Technological University of the Shannon
Midlands Midwest

Ollscoil Teicneolaiocha don Sionainn
Lar na Tire Lar Thiar

A Policy Language for Context-Aware Access Control in Zero-trust Network

By

Shiyu Xiao

Thesis

Submitted to the School of Engineering

Technological University of the Shannon Midlands Midwest

Athlone, Co. Westmeath, Republic of Ireland

In the partial fulfilment of the requirements for the degree of

Master of Science (Research)

Technological University of Shannon

(June 2023)

Supervised by:

Dr. Brian Lee

Dr. Nadia Kanwal

A Policy Language for Context-aware Access Control in Zero-trust Network

Approved by

Supervising Committee:

_____

(Dr. Brian Lee)

_____

(Dr. Nadia Kanwal)

Acknowledgements

I would like to express my deepest gratitude to Dr. Brian Lee and Dr. Nadia Kanwal for their guidance and support throughout my research. Their expertise, encouragement, and feedback have been invaluable to the completion of this thesis.

I would like to thank colleagues in Software Research Institute of Technological University of Shannon for their support and camaraderie throughout this journey.

Special thanks to my parents for their unwavering support and understanding during the ups and downs of this project.

Finally, I would like to thank the HEA Government of Ireland Scholarship and President's Seed Funding for providing the necessary resources and facilities for this research.

Thank you all for your contributions and support in making this thesis possible.

Declaration of Authorship

I, Shiyu Xiao, declare that this thesis titles, ‘A Policy Language for Context-aware Access Control in Zero-trust Network’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this Institute.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other Institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from work of others, the source is always given. With the exception of such quotation, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

(Shiyu Xiao)

Date:_____

Abstract

Evolving computing technologies such as cloud, edge computing, and the Internet of Things (IoT) are creating a more complex, dispersed, and dynamic enterprise operational environment. New security enterprise architectures such as those based on the concept of Zero Trust (ZT) are emerging to meet the challenges posed by these changes. Context awareness is a notion from the field of ubiquitous computing that is used to capture and react to the situation of an entity, based on the dynamics of a particular application or system context. However, there is limited research and discussion about the overlap between context awareness and Zero Trust, with existing literature often treating them as separate entities, leading to potential inefficiencies. One of the main challenges in merging the two concepts is the inflexibility of the programming languages and systems used in crafting access control policies, which sometimes result in excessively rigid policies. Addressing this challenge could be achieved through a new programming language specifically designed for greater flexibility and a wider consideration of contextual factors, leading to more robust security measures that align more effectively with the principles of Zero Trust.

This work conducts a systematic review of the previous research in context-aware access control to identify the various ways to capture and express context across different access control types and different application domains. Based on this review, it identifies how context can help provide dynamic policy-based solutions for zero-trust applications.

It extends a previous work which designed a policy language for risk-based access control in zero-trust networks. Specifically, this project extends the necessary language constructs to include and handle dynamic contextual attributes.

Finally, it provides a proof of concept to demonstrate that the extended language can give the correct access decisions based on the evaluation of contextual information in zero-trust network.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Motivation

In recent years, there has been a trend to move away from traditional "perimeter security defence" systems. These are based on so-called Demilitarised Zones (DMZ) i.e., a sub-network(s) containing an enterprise's external facing services, typically bookended by internal and external facing firewalls, (Gilman and Barth, 2017). The fundamental premise behind this defence system is that anything outside the perimeter is untrusted whilst everything inside the perimeter is trusted. A major weakness of this however is that once an adversary gains access to the internal network it becomes easy for them to move laterally throughout the network and so compromise other hosts and servers.

There have been several drivers for this weakening of the perimeter security model. The nature of enterprise networking has become more complex through the evolution of computing technology, (Cunningham et al., 2019). Initial developments in this regard included cloud computing and Bring Your Own Device (BYOD) whilst more recent approaches such as the Internet of Things (IoT) and edge computing have increased the complexity level further (Kim and Lee, 2017). The increase of remote working brought about by Covid has added even more to the mix. The net effect of these trends has been to create a highly dispersed and fragmented enterprise architecture, often with many of the enterprise applications running on third-party hardware. Moreover, the operating environment may often be highly dynamic due to end-users being situated in different locations or changes in operating conditions for devices in environments where the resources are constrained, or their availability may vary from moment to moment.

