system continuously monitors the devices in the system to enforce that only a known device exists in the system. In addition, it communicates the information from the identified device to the Trust Management System and evaluates the indicator values for the calculation of the trust level. These steps are marked with '0' in Fig. 4.1. When an IoT device makes an access request (arrow marked by "1") to the Access Decision Point, this component requests Trust Management System for the current trust value (arrow marked by '2'). In response the Trust Management System gets the device's latest trust indicators from the Monitoring System (arrow marked by "3"). The calculated trust value is then sent to the trust-based access control system (arrow marked with '4') and returns a set of allowable accesses (arrow marked by '5') based on the trust value and access policy. Finally, the Access Decision Point decides to allow or deny access (arrow marked with '6'). This decision is recorded in the Trust Management System (arrow marked by "7") to use it in further trust value computation.

4.2 Trust Evaluation Engine

In this section we describe the different factors and elements in play for trust value calculation.

- **Trust History Score:**

  The trust history score h_t at current time t_n is calculated from the trust value $V_{t_{n-1}}$ that was computed at time t_{n-1}. Let the time difference be, Δt = t_n- t_{n-1}. Then, the trust history score at time t_n is represented by h_t = is given by

$$h_t = V_{t_{n-1}} \times e^{-\left(\frac{(V_{t_{n-1}}-1)\Delta t}{2k}\right)}$$

Here $k \geq 1$, is a small integer referring to the rate of change of trust level and is system policy specific.

- **Trust Indicator Confidence Level:**

  The trust monitoring system continuously tracks the indicator values. Despite efforts to accurately measure trust indicators, there remains a degree of uncertainty regarding the precision of these values. The confidence in trust indicator value is a measure or percentage of accuracy of a given trust indicator value. To address these concerns, we have implemented a belief system based on subjective logic [48]. This system enables the adjustment of trust indicator values according to the perceived beliefs about the accuracy of the measurements.

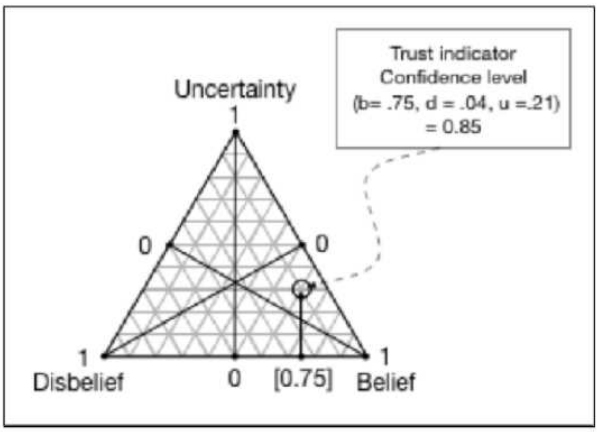

Figure 4.2: Trust Indicator Confidence Level Computation.

Each time a trust indicator k_j is assessed, the assertion "the measurement is accurate" is subjectively evaluated by linking it four values: (i) a belief $b_{k_j} \in [0, 1]$ that the proposition is true, (ii) a belief $d_{k_j} \in [0, 1]$ that the proposition is false, (iii) a belief $u_{k_j} \in [0, 1]$ representing uncertainty about the truth or falsity of the proposition, and (iv) an a priori probability $a_{k_j} \in [0, 1]$ concerning the truth of the proposition when no specific belief is held. Given the binary nature of this state space, a_{k_j} is set to 0.5, in line with the principles of subjective logic outlined in [48]. The relationship between b_j, d_j and u_j is such that b_j + d_j + u_j = 1. From these values, the confidence level of a measured trust indicator is calculated as o_j

= b_j + $a_j u_j$. It is assumed that as a trust indicator is evaluated, the corresponding values of belief, disbelief, and uncertainty are also provided. Fig. 4.2 illustrates an example of how confidence in a trust indicator value is computed.

- **Trust Indicator Score:**

  Each trust indicator is associated with an weight/importance. Trust indicator importance vector, $P_{\text{im}} = [p_1, p_2, \ldots, p_j, \ldots, p_m]$ where p_j annotates the importance of trust indicator k_j.

  Trust indicator score, denoted by $P_{\text{dev}} \in [0, 1]$ at time t_n is computed as a weighted average over the trust indicator vector values using the trust indicator importance vector, along with the corresponding confidence value.

  The normalized trust indicator single value function can be defined as,

$$f_j(k_j) = \left(\frac{|k_j|}{\max_j |k_j|} \right) \times \frac{100}{m}$$

  where $|k_j|$ is the absolute value of k_j, m is the total number of indicators. The next step is to sum these values together.

