# Zero Trust Score-based Network-level Access Control in Enterprise Networks

Leonard Bradatsch<sup>†</sup>, Oleksandr Miroshkin<sup>‡</sup>, Nataša Trkulja<sup>†</sup>, Frank Kargl<sup>†</sup>, 

† Ulm University, Institute of Distributed Systems, Ulm, Germany

‡ Ulm University, KIZ, Ulm, Germany

Email: {leonard.bradatsch,oleksandr.miroshkin,natasa.trkulja,frank.kargl}@uni-ulm.de

Abstract—Zero Trust security has recently gained attention in enterprise network security. One of its key ideas is making network-level access decisions based on trust scores. However, score-based access control in the enterprise domain still lacks essential elements in our understanding, and in this paper, we contribute with respect to three crucial aspects. First, we provide a comprehensive list of 29 trust attributes that can be used to calculate a trust score. By introducing a novel mathematical approach, we demonstrate how to quantify these attributes. Second, we describe a dynamic risk-based method to calculate the trust threshold the trust score must meet for permitted access. Third, we introduce a novel trust algorithm based on Subjective Logic that incorporates the first two contributions and offers fine-grained decision possibilities. We discuss how this algorithm shows a higher expressiveness compared to a lightweight additive trust algorithm. Performance-wise, a prototype of the Subjective Logic-based approach showed similar calculation times for making an access decision as the additive approach. In addition, the dynamic threshold calculation showed only 7% increased decision-making times compared to a static threshold.

Index Terms—Network security, access control, trust

## I. INTRODUCTION

Recently, Zero Trust (ZT) security has gained popularity in enterprise network security, replacing traditional perimeter security, which has shortcomings like insider threats [15]. ZT enforces strict security requirements focused on resource protection and unauthorized access prevention [5], [21]. This paper focuses on ZT's trust-based access control at the network level. It governs whether an entity, such as example employee Alice, can access network resources such as her enterprise's Gitlab code repository [10]. Each resource access request (RAR) undergoes a trust-based access decision, performed by a trust algorithm [21].

NIST divides trust algorithms into criteria-based and score-based algorithms [21]. Criteria-based solutions such as XACML are proven and well-defined. Therefore, they are not considered further in this paper.

In score-based trust algorithms, a trust score is calculated for each resource access request (RAR). This score reflects

© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This work was supported by the bwNET2020+ project which is funded by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). The authors alone are responsible for the content of this paper.

the trustworthiness of the entity, like Alice, based on trust attributes such as expected access time or correct password. These trust attributes are weighted to indicate their impact on the trust score. For instance, if Alice accesses at her usual time, the score rises by the trust attribute's weight. The more the trust attribute contributes to the trust in the entity, the higher its weight. Conversely, the trust score may drop for unusual access times. The RAR is approved if it exceeds a predefined threshold; otherwise, it is not. Score-based algorithms, which allow attribute weighting, offer more dynamic decisions than criteria-based ones [21], making them a current research focus.

However, there is still a lack of consensus on important points in the trust score calculation in the enterprise domain [12]. From our perspective, this circumstance includes the following three fundamental aspects: First, the literature lacks a comprehensive list of trust attributes from the enterprise domain that can be considered for trust score calculation. So far, only subsets of possible trust attributes are presented such as in [12], [21] and [10]. Second, a mathematical model that allows the quantification of trust attribute weights is missing. Most of existing literature such as [7], [17], [25] use fixed weights without further definition. Third, the literature does not sufficiently address how to determine the threshold the trust score is compared with. In works such as [7], [11], the threshold is not defined but only a static value.

This lack motivates the main contributions of this paper:

- A comprehensive list of trust attributes, based on a systematic literature review, for evaluating entity trust in an enterprise context. Additionally, we introduce a mathematical model for quantifying the corresponding attributes' weights.
- An approach for dynamic calculation of the threshold which we refer to as the risk level.
- A novel trust-score calculation method based on Subjective Logic [16], incorporating the above contributions and discussed regarding its pros and cons. Additionally, we offer a publicly accessible proof of concept evaluated for performance.

We also highlight the need for trust in three entities: *user*, *device*, and *communication channel*. Additionally, we introduce a basic additive trust algorithm for comparison and performance evaluation against more complex methods like the Subjective-Logic-based approach.

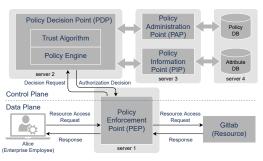


Fig. 1. Decision-making process in a ZT network, adapted from NIST [21]

The paper is organized as follows: Section II provides the technical background, while Section III motivates the need for trust in the mentioned three entities. Section IV provides a comprehensive list of trust attributes and their mathematical quantification. Section V introduces a new risk level calculation method. A straightforward additive approach for illustrating these concepts is presented in Section VI. Section VII introduces a novel fine-grained trust algorithm using Subjective Logic. Section VIII compares this with the additive approach, followed by a performance evaluation in Section IX. Lastly, we conclude by outlining future work and summarizing in Section X.

#### II. BACKGROUND

This section describes how ZT enterprise architectures enforce score-based access decisions at the network level, following NIST [21] guidelines and depicted in Figure 1.

When an entity like Alice requests access to a resource, like her enterprise's Gitlab server, the Policy Enforcement Point (PEP) initiates a trust-based decision request for the respective RAR. The PEP forwards this decision request to the Policy Decision Point (PDP), where the Policy Engine (PE) processes it. For this purpose, the PE fetches the relevant score-based policy from the policy database via the Policy Administration Point (PAP). Policies are expressed in score-based enabled policy languages like UCON+ [9] and are selected based on the requesting entity (here: Alice), action (here: resource access), requested resource (here: Gitlab) and context (e.g., RAR's date and time). Each policy specifies a trust algorithm and a threshold that represents the minimum amount of trust required for access. The specified trust algorithm calculates the trust score for the RAR, i.e., the entity involved in the RAR. This calculation uses trust attributes provided by a Policy Information Point (PIP). For example, Alice's usual access time could be a trust attribute. If the RAR is performed at Alice's usual access time, the trust attribute is considered met, which leads to a trust score adjustment based on the weight of the met trust attribute. For an additive trust algorithm, for example, the weight is then added to the trust score. After evaluating all trust attributes, the trust algorithm compares the calculated trust score to the threshold. If met, access is granted; otherwise, it's denied. The PDP then communicates this access decision to the PEP, which enforces it.