A new enterprise security model has consequently emerged to meet the challenges posed by these changes. This is designated as Zero Trust Networking (ZTN) which essentially regards the internal enterprise network as untrusted. It thus treats internal and external networks with the same degree of suspicion and subject to the same security checking and control and in this way seeks to prevent data breaches and limit internal lateral movement (Rose et al., 2020). Key principles of ZT include the requirement to validate every access on a per-session basis as well as the use of dynamic policy enforcement taking into account device and user attributes as well as, perhaps, other behavioural and environmental attributes. The zero-trust concept was introduced by Forrester Research Group (Kindervag, 2010) as a new, radical approach to enterprise security. The ZT approach truly took wings when Google implemented a ZT-based enterprise security architecture called "BeyondCorp", (Osborn et al., 2016a), (Ward and Beyer, 2014). ZT has since therefore, unsurprisingly, been embraced with gusto by the commercial world and many vendors today have ZTN product offerings leading to somewhat different definitions of the concepts.

1.2 Motivation

As it is described before, the ZT model has emerged as a promising access control paradigm, which shifts the focus from traditional perimeter-based security to a **context-aware** approach.

Context awareness provides the system with the ability to gather information about the environment at any given time and adapt system behaviours accordingly. To achieve context awareness in ZT system, it needs to understand the user, device, network, and other relevant factors before granting access. This step is crucial because it allows for a more dynamic and adaptive security measures that can respond to the ever-changing threat landscape. **However, the criticality of *context awareness* and the related issues of *risk* and *trust* in Zero Trust are not fully articulated, leading to a gap in the current body of knowledge and practice, particularly concerning how to capture and express these motions in access control policies.**

The current literature and practice primarily focus on the individual components of Zero Trust and context awareness, often treating them as separate entities. This separation creates a disjointed understanding and application of the two concepts, leading to potential vulnerabilities and inefficiencies.

One of the key challenges in integrating context awareness into Zero Trust is the rigidity of existing programming languages and systems used to design access control policies. These languages and systems often lack the flexibility needed to incorporate dynamic context effectively. This limitation can lead to policies that are either too rigid, failing to adapt to changing contexts, or too loose, potentially allowing unauthorised access.

A new programming language specifically designed to address these challenges could significantly improve the flexibility of designing access control policies. Such a language could allow for more nuanced and dynamic policies that take into account a wider range of contextual factors. This could lead to more robust and effective security measures, better aligning with the principles of Zero Trust.

1.3 Aims and Objectives

The **primary aim of this research is to enhance the integration of context awareness in Zero Trust security models through the development of an improved access control policy domain-specific language** (DSL).

This aim will be achieved through the following objectives:

1. Identify how to capture and express context and risk aware access control for ZT, based on existing approaches.
2. Map the requirements identified in the previous step to a set of access control policy language constructs to enable the expression and enforcement of ZT context aware access control policies.
3. Develop and implement a ZT context aware policy language.
4. Demonstrate the validity of the proposed approach through application to an Internet of Things scenario.

Through these objectives, this research aims to contribute to the advancement of cybersecurity by enhancing the integration of context awareness in Zero Trust and providing a more flexible and effective tool for designing access control policies.

1.4 Contributions

1. **Definition of mechanisms to capture and express context and risk awareness for ZT access control**. This was achieved by conducting a

comprehensive , systematic review of the existing literature and practice to determine the nature of methods and mechanisms to capture context awareness as well as to understand the relationship and interplay between risk and trust in context awareness. Subsequently the knowledge gained was mapped against ZT architectural principles and a set of requirements specifying what is needed to express context and risk aware access control for ZT. This contribution is addressed in Chapter 4 and led to the publication of a Systematisation of Knowledge (SoK) paper.