  Finally, the trust indicator score is calculated as,

$$P_{\text{dev}} = a_j = \frac{1}{m} f_j(k_j) o_j p_j \max_j \{ f_j(k_j) o_j p_j \}$$

- **Trust Value at current time:**

  The degree or level of trust that a trustor CM_i has on the trustee D_j at the current time t_n is expressed by the equaltion:

$$V_{t_n}(CM_i \overset{c}{\Rightarrow} D_j) = C_{\text{dev}}[\alpha P_{\text{dev}} + \beta h_t]$$

Here α, β are weights assigned to current trust indicator score P_{dev} and history score h_t. These weight values are system specific. In our work we add another factor the the calculation of trust, that is the indentity confidence C_{dev}. The weighted sum is multipied by C_{dev}. The trust model assumes that all devices are uniquely identifiable. The device fingerprinting method outlined in [20] provides probability and confidence as evaluation metric for an identified device. No fingerprinting model can ensure a probability or confidence level of 100%. Consequently, this parameter is further included into the trust level computations.

4.3 Trust Based Access Control(TrustBAC) Model

The trust model is employed in the TrustBAC model to maintain fine-grained access control in IoT networks. Unlike other access control models, this model incorporates trade-off factors in terms of cost: cost of effectuating permission and cost of consequences. This model provides a way to express access control policies with trade-off factors.

The key elements of this model are devices, permissions, access types, resources. Each device is linked to a trust history and a set of trust indicators to compute the trust level. The access control policy associates each access or set of accesses to a resource with a specific range of trust values that a device must meet to gain access to that resource. When a device requests access, it may be granted through one or more sets of permissions, each corresponding to different resources. Additionally, each permission is associated with both the cost of consequence and the cost of effectuation, which are part of the access control policy. When a device requests access to a resource, the TrustBAC model evaluates three factors:

1. Whether the device trust value is >= trust level needed for access?

2. What is the cost of consequences and cost of effectuating for a set of permission?

3. The set of permissions that effectuates the access request and at the same time minimizes the costs. To allow this, partial access can be allowed instead of complete denial.

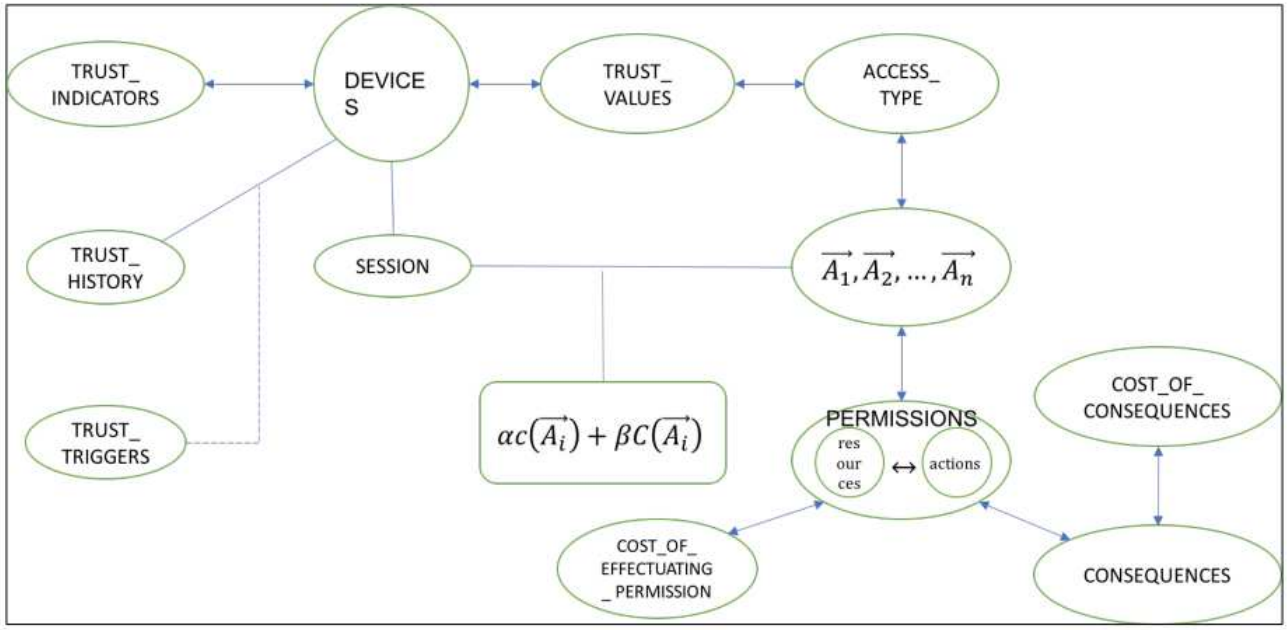

Figure 4.3: Trust Based Access Control Model.