## III. TRUST ENTITIES

As previously outlined, the trust score is calculated for the entity, like Alice, initiating a RAR. However, such requests often involve more than just human entities; they can also include non-human entities, such as Alice's laptop [10]. Defining the set of entities involved in a RAR is important, yet there's no consensus in existing literature on which entities should be considered. In [21] and [17] user, application, and device are used as entities. Related works such as [8]-[10], [13], [27] and [18] consider the user and device as entities. In the works [19], [30], [29] and [24] the user as an entity is considered. For the same purpose, a formal requestor is used as the entity in [28]. In [7], [20], and [2], the requesting hardware is used as the entity. Zaheer et al. [31] uses the microservices hosted on a server as the trust entity. In [11] and [16], the authors focus on attributes only. Tian et al. [25] uses the entities user, terminal, file, and channel.

We have focused on selecting a well-motivated set of entities from those already used in the literature. Based on the ZT threat model, we motivate the choice of three specific entities for the trust score calculation.

Gilman et al. define a ZT threat model in which all communication channels and end systems, comprising both the user and the device, are presumed compromised [12]. For instance, the communication channel could have been eavesdropped, the user might be a rogue employee or the device could be malware-infected. Therefore, initial trust can not be assumed in the entities *user*, *device*, or *communication channel*. Trust must be established in each individual entity, as a compromised entity can already cause damage on its own.

Individual trust scores can be calculated for each entity and summed first before comparison to a threshold, or each individual trust score can be compared to separate thresholds. Sections VI and VII consider both a single total trust score and three separate scores, respectively. Section VIII discusses the pros and cons of each method.

#### IV. TRUST ATTRIBUTES & TRUST SCORES

As discussed before, score-based trust algorithms need to calculate trust scores for the entities *user*, *device*, and *communication channel*. This calculation is based on trust attributes. A trust attribute is a property of an entity such as Alice's usual access times or her correct password that can be leveraged to build trust in that entity. In different literature such as in [10], [12], [18], [21], [22], subsets of trust attributes are used. To the best of our knowledge, there is no comprehensive list of trust attributes in the context of enterprise networks. Through systematic literature review, we collected all possible trust attributes and filtered them according to their applicability in the enterprise domain. In the following, we provide the resulting list of trust attributes that are categorized based on the entity to which they apply:

- a) User Trust Attributes:
- Authentication Factors: Splits into one attribute per factor presented by the user, e.g., Password, Face ID, etc. [12].

- Authentication Patterns: Indicates the user's typical authentication behavior, such as commonly used authentication factors [12].
- Enterprise Presence: Reflects the time since the user last interacted with a resource and whether interaction is expected. For instance, interaction during the user's vacation would be suspicious [18].
- Trust History: Records the trust scores from the user's previous RARs [12].
- Input Behavior: Reflects user's usual input behavior. For example, it can be looked at how fast the user types [3].
- Service Usage: Considers whether the requested service is common for the user [12], [21].
- Device Usage: Indicates if the device used for the RAR is among the user's known devices [21].
- Access Time: Reflects if the RAR is made at a time that is common for that user [21].
- Access Rate: Looks at whether the current request rate is within the usual range for this user [12], [21].
- Database Date Update: It is checked how up-to-date the data about the user is in the database [12].

## b) Device Trust Attributes:

- Connection Security: Reflects the willingness of the device to communicate securely. If not, it could indicate an attack, e.g., downgrade attack [22].
- Software Patch Level: Represents the security level of the software used for making the RAR [12].
- System Patch Level: Reflects the security level of the system software, e.g., OS running on the device [10].
- Type: Considers the device's type or hardware configuration. For instance, personal computers are more prone to infection than smartphones [12], [18].
- Fingerprint: Depicts the device's unique software and hardware combination and checks if it aligns with the usual configuration [12].
- Setup Date: Addresses the time elapsed since the device was last reset, with longer periods increasing the risk of infection [12].
- Location: Checks if the device's current location matches one of its usual locations [12], [21].
- Health: Indicates the current hardware condition, such as CPU load [10], [12].
- Managed Device: Determines if the device is managed, serving as a basis for the relevance of other device attributes [12], [18], [21].
- Vulnerability Scan: Results of the last antivirus scan [10].
- Authentication Factors, Enterprise Presence, Trust History, Service Usage, User Usage, Database Data Update: Analogous to the corresponding user attributes but in relation to the device.

## c) Communication Channel (CC) Trust Attributes:

- Authentication: Method of authentication for the CC.
- Confidentiality: Type of CC's confidentiality protection.
- Integrity: The way of integrity protection for the CC.

If not all attributes apply to a specific enterprise, a subset can be used with an accordingly adjusted threshold.

To be able to calculate a trust score, a trust attribute has a weight associated to it where this weight represents the proportionate impact on the increase of trust on this entity if this attribute is met. Additionally, there can be different degrees of fulfillment. For each degree of fulfillment, a separate weight can be defined. For example, the weight can decrease the more the access time for Alice's RAR deviates from her usual access times, i.e. the more conspicuous it is.