2. **Design and implementation of a policy language for context-aware access control (CAAC) in Zero Trust**: This involved identifying extensions to the existing PAROLE policy language based on the requirements from the precious step. The PAROLE language implementation is extended to incorporate proposed changes as well as to implement missing language processing components including the symbol handling and language interpreter. This is described in Chapter 6.
3. **Evaluation of ZT context aware access control in the Parole policy system**: This was achieved through definition of an Internet of Things scenario including specification of the PAROLE policies, construction of a scenario testbed and subsequent execution a series of experiments to demonstrate the satisfactory performance of the PAROLE system to express and enforce ZT context aware access control. This is described in Chapter 7.

1.5 Publication from This Research

- Xiao, S., Ye, Y., Kanwal, N., Newe, T., Lee, B., 2022. SoK: context and risk aware access control for zero trust systems. Secur. Commun. Netw. 2022.

1.6 Structure of Thesis

The rest of the thesis is structured as follows. Chapter 2 describes the methodology that is used to accomplish the project. In Chapter 3, it specifically outlines the background and review of literature. Then in Chapter 4, it explores the integration of zero trust and context-aware access control through summarizing previous researches on how context can be used in zero-trust based system. Based on the investigation in Chapter 4, Chapter 5 presents the design of Parole policy language in detail. Then in

Chapter 6, it presents in detail how the interpreter of Parole is implemented. Chapter7 describes the experimental result to show Parole can handle the context correctly. The conclusion and future work are outlined in Chapter 8.

Chapter 2

Methodology

2.1 Research Methodology Fundamental

Research methodology was defined by Brent and Leedy (2016) as "the general approach the researcher takes in carrying out the research project". There are various ways to classify the research methods, while most common taxonomy is quantitative and qualitative research methods. According to Williams (2007),

- **Quantitative Research** was originally developed in physical science (Creswell and Guetterman, 2018). The processes of quantitative research involve the collection of data, using mathematical model to train data, and statistical analysis. Examples of quantitative methods now well used in the social sciences including laboratory-based experiments, simulations, and surveys.
- **Qualitative Research** was developed in social sciences to enable the researcher to develop a level of detail from high involvement in the actual experiences (Creswell, 2009). Examples of qualitative methods are Case Study, Ethnography Study Grounded Theory Study, and etc. Qualitative data sources include observation and participant observation, interviews and questionnaires, documents and texts, and the researcher’s impressions and reactions.

2.2 Research Methodology for the Project

To systematise the existing knowledge at the aforementioned topics, it is necessary to carry out a literature analysis following the Information Systems Research methodology (Okoli, 2015). As Figure 1 (Peffers et al., 2007) guides, the initial step is to identify relevant literature by combining keyword, backward, and forward search in different ICT databases including ACM DL, IEEE Xplore, Springer Link, Web of

Science, and Scopus. These databases can be accessed through their official websites or using Google Scholar search engine and redirect to the related web pages.

2.2.1 Building Foundation

The project initiated with an exhaustive survey of individual keywords in Google Scholar to identify comprehensive or Systematisation of Knowledge (SoK) papers. Notably, the most recent and highly cited papers were selected and reviewed multiple times to gain a broad understanding of each topic. For instance, the paper by Perera et al. (2014) is selected for context-aware computing, while the one by Rose et al. (2020) is chosen for zero-trust networking.

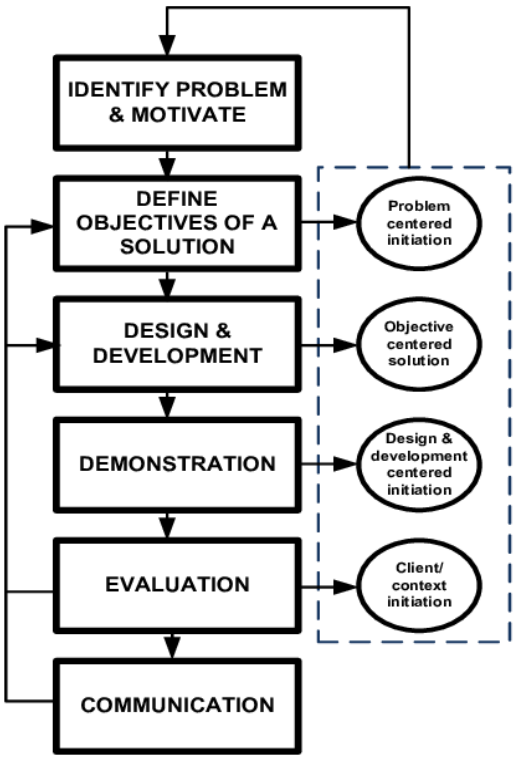

Figure 1 Example of a Science Research Methodology (Peffers et al., 2007)