Fig. 4.3 summarizes the trust based access control model, outlining the model elements and their interaction. Whenever a device makes an access request on a resource, its corresponding indicators and history are pulled to calculate the trust value. This trust value is then used to determine the allowable access set in particular to that resource according to the access control policy. For this set, the cost factors are defined further. Finally, a set of access is chosen such that it enables the request and minimizes the cost. The cost optimization function is described in the following section in detail.

4.4 Cost Function for the TrustBAC Model

The TrustBAC model considers trade off in terms of cost actors. There are two types of cost associated with allowing access: the cost of consequence and the cost of effectuating permissions. An access decision is made such as the cost is minimized. In this section, we outline the cost function that minimized both types of costs. Before further explaining the cost function, we describe some key concepts as follows:

- Consequence: A consequence is the resulting system state when a permission is applied.

- Cost of effectuating permission: The cost of effectuating a permission is represented by a real number from a discrete set of non-negative real numbers. This unit less cost amount is incurred by the system when it enables the permission. The total cost of effectuating the permission set is determined by a summation of cost for each permission.

- Cost of consequence: When a permission is effectuated, it triggers a change in the system's state, known as the consequence. This updated state may improve, degrade, or maintain the previous level of security. Regardless of the outcome, this transition incurs a cost associated with the new state. The cost of consequence, denoted by a real number from a finite, discrete set of non-negative real numbers that quantify this cost. This cost is unit less, providing a standardized measure of impact.

Algorithm 1 contains the formal definition of the cost function. The goal of this function is to define a set of access permissions that, when granted, will not exhaust the limited resources of the system when they are effectuated, and to keep the impact of consequences at a manageable level, ensuring that the system can sustain its operations effectively. Hence, the function has a minimizing constraint such that the weighted summation of both costs is minimized. The cost of consequence should be lower than the system threshold of impact. Beyond this threshold, the system will not be able to recover after negative impact. Similarly, the cost of effectuating depends on system resources. A threshold on resources maintains this.

4.5 TrustBAC Model Implementation

This section describes the implementation of the trust Based Access Control model through a software suite. Our primary contribution to the TrustBAC model is this software suite. This microservice-based software system simulates different components from the model and enforces dynamic access control when deployed in an IoT network. The development of this tool is currently in progress.

Algorithm 1 Formal Definition of the Cost Function ()

- Let $A = \{A_1, A_2, \ldots, A_n\}$ be the set of allowed accesses based on trust value.
- Let CC be the total cost of consequences for A.
- Let CP be the total cost of permissions for A.
- The objective is to minimize the total cost, $cf(\alpha CC + \beta CP \mid A)$, which is a weighted sum of CC and CP, subject to resource constraints on CP and a maximum threshold on CC.

$$\text{Minimize: } cf(\alpha CC + \beta CP \mid A)$$

Subject to:

$$g(CP) \leq C_1,$$
$$th(CC) \leq C_2,$$

Where:

- $cf(CC, CP \mid A)$ is the function representing the total cost that needs to be minimized, dependent on both the cost of consequence and the cost of permission.
- $g(CP)$ is the resource constraint function for the cost of permission, where C_1 is the maximum allowable resource limit for CP.
- $th(CC)$ is the maximum limit threshold for the cost of consequence, where C_2 is the maximum allowable impact limit for CC.
- α and β are system-specific weights assigned to the types of costs.

4.5.1 Technological Stack

The software suit is built with Spring Boot and MariaDB for the backend and React for the frontend. Spring Boot is a highly popular framework currently which simplifies the development of new applications through its convention over configuration approach. MariaDB includes performance enhancements in query optimization, indexing, and caching and new features that are not found in MySQL. For the frontend, React is a popular JavaScript library known for its component-based architecture and efficient updates using a virtual DOM.

Additional technologies are used for optimized and faster services. Redis Based caching helps to prevent reduces databases load for frequent queries, and to improve API latency. A message Queuing tool, RabbitMQ is used to push the updated state to other services and databases for asynchronous processing.

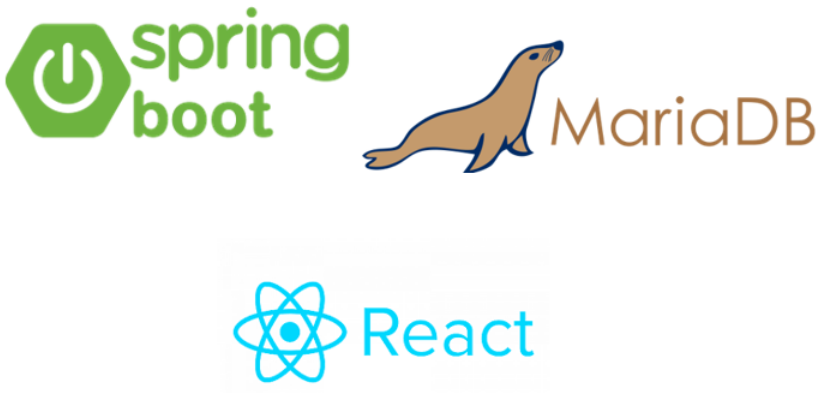

Figure 4.4: Technological Stack.