In grey literature on ZT access control [12], [18], [21], a quantifiable model for weighting trust attributes is is left open. In [11], the authors link trust attributes to the binary access decisions allow or deny. Yao et al. [29] map trust attributes to three possible values: trusted, partially trusted, and untrusted. In other academic literature [2], [7], [9], [17], [24], [25] fixed weights are used without further definitions. To the best of our knowledge, there is currently no general model for quantifying trust attribute weights.

This lack motivates the following mathematical model that allows the quantification of trust attribute weights. We start by defining four components for a trust attribute:  $W_{tv}$ ,  $W_{wv}$ ,  $w_{av}$ and a weighting function  $f_W$ . The set of target values  $W_{tv}$ represent the trust attribute's values that lead to an increase in trust. Dependent on the trust attribute, the possible target values can be, for example, the continuous set of real numbers or a discrete set of strings. For the trust attribute evaluation, the trust in the entity is increased if  $w_{av} \in W_{tv}$ .  $w_{av}$  is the actual value of this trust attribute at the time when the trust algorithm evaluates it. To determine the trust increment for the respective trust attribute's target values, a set of weight values  $W_{wv}$  is defined for each trust attribute. Each target value in  $W_{tv}$  is mapped onto a weight value in  $W_{wv}$  so that  $f_W: W_{tv} \to W_{wv}$ . As described at the beginning of this section, the weighting function  $f_W$  can be used to map different degrees of fulfillment, represented by the different elements of  $W_{tv}$ , to different weight values in  $W_{wv}$ . Values for all four components vary by enterprise and should be set by security experts. Examples are given in Sections VI and VII.

# V. RISK ATTRIBUTES & RISK LEVEL

After the trust scores are calculated based on the trust attribute weights, they are compared to a predefined threshold. The threshold represents the minimum level of trust necessary to access the requested resource. In [25] and [24], the necessary trust depends on the sensitivity of the requested resource. Vanickis et al. [27] and Ramenzanpour et al. [20] base the threshold level on the risk inherent in the access. In [1], [7], [9], [11]–[13], [17], [21], [28], [29] and [2] the threshold is not defined. Yao et al. [30] make their trust threshold dependent on the environment in which the resource is located. In [8], Da Silva et al. describe an context-dependent additive approach to calculate a security level akin to a threshold. However, their work focuses on smart homes and has limited applicability to enterprise networks.

To our knowledge, there is no approach to dynamically calculate the threshold in ZT enterprise networks in a clearly defined manner. In the context of ZT, a relatively static value is used as a threshold without a clear explanation for how it was determined. We introduce an approach for dynamic threshold calculation based on a risk level. A dynamic calculation allows a timely threshold adjustment according to changing risk conditions such as an increased network threat level.

We argue that the threshold depends on the risk inherent in accessing the requested resource. Risk is composed of the possible damage to the enterprise (impact) and the probability of damage (chance of occurrence) [23]. The higher the level of risk, the higher the level of trust necessary, i.e., the higher the threshold. Therefore, the threshold is equal to the risk level, and for the rest of the paper, we use the risk level as the threshold. We further argue that the same risk level can be used here for all entity trust scores. This is based on the fact that any risks posed by a resource will impact the risk level for all entities. This is because an outdated system patch level can be exploited, for example, by a maliciously modified packet, a malicious user, but also by an infected device.

Analogously to the trust scores, the risk level can be calculated by an algorithm based on attributes with assigned weights. Risk attributes are used for this purpose and their weights are evaluated analogously to the trust attribute weights. For risk attributes,  $W_{wv}$  weights the risk of damage to an enterprise if the RAR were granted. The higher the risk inherent in a target value  $W_{tv}$ , the higher the mapping weight value in  $W_{wv}$ . Below we provide a comprehensive list of risk attributes based on systematic literature review:

- Request Protocol: Protocol used for the requested access, e.g. HTTP or FTP [9].
- Request Action: Requested interaction with the resource, e.g. HTTP GET or POST [10].
- Data Sensitivity: Sensitivity of the requested data and the resulting potential damage in the event of a leak [6].
- Service Software Patch Level: Patch level of the software serving the request [10], [12].
- System State: Operational state of the system, e.g. normal or in maintenance [12].
- System Threat Level: Security state of the system, e.g. normal or under attack [10], [12], [21].
- System Patch Level: Patch level of the system software, especially OS patch level [10], [12].
- Network State: Operational state of the network, e.g. normal or in maintenance [4].
- Network Threat Level: Security state of the network, e.g. normal or under attack [10], [12], [21].

Two concrete algorithms for calculating the risk level including risk attributes with associated weights are introduced in Sections VI and VII.

#### VI. ADDITIVE-BASED TRUST ALGORITHM

In this section, we introduce a straightforward additive trust algorithm. It illustrates the principles from Sections III, IV, and V and is later used to compare against the Subjective-Logic-based approach introduced in Section VII. For the sake of simplicity, a single total trust score combining all entities' trust attributes is calculated additively for each RAR. This score is then compared to the RAR's risk level, and access is granted only if the trust score surpasses the risk level.

#### A. Trust Score

For each RAR, the total trust score  $TS_{total} \in \mathbb{R}$  is initialized with 0. After that, all considered trust attributes are evaluated.  $W_{tv}$  and  $W_{wv} = \{wv|wv \in \mathbb{R}\}$  must be defined for each attribute where  $f_W: W_{tv} \to W_{wv}$ . If  $w_{av} \in W_{tv}$ , the respective weight value wv where  $f_W(w_{av}) \to wv$  is added to  $TS_{total}$ . This results in the following equation:

$$TS_{total} = \begin{cases} TS_{total} + f_W(w_{av}) & \text{if } w_{av} \in W_{tv} \\ TS_{total} & \text{else} \end{cases}$$
 (1)

For example, Alice wants to access her enterprise's Gitlab. She uses *Password Authentication* and her correct password is "1234". For her, this results in  $W_{tv} = \{\text{``1234''}\}$ . We have chosen the example weight of 5 to add in case the entered password is correct. Consequently, the set of weight values is  $W_{wv} = \{5\}$  where  $f_W(\text{``1234''}) \to 5$ . Alice enters her password correctly  $(w_{av} = \text{``1234''})$ , which results in  $TS_{total} = 0 + f_W(\text{``1234''}) = 5$ .