The subsequent step involved identifying papers whose topics could be combined. Given that the project encapsulates four main themes: context-aware access control models, trust, policy language, and zero-trust architecture, there is room for combining related topics. Context-aware access control models were thus combined with trust, considering both are pivotal subjects within the cybersecurity domain. By employing the same combination methodology, three other pairs were identified: trust with policy language, context-aware access control models with zero-trust architecture, and policy language with context-aware access control models.

The process then moved to application of these combined strings (e.g., "trust" AND " policy language") to locate related research from the previously mentioned databases, restricting the search to papers published after 2010 (or a closer date). These databases were preferred due to their established reputation and vast collection of papers within the fields of information systems, computer science, and cybersecurity. A set of stringent quality criteria was also devised and applied to ensure the selection of high-quality publications relevant to the study's theme.

- Selection of publications beyond a certain timeframe is critical to ensure the content reflects contemporary advancements and state-of-the-art research.
- The number of citations a paper has garnered can serve as a proxy for its influence and relevance. It's prudent to note that older papers should typically have more citations due to their extended presence in the academic sphere.
- The venue of publication carries significant weight and can sometimes supersede the citation count. Research published in top-tier conferences or high-impact journals is often indicative of high-quality work. Thus, an assessment of the publishing venue is essential, employing standard academic metrics such as the Impact Factor or h5-Index, to ensure the credibility and significance of the selected papers.

2.3 Literature Review

Reviewing related survey papers provides a comprehensive overview of the field. However, to delve deeper into a specific area, it becomes essential to refer to individual research studies. Having identified the most relevant papers, the abstract and introduction sections serve as the primary points of focus. The abstract offers a succinct summary of the research, while the introduction elucidates the research rationale and methodologies.

Dynamic context-aware information access in virtual organizations	2009	inc	RBAC(b)	virtual organization	To define access rights of subjects to objects not statically, but to use their properties dynamically for authorization.	**contextual information**: attributes of subjects, objects, environment relational attributes that describe the direct and transitive relations entities.
CA-UCON: a context-aware usage control model	2011	inc	UCON(b)	Usage Control	serving as judgement parameter of Condition.	**Environmental context:** subject context, object context, information communication technology(ICT) context and physical environment **Usage Control Model Things:** S, ATT(S), O, ATT(O), Rights, Authorisati Obligation, Conditions;

Figure 2 an Example of Literature Management

In scenarios where a topic encompasses a variety of categories or models, comparisons within and across these types become necessary. An effective method for comparing

the selected papers involves extracting key information such as the problem addressed by the paper and the employed solution. Recording these specifics in a structured format, like an Excel spreadsheet (as depicted in Figure 2), aids in the process. Allocating a column for classification tags can facilitate filtering of different models at a later stage.

Following this, based on the tags, a summary of shared and distinct attributes for each model can be crafted, which can then be compared with those of other models. This summary can subsequently be integrated into a literature review. Conversely, when a topic pertains to a single subject, a straightforward comparison of differences across papers and summarizing these variances is sufficient.

2.4 Design and Implement

Upon completion of the literature review, the researcher should possess a comprehensive understanding of several aspects. These include: 1) research aim, 2) a summary of related works; 3) identified research gaps; 4) a method to address the identified research problems. In relation to the current project:

Project Aim: This project aims to design and implement an interpreter for the lightweight policy language. The policy language firstly needs to have the ability to control ABAC system under ZTN, secondly it needs to be lightweight but expressive define attributes needs to be evaluate and access policies which reason about the afore-defined attributes.