4.5.2 Microservices

Microservice design architecture is followed for the development. Each microservice simulates one component from the trust framework. The microservices are listed below:

- TMS:

  The Trust Management System is responsible for monitoring the indicators and trust values.

It has an immutable database to store all the trust related data. This component also implements the evaluation engine form the indicators. Moreover, all history analysis is done here to check any anomaly.

- ACS:

  This service mirrors the trust based access control system. All access policies are established here. The cost function is applied on the initial access set.

- ADP:

  This service receives the access request from IoT device and respectively calls TMS and ACS to handle the request. It gets the trust value from TMS. Then determines the possible access set from ACS by passing that trust value. Finally, ADP decides on a access set and allows it. ADP is responsible for pushing the access history to TMS for future evaluation.

The microservices communicates using RESTful (Representational State Transfer) APIs. These endpoints allow services to communicate over HTTP, exchanging data in formats like JSON (JavaScript Object Notation).

4.5.3 Security Measures

Multiple security and privacy preserving measures are taken within the software system. For example, data encryption, JWT (JSON Web Token) authentication, OAuth2 token.

- Data Encryption:

  All the stored data in databases are encrypted and hashed to enforce that none can read and alter the data. Communication among microservices is also preserved using this technique.

- JWT Authentication:

  JWT authentication is a great way to ensure that any previous requests are not copied and relayed again. This token contains an expire time, signature, headers, and other components which help efficiently transmitting secure information.

Figure 4.5: Security Features.

Chapter 5

Administration in TrustBAC

In this section, we present our administrative model for trust-based access control(TrustBAC). TrustBAC provides a fine-grained access control for IoT devices depending on how trust values of the devices are changing over time. However, like other access control systems, TrustBAC system is not static changes in access control state are inevitable. With the introduction of new resources in the system, new trust-based configurations need to be implemented. On top of this, existing configurations might change depending on system requirements and nature. Finally, there may be a need for prompt removal of malicious devices. These tasks demand the approval of human operators/admins, highlighting the need for trust-based administration.

5.1 Parameters and Definitions

In this subsection we detail all elements behind the formalism and intuition of the administrative model. We define the different elements as follows.

- Admin and Admin Category:

  Administrators are human operators authorized to carry out specific administrative tasks such as approval, rejection, and configuration updates. They can be classified into different types based on their roles and system needs. For instance, a system might designate root administrators (Ar), general administrators (A), or other specialized categories tailored to meet particular system requirements.

- Admin Operations:

  These operations, executable by an administrator, are required to handling configuration requests. For example, an admin has the authority to approve or reject mappings between trust values and access set policies.

- Admin Indicators:

  Similar to device trust indicators, administrators are associated with their own measurable characteristics. We have identified several indicators specific to an admin, though the list provided here is not exhaustive. Additional and different indicators may be defined based on system requirements.

  1. Admin designation
  2. Operational aptitude
  3. Incorrect action ratio

- Admin trust value:

  Admins are associated with a trust value calculated from their indicator value. This trust value can be calculated using the trust model discussed in previous chapter.

5.2 Break Glass: Handling Exception in TrustBAC

Break-glass administration refers to an emergency access protocol that allows for temporary elevation of access rights in critical situations, bypassing standard security procedures. This protocol is typically used to grant immediate, unrestricted access to systems or authority in the event of a failure, ensuring that necessary actions can be taken swiftly to resolve the situation. The use of break-glass procedures is closely monitored and audited to prevent abuse and ensure that such extraordinary access is used only when necessary.

For this system, general admins who will have traditional authentication methods to get into the system. This an event of system failure this authentication process may fail. Also to perform certain admin operations, they need access to configuration from database or other components. This might also be impossible when the system is critical. This is where we will use break glass administration.

We will incorporate a highly privileged admin role for this purpose. We will designate this admin as root. This root admin will be created from the configurations when the system is started.

This will be associated with multiple configurations at the application level. An example of configurations can be, but not limited to:

- Enable break glass
- Root credentials
- Time constraint for operations
- MFA
- Audit Period
- Refresh credentials and reboot period

5.3 Break Glass Administration in TrustBAC

In an emergency, the root admin will be activated through designated channels. The designated root admin must authenticate using their credentials, depending on whether the system currently allows that. Following authentication, the trust level of the root admin is verified to ensure it meets or exceeds a specified threshold. Once these criteria are satisfied, the root admin is activated, initiating a timer for the tasks they are set to undertake. Typically, each component has a single root admin. If a root admin is already active, no additional admins are allowed in the system. Formalized description of the steps involved are provided below:

Authentication

- Let S be the system state that indicates if break glass is currently allowed ($S = 1$ for allowed, $S = 0$ for not allowed).
- Define a function Auth(c) where c represents the credentials. This function returns `true` if the authentication is successful, and `false` otherwise.