#### B. Risk Level

After  $TS_{total}$  has been calculated, it is compared to a total risk level  $RL_{total} \in \mathbb{R}$ , which is initialized with 0. To determine the final  $RL_{total}$ , all risk attributes are evaluated analogously to the trust attributes.  $RL_{total}$  is then formed additively in the same way as  $TS_{total}$ :

$$RL_{total} = \begin{cases} RL_{total} + f_W(w_{av}) & \text{if } w_{av} \in W_{tv} \\ RL_{total} & \text{else} \end{cases}$$
 (2)

For example, we consider again the enterprise's Gitlab service. For the risk attribute *System Patch Level*, the enterprise maintains two values: up-to-date and outdated. The enterprise considers access to up-to-date systems as not risky and thus up-to-date is not an element of the set of target values  $W_{tv}$  for this risk attribute. Consequently,  $W_{tv} = \{"outdated"\}$  and  $W_{wv} = \{10\}$  where  $f_W("outdated") \rightarrow 10$ . Assuming the Gitlab server runs an outdated version  $(w_{av} = "outdated")$ , this leads to  $RL_{total} = 0 + f_W("outdated") = 10$ .

## C. Access Decision

Access to the service is granted only if the following inequality is satisfied:

$$TS_{total} > RL_{total}$$
 (3)

Based on the previous example for  $RL_{total}=10$  and  $TS_{total}=5$ , this leads to the inequality 5>10 and thus the RAR is rejected.

#### VII. SUBJECTIVE LOGIC-BASED TRUST ALGORITHM

We now introduce a fine-grained trust algorithm using Subjective Logic (SL) [14], contrasting the additive approach. The SL-based method considers individual entity trust scores and varying degrees of attribute fulfillment, along with dynamic risk levels.

SL is used for decision-making under uncertainty. Uncertainty in a decision arises when a decision-maker lacks complete information at the time of making a decision. In the ZT context, SL informs access decisions based on trust, as for the additive approach. However, this approach offers finer granularity by forming separate opinions on the trustworthiness of the *user*, *device*, and *communication channel*. Access is granted only if the trust scores for each entity surpass the associated risk level. We will outline how these individual trust scores and risk levels are calculated.

## A. User Trust Score

To obtain the user trust score, we first form a binary opinion  $\omega_X^A$  about whether the user is considered trustworthy. In SL,  $\omega_X^A$  represents the opinion about the proposition X made by the subjective agent A. The proposition X is a random variable  $X \in \mathbb{X} = \{x, \overline{x}\}$ . In the case of the user trust, the binary random variable  $UT \in \mathbb{X} = \{ut, \overline{ut}\}$  represents the proposition about the user trustworthiness UT:

- ut: the user who performs this RAR is trustworthy
- $\overline{ut}$ : the user who performs this RAR is not trustworthy

Here, A represents the PDP, which forms an opinion about the user trustworthiness. This results in the opinion  $\omega_{ut}^{\text{PDP}} = \{b_{ut}, d_{ut}, u_{ut}, a_{ut}\}$ . The parameters are defined as follows:

- ullet  $b_{ut}$ : belief that the user is trustworthy based on evidence
- d<sub>ut</sub>: disbelief that the user is trustworthy based on evidence
- $u_{ut}$ : uncertainty whether the user is trustworthy due to lack of evidence
- $a_{ut}$ : base rate represents prior knowledge about user trustworthiness. If no prior knowledge is available, SL defines a default value for binary opinions of 0.5.

where  $b_{ut}, d_{ut}, u_{ut}, a_{ut} \in [0, 1]$  and the following additive requirement [14] is satisfied:

$$b + d + u = 1 \tag{4}$$

The overall opinion  $\omega_{ut}^{\text{PDP}}$  is a composition of individual opinions from subjective agents evaluating trust attributes. In our approach, we have on agent for each trust attribute. Each agent forms its own opinion  $\omega_{ut}^{\text{Attribute}}$  based on the evidence of the attribute it evaluates. These opinions are then merged to create  $\omega_{ut}^{\text{PDP}}$  using the weighted belief fusion operator  $\hat{\diamond}$  for SL, as defined in [26]. We argue that this fusion is well suited for trust-building as it allows for trust attribute weighting. For instance, retina authentication could be weighted higher than password authentication due to its arguably better security. This fusion express the weighting of opinions based on their uncertainty. Consequently, the set of weight values is defined as  $W_{wv} = \{u_{ut}^A|u_{ut}^A \in [0,1]\}$ . The lower the uncertainty

of an opinion, the higher the weighting during the fusion. The fused opinion is defined as  $\omega_{ut}^{\text{PDP}} = (b_{ut}^{\lozenge \lozenge \lozenge \lozenge \lozenge}, d_{ut}^{\lozenge \lozenge \lozenge \lozenge}, u_{ut}^{\lozenge \lozenge \lozenge \lozenge}, a_{ut}^{\lozenge \lozenge \lozenge \lozenge})$  where  $\mathbb A$  is the set of subjective agents; in this case all agents that evaluate user trust attributes. From this overall opinion  $\omega_{ut}^{\text{PDP}}$ , we calculate the user trust score using projected probability [14]  $P(UT=ut) = b_{ut}^{\lozenge \lozenge \lozenge \lozenge \lozenge} + u_{ut}^{\lozenge \lozenge \lozenge \lozenge} \cdot a_{ut}^{\lozenge \lozenge \lozenge}$ . This user trust score represents the probability that the user is trustworthy.

We now provide a simplified example using the subjective agent PWAuth, which forms an opinion  $\omega_{ut}^{PWAuth} = \{b_{ut}, d_{ut}, u_{ut}, a_{ut}\}$  on user trustworthiness. This opinion is based on two evidence factors: The correctness of the entered password and the already failed attempts to enter the correct password. So here  $w_{av}$  is a pair (pw, att) where pw is the entered password and  $att \in \mathbb{N}_0$  is the amount of already failed attempts. Consequently,  $W_{tv}$  is a set of pairs (pwc, att) where pwc is the correct password. Assuming Alice enters her password correctly after 5 failed attempts  $(w_{av} = (pwc, 5))$ , this results in the following example parameter values:

- $b_{ut} = 0.2$  (evidence that user is trustworthy as  $w_{av} \in W_{tv}$ )
- $d_{ut} = 0.6$  (evidence for untrustworthiness as  $att \neq 0$ , which could be a sign of a password guessing attack)
- $u_{ut} = 0.2$  (A residue of uncertainty since Alice could have mistyped 5 times)
- $a_{ut} = 0.5$  (no prior knowledge is available)

More failed attempts reduce uncertainty  $u_{ut}$  as they serve as evidence, making the subjective agent PWAuth more confident in its opinion. This decrease in uncertainty leads to increased disbelief  $d_{ut}$ , as accumulating failed attempts suggest an illegitimate login attempt. For the attribute weights, we get the mapping  $f_W:(pwc,att)\to u_{ut}$ . For Alice it is  $f_W:(pwc,5)\to 0.3$ . Since we are evaluating only one attribute in this example, we obtain the overall opinion  $\omega_{ut}^{\text{PDP}}=\omega_{ut}^{\text{PWAuth}}=\{0.2,0.6,0.2,0.5\}$  from it. We can now calculate the probability  $P(UT=ut)=0.2+0.2\cdot 0.5=0.3$  that Alice is trustworthy.

# B. Device Trust Score

For device trustworthiness DT, the random variable  $DT \in \mathbb{X} = \{dt, \overline{dt}\}$  has the two outcomes:

- dt: the device that performs this RAR is trustworthy
- $\overline{dt}$ : the device that performs this RAR is not trustworthy

This results in the overall opinion  $\omega_{dt}^{\text{PDP}}=(b_{dt}^{\Diamond \mathbb{A}},d_{dt}^{\Diamond \mathbb{A}},u_{dt}^{\Diamond \mathbb{A}},a_{dt}^{\Diamond \mathbb{A}}),$  which is build analogously as for the user trustworthiness. From this opinion, the device trust score  $P(DT=dt)=b_{dt}^{\Diamond \mathbb{A}}+u_{dt}^{\Diamond \mathbb{A}}\cdot a_{dt}^{\Diamond \mathbb{A}}$  is calculated.

# C. Communication Channel Trust Score

Here, the PDP forms the overall opinion  $\omega_{cct}^{\text{PDP}} = (b_{cct}^{\widehat{\Diamond}\mathbb{A}}, d_{cct}^{\widehat{\Diamond}\mathbb{A}}, u_{cct}^{\widehat{\Diamond}\mathbb{A}}, a_{cct}^{\widehat{\Diamond}\mathbb{A}})$  about the communication channel trustworthiness  $CCT \in \mathbb{X} = \{cct, \overline{cct}\}$  with the two outcomes:

- cct: the communication channel is trustworthy
- $\overline{cct}$ : the communication channel is not trustworthy

The overall opinion  $\omega_{cct}^{\text{PDP}}$  is formed analogously as for the user trustworthiness. From this, the communication channel trust score  $P(CCT=cct)=b_{cct}^{\Diamond \mathbb{A}}+u_{cct}^{\Diamond \mathbb{A}}\cdot a_{cct}^{\Diamond \mathbb{A}}$  is calculated.

## D. Risk Level

For determining the risk level, an opinion  $\omega_{rod}^{PDP}$  is build on the risk of damage (ROD) to the enterprise if the RAR is permitted. This proposition is represented by the random variable ROD  $\in \mathbb{X} = \{rod, rod\}$  with

- rod: Permitting the RAR will cause damage
- $\overline{rod}$ : Permitting the RAR will not cause damage

The parameters for the risk opinion  $\omega_{rod}^{PDP}$   $\{b_{rod}, d_{rod}, u_{rod}, a_{rod}\}$  are defined as

- $b_{rod}$ : belief that permitting the RAR causes damage to the enterprise based on evidence
- $d_{rod}$ : disbelief that permitting the RAR causes damage to the enterprise based on evidence
- u<sub>rod</sub>: uncertainty whether permitting the RAR causes damage to the enterprise due to lack of evidence
- $a_{rod}$ : base rate based on prior knowledge about permitted RARs that caused damage to the enterprise

where  $b_{rod}, d_{rod}, u_{rod}, a_{rod} \in [0, 1]$  and the additive requirement (4) is satisfied.