5.3.1 Verification Process

- Let $V(c, S)$ be the verification process that encompasses Auth(c) and the system state S. This function returns true if both the authentication is successful and the system state allows for the necessary authentication method.

Trust Level Check

- Let T be the trust level of the designated root admin. Define a threshold value θ for the trust level.

- The trust level check can be represented as $T > \theta$. This condition must be true for the break glass process to continue.

Activation and Set Time Constraint

- Let R represent the activation of the root admin, where $R = 1$ indicates active, and $R = 0$ indicates inactive.

- Define R as a function combining the verification process and trust level check: $R = f(V, T > \theta)$, where f returns 1 (active) if both conditions are true, and 0 otherwise.

- Upon activation ($R = 1$), a timer constraint τ is started to track the duration of root admin operations. The root admin will have a certain time limit to perform the operations.

Summary of the Process

- Activation Request: A

- Authentication Verification: $V(c, S) = \text{Auth}(c) \wedge (S = 1)$

- Trust Level Check: $T > \theta$

- Root Admin Activation & Timer: $R = f(V, T > \theta)$, where f activates R and starts τ

5.4 Administrative TrustBAC Architecture

we provide a high-level overview of the framework with the components involved and their corresponding interactions. Fig. 5.1 shows such framework architecture. The framework consists of the trust management system which has two components - the trust model and trust evaluation engine, the Trust Based Authorization System, the Monitoring System that monitors admin actions and attributes, and the Configuration Administration Point which determines whether an admin can perform operations on certain configurations. The interactions between these components are indicated by directed and labeled arrows in Figure 1. The number against a label represents the ordering of interactions in an administrative decision.