The risk opinion  $\omega_{rod}^{\text{PDP}}$  is a composition of the opinions formed by the respective subjective agents. Each subjective agent collects evidence based on the respective risk attribute it evaluates. All individual opinions regarding risk are then fused to obtain the overall opinion  $\omega_{rod}^{\text{PDP}}$ . As fusion we use the associative cumulative fusion operator  $\diamond$  defined in [14]. The cumulative fusion is defined as  $\omega_X^{\rm A}=(b_X^{A\diamond B},d_X^{A\diamond B},u_X^{A\diamond B},a_X^{A\diamond B})$ where A and B are two agents that formed an opinion about the same proposition X. For the total risk opinion  $\omega_{rod}^{PDP}$  we fuse all single agents' opinions sequentially. This results in the overall opinion  $\omega_{rod}^{\text{PDP}} = (b_{rod}^{\diamond \mathbb{A}}, d_{rod}^{\diamond \mathbb{A}}, u_{rod}^{\diamond \mathbb{A}}, a_{rod}^{\diamond \mathbb{A}})$  where  $\diamond \mathbb{A}$ represents the set A of all risk agents that are all fused by the operator  $\diamond$ . We argue that the fusion operator used for trust scores is unsuitable here, as weighted fusion can lead to lower belief values (here: the belief in causing damage). Unlike trust scores, where reduced belief in trustworthiness is acceptable if single trust attributes are not met, the belief in the risk of damage should not decrease. For instance, an outdated system patch can not be compensated by an updated service patch. Existing risks just add up. A cumulative fusion method, summing the evidence parameters b and d from individual risk opinions, is more appropriate here. Consequently, for the SL risk level calculation risk attribute weights are directly derived from the respective opinion's evidence parameters  $b_{rod}^A$  and  $d_{rod}^A$ :  $W_{wv} = \{(b_{rod}^A, d_{rod}^A) | b_{rod}^A, d_{rod}^A \in [0, 1]\}$ . To get the final risk level, the projected probability P(ROD = rod) = $b_{rod}^{\diamond \mathbb{A}} + u_{rod}^{\diamond \mathbb{A}} \cdot a_{rod}^{\diamond \mathbb{A}}$  is formed.

As an example, this could result in a risk opinion  $\omega^{\mathrm{SPL}}_{rod} = \{b_{rod}, d_{rod}, u_{rod}, a_{rod}\}$  formed by the agent that evaluates the attribute  $System\ Patch\ Level$  of the Gitlab server. The enterprise manages two patch levels: "outdated" and "upto-date". Thus, the set of attribute target values is  $W_{tv} = \{$ "outdated", "up-to-date" $\}$ . The example parameter values for a system with the latest patch ( $w_{av} =$  "up-to-date") could look like this:

- $b_{rod} = 0.0$  (no evidence that permitting this RAR will cause damage to the enterprise)
- $d_{rod}$  = 0.8 (evidence that permitting this RAR will cause no damage as  $w_{av}$  = "up-to-date")
- $u_{rod} = 0.2$  (uncertainty since there could be undetected vulnerabilities for the latest patch)
- $a_{rod} = 0.5$  (no prior knowledge is available)

For the function  $f_W:W_{tv}\to (b_{rod}^A,d_{rod}^A)$ , we then get the example mapping  $f_W:"up\text{-}to\text{-}date"\to (0.0,0.8)$ . Since we consider only one attribute in the example, we get the overall risk opinion  $\omega_{rod}^{\text{PDP}}=\omega_{rod}^{\text{SPL}}=\{0.0,0.8,0.2,0.5\}$ . This results in  $P(\text{ROD}=rod)=0.0+0.2\cdot0.5=0.1$ .

## E. Access Decision

To make an access decision, the individual entity trust scores are compared to the risk level. As we will discuss in Section VIII, comparing individual scores to the risk level prevents trust compensation. Access is only permitted if the trust in each entity exceeds the risk of damage. This results in the following three inequalities that must all be satisfied:

$$P(ut) > P(rod), P(dt) > P(rod), P(cct) > P(rod)$$
 (5)

The example user trust score (P(ut) = 0.3) and risk level (P(rod) = 0.1) lead to the inequality 0.3 > 0.1. Considered for the user trust score, the RAR would be permitted.

## VIII. DISCUSSION & RELATED WORK

Following the introduction of both trust algorithms, we explore their qualitative aspects, such as expressiveness.

Score-based trust algorithms use attribute weights for trust score calculations. In the additive method, an attribute is either fully weighted or not, while the SL-based approach allows for nuanced weighting through continuous belief and disbelief values. For example, as discussed in Section VII, more failed password attempts by Alice increases disbelief in her identity. As we gather evidence through failed attempts, the uncertainty diminishes, leading to a higher weighting of the attribute. While prior studies have explored trust score calculations, they often have a different focus or less generality. Dimitrakos et al. [9] implemented an SL-based approach with focus on device authentication by sensors in the IoT domain. The authors in [8] use different discrete values for specifying their trust weights in the context of smart homes. Albuali et al. [2] provide a fine-granular model while focusing on three behavioral aspects in cloud computing. In [24] and [25] fixed weights are considered for their calculations. Yao [30] calculate attribute target values for three behavioral aspects using historical data but have not vet elaborated on their method for setting and adjusting weights. While our SL-based algorithm allows for fine-grained weight adjustments, it increases the demand to determine reasonable weights by a security expert for each use case. Additionally, the mathematical complexity of the SL-based approach increases the risk of errors.

Similarly, we have explored different trust score approaches in our algorithms. The additive approach calculates a single, total trust score, a common practice in the majority of the literature such as in [11], [16], [30] or [8]. In [25], single trust scores for the user, terminal, and channel are calculated but before being compared to the threshold, all are subtracted from each other to build one total score. One total trust score simplifies risk-level reconciliation but lacks granularity. In contrast, our SL-based approach calculates separate scores for each entity, allowing for finer access control and preventing unintended trust compensation. For instance, a high device score should not offset an untrustworthy user. However, managing individual scores for each entity complicates matching them with appropriate risk levels.

In addition to entity trust scores, we presented a calculation method for the threshold, we refer to as risk level. As discussed in Section V, this threshold is a relatively static value in the current literature, with mostly no guidance on how to set it. By calculating the risk level based on risk attributes, it is possible to dynamically recalculate this threshold for each RAR. Due to the weighted influence of the individual risk attributes, the adjustment can be realized with the same granularity as for the trust value. This also means that the threshold can be automatically adapted to the enterprise's current risk situation for each RAR.

#### IX. PERFORMANCE EVALUATION

As poor user experience can compromise security, performance is a crucial factor [12]. In this section, we evaluate the decision-making time of both trust algorithms in a realistic ZT enterprise setting following ZT requirements of mutual authentication, encryption, and integrity protection for all communication channels [5]. We aim to assess (1) how the fine-grained SL-based algorithm impacts decision-making time compared to the straightforward additive algorithm, and (2) the effect of dynamic versus static risk level calculation, the latter commonly seen in the literature.

We set up a testbed featuring all components shown in Figure 1. Each of the four servers, equipped with an Intel Xeon E5-2630 v3 CPU (32 Cores, 64 Threads), 128 GB DDR4 RAM, and a Samsung SSD 850 EVO 500GB, are interconnected using Mellanox Connect-X4 100G network cards and direct server cabling. Implemented in Golang 1.20.2 on Ubuntu 22.04.2, the setup aligns with Section II and is publicly available on GitHub<sup>1</sup>. All communications are secured with mutual TLS. The databases and ZT components communicate via REST APIs and SQL, respectively. We filled all 29 trust and 9 risk attributes with dummy values and weights, which do not affect performance compared to realistic data.

We measured the decision-making time from when the PEP sent the decision request to when the authorization decision was received, as shown in Figure 1. That allowed us to assess the impact of trust algorithm differences, also considering other factors like network operations. Both trust algorithms' pure calculation time was in the low microsecond range (10- $40\mu$ s). We measured decision-making time in two scenarios, comparing additive and SL-based approaches with both static

and dynamic risk levels. The PEP ran 1 to 128 parallel instances, each sending 1000 requests. Tests were repeated 10 times and reported values represent the median.

In the first test scenario, we examined the PIP's worst-case scenario, where its attribute cache is empty and all trust attributes must be retrieved from the database. As shown in Figure 2, decision-making time increases even before reaching the optimal CPU load of 64 parallel PEP instances due to blocking network system calls in kernel space. At 64 parallel PEP instances, both the additive trust algorithm (296ms) and the SL-based trust algorithm (294ms) exhibit nearly identical decision-making times with a static risk level. However, when calculating a dynamic risk level, which requires loading additional risk attributes from Section V, decision-making time increases by 7% for both algorithms (317ms for additive, 316ms for SL-based). This slight difference is due to all attributes being loaded in one batch per request. Once the PIP

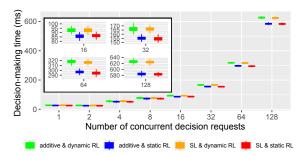


Fig. 2. Decision-making time in the case the PIP queries all attributes from the database. For the measured times, the quartiles are presented.

has loaded all attributes, it serves the PDP's attribute requests from its cache. The results of this second test scenario are shown in Figure 3. In the case of 64 parallel PEP instances, the decision-making time for both trust algorithms with static risk level (126ms) as well as with dynamic risk level calculation (133ms) is about 42% shorter compared to the first test scenario. The tests showed that both trust algorithms have the

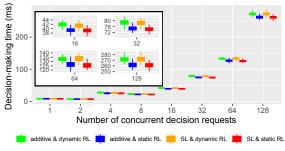


Fig. 3. Decision-making time in the case the PIP holds all attributes in the cache. For the measured times, the quartiles are presented.

same decision-making time. The additional dynamic risk level calculation increases the decision-making time by only 7% and is of little impact compared to the dominant time factors of database transfers and network-related operations. In summary, dynamic risk level calculation and the SL-based trust algorithm do not have a significant impact on performance.

 $<sup>^{1}</sup> https://github.com/zintpavowj/ZT\_Score-based\_Network\_Level\_AC$ 

#### X. CONCLUSION & FUTURE WORK

In this paper, we initially identify three key trust entities - *user*, *device*, and *communication channel* - essential in RARs. These are motivated by the ZT threat model, which presumes initial distrust in these entities. Hence, trust calculations for RARs must include all three.

Trust in the three entities is determined through trust attributes. Based on a systematic literature review, we provide a comprehensive list of 29 such trust attributes for enterprise settings, categorized by applicable entities. Due to a gap in the literature for attribute weighting, we introduce a mathematical model for this purpose. Our model maps continuous trust-building target values to weights and allows for varying degrees of attribute fulfillment.

For access decisions, calculated trust scores are compared to a threshold. We argued that the required trust level depends on the potential risk of damage from granting access, leading us to introduce a novel risk level calculation method as the threshold. Similar to trust scores, this risk level is determined by weighted risk attributes, enabling dynamic risk level calculations for each RAR.

To illustrate the discussed contributions, we first described a basic additive trust algorithm that calculates a single trust score, compared against a dynamically calculated risk level. We then introduced a new trust algorithm based on SL, which effectively utilizes all prior contributions. This algorithm compares separate trust scores for each entity, allowing for nuanced attribute fulfillment, against a risk level.

We observed that the SL-based algorithm allows for fine-grained trust calculations, improving decision expressiveness but adding complexity in attribute weighting. The performance evaluation showed that this complexity, along with dynamic risk levels, does not significantly impact decision-making time compared to the straightforward additive approach with a static risk level. Therefore, performance is not a barrier to using the more expressive SL-based approach. Future research will explore if this leads to more accurate access decisions.

Therefore, the next step is to evaluated how the weights for the attributes can be determined best for each trust algorithm. Currently the usual approach is that for each use case the responsible security expert has to define them. Based on this, it is necessary to investigate which approach leads to more accurate access decisions. Here, accurate is to be understood as the lowest possible false positive and false negative rates.