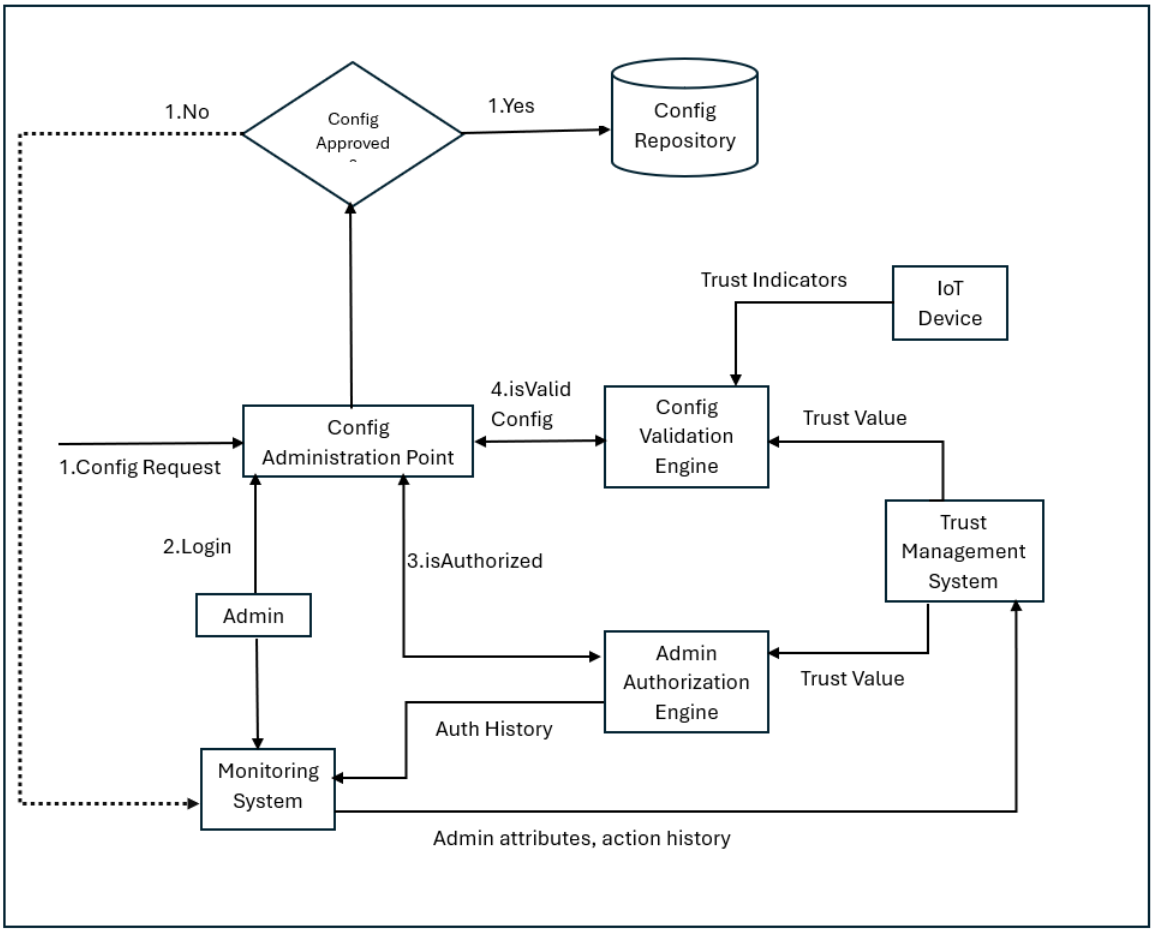

Figure 5.1: Administrative TrustBAC Architecture.

When a configuration request is received by the Config Administration Point (arrow labeled '1'), the admin intervention is required. When an admin logs into the system (arrow labeled "2"), he can see the existing pending request list. From there, the admin can perform specific operations (approve/reject/suspend) for a single configuration. The monitoring system continuously monitors admin attributes and action history, and pushes these values to the Trust Management system for trust level calculation. At the time of an admin operation, two checks are done. First, the admin authorization verification request is sent to the Admin Authorization Engine (arrow labeled '3'). The Admin Authorization Engine checks the requests for the trust level from TMS and checks it against a threshold trust value. The second check is performed for the integrity of the configuration request (arrow marked '4'). For example, in the case of an IoT device approve/suspend, the device trust values are checked against a threshold value. Another example of validation is to verify whether the trust value of the device with the access set mapping is complete and disjoint. Upon successful validation, the admin is approved to perform the respective operation. The Monitoring System logs all this information for auditing and future trust level calculation.

5.5 Example of Administrative TrustBAC

We present few general cases where administrative TrustBAC is applied. These cases and their steps may vary on the basis of systems and their requirements.

5.5.1 Case 1: Device Registration

When a device is first onboarded to the system, an administrative decision is made on whether to allow it further in the system or not. This is crucial for sensitive systems such as a nuclear power plant where random devices are not allowed to connect to the network. The trust monitoring system investigates the initial indicators of the device along with manufacturing information. We assume that the system is in normal state and that no emergency event is occurring that might activate break glass protocols. A general admin can approve the request to register a device on the network. According to the administrative TrustBAC model, two validations are done. First the model will

check whether the admin is authorized to perform this adminitrative task. The admin must have trust value above a system-specified threshold to approve/deny a new device. The second validation will verify whether the deivce's initial trust level satisfies a system threshold. Depending on system requirements, this step may include investigating manufacturer information, trust indicators, etc. Finally, the admin can approve or deny the device to connect to the network. These decisions are recorded in the system for auditing purposes, and can be a factor in calculating the level of admin trust.

5.5.2 Case 2: Trust-to-Access Configuration

The access control system contains a mapping of trust level to access permissions for each system resource. This mapping to trust level ranges is strictly system-specific and thoroughly done by human operators in general. Moreover, they should be incorporated in the system after verifications done by administrators. For this example we assume that the system is in normal state. The admin's trust level is verified first. After that both the system and admin can investigate the trust level to permissions mapping configurations. The admin can validate whether the configuration is satisfying system requirements. Further the mappings have to be complete and disjoint. After these conditions are met, the admin may approve (deny) the mapping configurations. This decision is also recorded in the systems.

5.5.3 Case 3: Trust Indicators Registration

Trust indicators are system specific. For any particular IoT system, they have to be identified, and their weights have to be determined. This job this generally done by administrators. Authorized admin can check whether a trust indicator is appropriate for the system, assign it weight, and finally approve/disapprove it to be incorporated in the system.

5.5.4 Case 4: Suspending Malicious Device

A device may get malicious at any moment. Then it is the responsibility of administrator's to further examine and suspend the malicious device. For this case we assume that the system

is in an emergency state and break glass protocols are activated. As a result, the root admin is currently logged into the system and no other admin is allowed to login or perform any task. As root administrators are verified when they log into the system, no further verifications will be made for any administrative task they are to perform. In this case, the root admin will investigate the nalicious device: device's current and past trust values, trust indicator values, and take a decision on whether to suspend the device or not based on his findings. Depending on system functionality, the root administrator may perform additional tasks to bring the system back to secure state.

Chapter 6

Conclusion and Future Work

In this work we provide a trust based robust IoT ecosystem. The trust framework continuously monitors the trust indicators of the devices to compute the trust level. Furthermore the trust based access control(TrustBAC) model controls access employing that trust level, considering trade-offs in terms of cost of consequence and cost of effectuating permissions. Finally, we propose a trust-based administrative model based on the same intuition of trust level, respective to administrators.

There are few directions that are still to be explored in this research work. For the administrative TrustBAC to work efficiently, we have to list administrator attributes extensively. Similar to device indicators, they are system specific. We have to curate the list of admin attributes for a variety of IoT spaces. At the same time, we also need to identify additional device indicators. Both device indicators and admin attributes are crucial for our model's improved performance. We plan to complete developing the software tool for TrustBAC and administrative TrustBAC. Then we can run experiments using this tool in both industry and home settings. This will help us look into performance issues in the tool, and finally investigate IoT systems where this tool is more applicable.

Bibliography

[1] Amir Rahmani, Suleyman Bayramov, and Behnam Kiani Kalejahi. Internet of things applications: Opportunities and threats. *Wireless Personal Communications*, 122, 01 2022.

[2] Brian Krebs. Mirai iot botnet co-authors plead guilty. https://krebsonsecurity.com/2017/12/mirai-iot-botnet-co-authors-plead-guilty/, 2017.

[3] Sowmya Ravidas, Alexios Lekidis, Federica Paci, and Nicola Zannone. Access control in internet-of-things: A survey. *Journal of Network and Computer Applications*, 144:79–101, 2019.

[4] Open Connectivity Foundation. OCF Security Specification Version 2. Technical specification, 2018.

[5] David Ferraiolo and D. Kuhn. Role-based access controls. 03 2009.

[6] David Ferraiolo, J. Cugini, and D. Kuhn. Role-based access control: features and motivations. 01 1995.

[7] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[8] David Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role based access control. *ACM Trans. Inf. Syst. Secur.*, 4:224–274, 08 2001.

[9] Alan C O'Connor and Ross J Loomis. 2010 economic analysis of role-based access control. *NIST, Gaithersburg, MD*, 20899, 2010.

[10] Richard Kuhn, Edward Coyne, and Timothy Weil. Adding attributes to role-based access control. 2010.

[11] Chung Tong Hu, David F Ferraiolo, David R Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (abac) definition and considerations [includes updates as of 02-25-2019]. Technical report, 2019.

[12] Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In *Data and Applications Security and Privacy XXVI: 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Paris, France, July 11-13, 2012. Proceedings 26*, pages 41–55. Springer, 2012.

[13] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. Attribute transformation for attribute-based access control. In *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*, pages 1–8, 2017.

[14] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. Abac with group attributes and attribute hierarchies utilizing the policy machine. In *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*, pages 17–28, 2017.

[15] Asma Alshehri and Ravi Sandhu. Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pages 530–538. IEEE, 2016.

[16] Bruhadeshwar Bezawada, Kyle Haefner, and Indrakshi Ray. Securing home iot environments with attribute-based access control. In *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*, ABAC'18, page 43–53, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Sudip Chakraborty and Indrajit Ray. Trustbac: integrating trust relationships into the rbac model for access control in open systems. SACMAT '06, page 49–58, New York, NY, USA, 2006. Association for Computing Machinery.

[18] Theo Dimitrakos, Tezcan Dilshener, Alexander Kravtsov, Antonio La Marra, Fabio Martinelli, Athanasios Rizos, Alessandro Rosetti, and Andrea Saracino. Trust aware continuous

authorization for zero trust in consumer internet of things. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1801–1812, 2020.

[19] Parikshit N. Mahalle, Pravin A. Thakre, Neeli Rashmi Prasad, and Ramjee Prasad. A fuzzy approach to trust based access control in internet of things. In *Wireless VITAE 2013*, pages 1–5, 2013.

[20] Indrajit Ray, Diptendu M. Kar, Jordan Peterson, and Steve Goeringer. Device identity and trust in iot-sphere forsaking cryptography. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 204–213, 2019.

[21] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The arbac97 model for role-based administration of roles. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):105–135, 1999.

[22] Ravi Sandhu and Qamar Munawer. The arbac99 model for administration of roles. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pages 229–238. IEEE, 1999.

[23] Sejong Oh and Ravi Sandhu. A model for role administration using organization structure. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 155–162, 2002.

[24] Ninghui Li and Ziqing Mao. Administration in role-based access control. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 127–138, 2007.

[25] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. Uni-arbac: A unified administrative model for role-based access control. In *Information Security: 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016. Proceedings 19*, pages 218–230. Springer, 2016.

[26] Sejong Oh and Ravi Sandhu. A model for role administration using organization structure. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 155–162, 2002.

[27] Jiwan Ninglekhu and Ram Krishnan. Aarbac: Attribute-based administration of role-based access control. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 126–135, 2017.

[28] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The DET curve in assessment of detection task performance. In *Proc. 5th European Conference on Speech Communication and Technology (Eurospeech 1997)*, pages 1895–1898, 1997.