#### REFERENCES

- I. Ahmed, T. Nahar, S. S. Urmi, and K. A. Taher, "Protection of sensitive data in zero trust model," in *Proceedings of the International Conference* on Computing Advancements, 2020, pp. 1–5.
- [2] A. Albuali, T. Mengistu, and D. Che, "Ztimm: A zero-trust-based identity management model for volunteer cloud computing," in *International Conference on Cloud Computing*. Springer, 2020, pp. 287–294.
- [3] M. L. Ali, J. V. Monaco, C. C. Tappert, and M. Qiu, "Keystroke biometric systems for user authentication," *Journal of Signal Processing Systems*, vol. 86, no. 2-3, pp. 175–190, 2017.
- [4] A. L. Aliyu, P. Bull, and A. Abdallah, "A trust management framework for network applications within an sdn environment," in 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE, 2017, pp. 93–98.

- [5] L. Bradatsch, M. Haeberle, B. Steinert, F. Kargl, and M. Menth, "Secure service function chaining in the context of zero trust security," in 2022 IEEE 47th Conference on Local Computer Networks (LCN), 2022, pp. 123–131.
- [6] B. Chen, S. Qiao, J. Zhao, D. Liu, X. Shi, M. Lyu, H. Chen, H. Lu, and Y. Zhai, "A security awareness and protection system for 5g smart healthcare based on zero-trust architecture," *IEEE Internet of Things Journal*, 2020.
- [7] T. Chuan, Y. Lv, Z. Qi, L. Xie, and W. Guo, "An implementation method of zero-trust architecture," in *Journal of Physics: Conference Series*, vol. 1651, no. 1. IOP Publishing, 2020, p. 012010.
- [8] G. R. da Silva, D. F. Macedo, and A. L. dos Santos, "Zero trust access control with context-aware and behavior-based continuous authentication for smart homes," in *Anais do XXI Simpósio Brasileiro em Segurança* da Informação e de Sistemas Computacionais. SBC, 2021, pp. 43–56.
- [9] T. Dimitrakos, T. Dilshener, A. Kravtsov, A. L. Marra, F. Martinelli, A. Rizos, A. Rosetti, and A. 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), 2020, pp. 1801–1812.
- [10] J. Garbis and J. W. Chapman, Zero Trust Security. Springer, 2021.
- [11] N. Ghate, S. Mitani, T. Singh, and H. Ueda, "Advanced zero trust architecture for automating fine-grained access control with generalized attribute relation extraction," *IEICE Proceedings Series*, vol. 68, no. C1-5, 2021.
- [12] E. Gilman and D. Barth, Zero Trust Networks: Building Secure Systems in Untrusted Networks. O'Reilly Media, Inc., Jun. 2017.
- [13] K. Hatakeyama, D. Kotani, and Y. Okabe, "Zero trust federation: Sharing context under user control towards zero trust in identity federation," in 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). IEEE, 2021, pp. 514–519.
- [14] A. Jøsang, Subjective Logic: A Formalism for Reasoning Under Uncertainty. Springer Cham, 2016.
- [15] J. Kindervag, S. Balaouras, and L. Coit, "No More Chewy Centers: Introducing The Zero Trust Model Of Information Security," Forrester, Tech. Rep., Sep. 2010.
- [16] S. Mandal, D. A. Khan, and S. Jain, "Cloud-based zero trust access control policy: an approach to support work-from-home driven by covid-19 pandemic," *New Generation Computing*, vol. 39, no. 3, pp. 599–622, 2021.
- [17] S. Mehraj and M. T. Banday, "Establishing a zero trust strategy in cloud computing environment," in 2020 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2020, pp. 1–6.
- [18] B. Osborn, J. McWilliams, B. Beyer, and M. Saltonstall, "Beyondcorp: Design to deployment at google," ;login:, vol. 41, pp. 28–34, 2016.
- [19] D. Puthal, S. P. Mohanty, P. Nanda, and U. Choppali, "Building security perimeters to protect network systems against cyber threats [future directions]," *IEEE Consumer Electronics Magazine*, vol. 6, no. 4, pp. 24–27, 2017.
- [20] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5g/6g tactical networks: Principles, challenges, and the role of machine learning," arXiv preprint arXiv:2105.01478, 2021.
- [21] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Computer Security Resource center, Aug. 2020.
- [22] H. Sateesh and P. Zavarsky, "State-of-the-art vanet trust models: Challenges and recommendations," in 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEM-CON). IEEE, 2020, pp. 0757–0764.
- [23] Y. Shi, D. L. Olson, and A. Stam, Advances in multiple criteria decision making and human systems management: Knowledge and wisdom. IOS press, 2007.
- [24] Y. Tao, Z. Lei, and P. Ruxiang, "Fine-grained big data security method based on zero trust model," in 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2018, pp. 1040– 1045.
- [25] X. Tian and H. Song, "A zero trust method based on BLP and BIBA model," in 2021 14th International Symposium on Computational Intelligence and Design (ISCID), 2021, pp. 96–100.
- [26] R. W. van der Heijden, H. Kopp, and F. Kargl, "Multi-source fusion operations in subjective logic," *CoRR*, vol. abs/1805.01388, 2018.
- [27] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access Control Policy Enforcement for Zero-Trust-Networking," in 2018 29th Irish Signals and Systems Conference (ISSC). IEEE, Jun. 2018, pp. 1–6.

- [28] M. Xiaoning, "Formal description of trust-based access control," *Physics Procedia*, vol. 33, pp. 555–560, 2012.
- [29] K. Yang, D. Li, L. Zhou, and K. Cheng, "Research on adaptive dynamic access control model based on blockchain and token," in *Journal of Physics: Conference Series*, vol. 2166, no. 1. IOP Publishing, 2022, p. 012042
- [30] Q. Yao, Q. Wang, X. Zhang, and J. Fei, "Dynamic access control and authorization system based on zero-trust architecture," in 2020 International Conference on Control, Robotics and Intelligent System, 2020, pp. 123–127.
- [31] Z. Zaheer, H. Chang, S. Mukherjee, and J. Van der Merwe, "eztrust: Network-independent zero-trust perimeterization for microservices," in Proceedings of the 2019 ACM Symposium on SDN Research, 2019, pp. 49–61.