[29] Richard Lippmann, David Fried, Keith Piwowarski, and William Streilein. Passive operating system identification from tcp/ip packet headers. In *Workshop on Data Mining for Computer Security*, volume 40. Citeseer, 2003.

[30] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[31] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, J Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *USENIX Security Symposium*, volume 3, pages 16–89, 2006.

[32] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, page 99–110, New York, NY, USA, 2007. Association for Computing Machinery.

[33] Jérôme François, Humberto Abdelnur, Radu State, and Olivier Festor. Automated behavioral fingerprinting. In Engin Kirda, Somesh Jha, and Davide Balzarotti, editors, *Recent Advances in Intrusion Detection*, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[34] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. Fingerprinting mobile devices using personalized configurations. *Proceedings on Privacy Enhancing Technologies*, 2016.

[35] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the third ACM conference on Wireless network security*, pages 169–174, 2010.

[36] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 116–127, 2008.

[37] Suman Jana and Sneha Kumar Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 104–115, 2008.

[38] Sakthi Vignesh Radhakrishnan, A Selcuk Uluagac, and Raheem Beyah. Gtid: A technique for physical device and device type fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 12(5):519–532, 2014.

[39] David Formby, Preethi Srinivasan, Andrew M Leonard, Jonathan D Rogers, and Raheem A Beyah. Who's in control of your control system? device fingerprinting for cyber-physical systems. In *NDSS*, 2016.

[40] Tom Van Goethem, Wout Scheepers, Davy Preuveneers, and Wouter Joosen. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In *Engineering Secure Software and Systems: 8th International Symposium, ESSoS 2016, London, UK, April 6–8, 2016. Proceedings 8*, pages 106–121. Springer, 2016.

[41] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in

iot. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 2177–2184. IEEE, 2017.

[42] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. Iotscanner: Detecting and classifying privacy threats in iot neighborhoods. arXiv preprint arXiv:1701.05007, January 2017.

[43] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. Behavioral fingerprinting of iot devices. In *Proceedings of the 2018 workshop on attacks and solutions in hardware security*, pages 41–50, 2018.

[44] Maxwel Bar-on, Bruhadeshwar Bezawada, Indrakshi Ray, and Indrajit Ray. A small world–privacy preserving iot device-type fingerprinting with small datasets. In Mohamed Mosbah, Florence Sèdes, Nadia Tawbi, Toufik Ahmed, Nora Boulahia-Cuppens, and Joaquin Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 104–122, Cham, 2024. Springer Nature Switzerland.

[45] Indrajit Ray and Steven J. Goeringer. Systems and methods for distributed trust model and framework, August 2021. U.S. Patent No. 11108557.

[46] Indrajit Ray and Steven J. Goeringer. Systems and methods for distributed trust model and framework, July 2023. U.S. Patent No. 11695558.

[47] Alper Kerman. Zero trust cybersecurity:'never trust, always verify'. *NIST Blog*, 2020.

[48] Audun Jøsang. *Principles of Subjective Logic*, pages 83–94. Springer International Publishing, Cham, 2016.

[49] IONOS editorial team. What is microservice architecture? 07 2023.

[50] Jochen Nickel. Mastering identity and access management with microsoft azure. page 84, 07 2018.

[51] Introduction to json web tokens. 07 2018.

[52] Chris Sevilleja. The anatomy of a json web token. 2015.

Appendix A

Microservice

In software engineering, a microservice architecture is an architectural pattern that arranges an application as a collection of loosely coupled, fine-grained services, communicating through lightweight protocols. One of its goals is that teams can develop and deploy their services independently of others. This is achieved by reducing several dependencies in the code base, allowing developers to evolve their services with limited restrictions from users and to hide additional complexity from users [49]. As a consequence, organizations are able to develop software with fast growth and size, as well as use off-the-shelf services more easily. Communication requirements are reduced. These benefits come at a cost to maintaining the decoupling. So, a microservice architecture can be a good choice only if the application is too complex to manage as a monolith. Interfaces should be carefully designed and treated as a public API. One technique that is used is having multiple interfaces on the same service, or multiple versions of the same service, so as to not disrupt existing users of the code.

Appendix B

JSON Web Token

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. For example, a server could generate a token that has the claim "logged in as administrator" and provide that to a client. The client could then use that token to prove that it is logged in as admin. The tokens can be signed by one party's private key (usually the server's) so that any party can subsequently verify whether the token is legitimate. If the other party, by some suitable and trustworthy means, is in possession of the corresponding public key, they too are able to verify the token's legitimacy. The tokens are designed to be compact [50], URL-safe [51] and usable, especially in a web-browser single-sign-on (SSO) context. JWT claims can typically be used to pass identity of authenticated users between an identity provider and a service provider, or any other type of claims as required by business processes [52]. JWT relies on other JSON-based standards: JSON Web Signature and JSON Web Encryption.