Related Work: Current research is predominantly focused on developing apt security models that can adapt to the demands of cloud computing and the Internet of Things (IoT), both crucial elements of the pervasive computing era. While these security models have evolved rapidly, very few studies have discussed their practical implementation. Moreover, policy languages, such as those proposed by Kagal (2002) and Damianou et al. (2001), were not specifically designed for Attribute-Based Access Control (ABAC). "eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01" (2017) is the language most often used. It was created using XML. Nevertheless, this also means it has the same problems, like its overly complicated language rules.

Research Gap: The evolution of pervasive computing has intensified cybersecurity challenges. Privacy breaches have become more common, digital assets are at higher risk from hackers, and unseen computer profiling has become remarkably precise, among other issues. To safeguard data's confidentiality, integrity, and availability (CIA), researchers have introduced access control models. These models grant a subject access to an object only if the subject is authenticated and authorised. The most prevalent model is the role-based access control model (RBAC), which grants permissions based on the subject's role within an organisation. However, the traditional RBAC model no longer suffices given the dynamic nature and increased interconnectivity of IoT devices. This paradigm shift from on-site to teleworking (Harris, 2003) has changed the trust dynamics, as even internal sources cannot be trusted unconditionally. Consequently, the concept of Zero Trust Network (ZTN), which operates on a "Never trust, always verify" principle, has emerged. ZTN authenticates requests both from within and outside an organisation. The predicament arises from the fact that RBAC supports only one-way authentication (subject to object), which does not meet the stringent requirements of ZTN. The recently proposed Attribute-Based Access Control (ABAC) (Computer Security Division, 2016) aims to evaluate all pertinent attributes of session participants and meets the needs of ZTN. However, the policy language used to control ABAC systems is complex and prone to errors. This project, therefore, aspires to design a policy language for ABAC systems that simplifies control and reduces the likelihood of mistakes.

Methods: Initially, the project requires gaining a robust understanding of parsing theory and becoming familiar with a parser generator tool named ANother Tool for Language Recognition (ANTLR4) (Parr, 2013). This is followed by the crafting of the policy language syntax using ANTLR4. The subsequent step involves leveraging the ANTLR4 parse tree visitor pattern to navigate the auto-generated parser, which serves to implement the semantic function of each grammar rule. Another phase involves the development of an interpreter to execute the access control policies. The interpreter's efficacy is assessed through a series of test cases and an evaluation of its processing speed. This entire procedure may undergo several iterations until it aligns with the set expectations.

Chapter 3

Background and Literature Review

3.1 Introduction

This chapter will review the literature about zero-trust security and context-aware access control models, and explain the relationships among the topics in order to provide a general background about the project.

3.2 Zero Trust Security

This section will first introduce the reason why zero trust (ZT) security comes out, in other words, the challenges and shortcomings of legacy perimeter-based security in modern network environment. Then, it will talk about the basic concepts of ZT and its application – Zero Trust Architecture, including techniques to achieve ZTA and real-world examples of ZTA.

3.2.1 Challenges of Perimeter-security

The emergence of ZT security is to make up the deficiency of traditional perimeter-based security paradigm. Perimeter-based security gives those whose network address is inside the network perimeter default trust and freedom to move unhindered throughout the network. Conversely, the outsiders without internal network address are considered untrusted or even hostile. A typical presentation of perimeter security is shown in Figure 3 (Gilman and Barth, 2017).

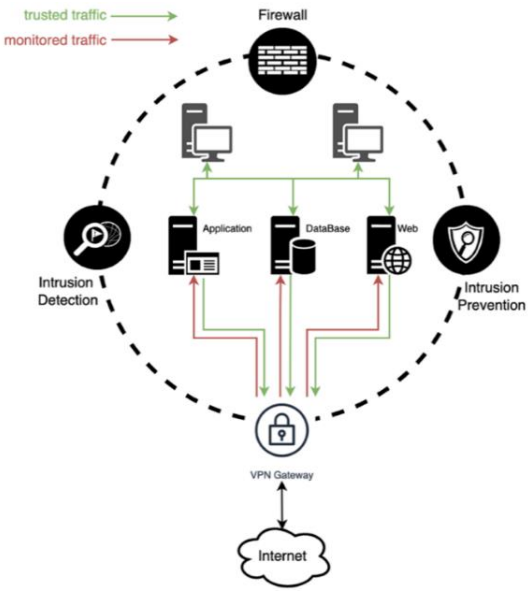

Figure 3 Perimeter-based Security

This security scheme implies that:

- The **outsiders** have difficulty to access inside resources as they are untrusted. The outermost network perimeter is deployed a series of security products, such as firewalls, intrusion detection system, and intrusion prevention system, to enhance the guard. Commonly, every access from outside needs to be verified and monitored to ensure malicious behaviours cannot be applied.
- The **insiders** can access the resources with little efforts because it is believed that the threats have been kept outside of the perimeter. Therefore, the barrier of each network segment need not to be strong, and the monitor of network traffic is not strict.

However, as the network paradigm changes significantly, the attack surface of enterprise has been enlarged accordingly. The perimeter-based security has shown its inadequacy to handle:

- **IoT and Edge Computing.** Network resources used to be expensive, and the enterprise network application is relatively simple. Therefore, enterprise chose to deploy network infrastructure on premise. However, with the development of key enabling technologies of IoT and Edge Computing, such as wireless network, cloud computing, and smart devices, enterprises are embracing distributed computing through deploying resources in the cloud.

Obviously, the distributed environment in turn is blurring the clear dividing line between intranet and other networks.

- **Bring your own device (BYOD)/Working from home.** The trend of BYOD further enlarges the difficulty of maintaining the security perimeter. Especially under Covid-19 epidemic, working from home has become normality, which has turned the perimeter from static to dynamic one. Therefore, it is costly and slow for the security products on the perimeter needs to reconfigure every time when there is a new device has joined in.
- **Cross-enterprise business mission.** As the scale of the enterprise keeps growing, there emerges cross-enterprise business missions which require the resource interaction among the stakeholders. However, this requirement to some degree grants trust to the outsiders. This action violates the intention of perimeter security which believes the perimeter can prevent the suspicious requests from the outside and entangles the perimeter with other companies (Cunningham et al., 2019).
- **More sophisticated malware.** Malwares were primitive, so that the security products were effective to detect threats and prevent risks. However, malwares have made such significant evolution in its sophistication and infectious ability that goes beyond the smartness of anti-virus software. This means that the security products deployed at the perimeter and in the enterprise may not be adequate to detect sophisticated attacks such as Advanced Persistent Threats (APT) (Joint Task Force Transformation Initiative, 2011), ("Equation Group: Questions And Answers," 2015) and data breach can thus happen more easily.

3.2.2 Zero Trust Concepts

To mitigate the limitations of perimeter-based security, Kindervag (2010) in Forrester Research Group envisioned a concept called "Zero Trust". ZT is a direct name indicating that no trust should be granted by default without verification. As the slogan says: " *Never trust. Always Verify* " Rose et al. (2020) provided an operative definition for ZT:

" *Zero trust (ZT) provides a collection of concepts and ideas designed to minimize uncertainty in enforcing accurate, least privilege per-request access decisions in information systems and services in the face of a network viewed as compromised.* "

More specifically, ZT contains three basic concepts:

- **Explicit verification on every request**. From the perspective of ZT security, every request is untrusted regardless of its network address, until it is fully verified based on all available data points. In other words, verification should always happen before granting access permissions.
- **Adopt principle of least privilege**: To reduce the risk, one of the methods is to reduce the impact of the threat. Through **principle of least privilege** (PoLP) (Mayfield et al., 1991), the system can control the access to just-in-time and just-with-access. In the meantime, it can minimise the infectious radius, or say consequence severity once there is compromised computer in the system.
- **Inspect and log every traffic**: Another method to reduce the risk is to reduce the possibility of threat emergence. This needs the system to continuously inspect and log every egress and ingress for the sake of finding and reacting to the potential suspicious activity.

Table 1 Perimeter-based Security vs. Zero-trust Security

Factors		**Perimeter-based Security**	**Zero-trust Security**
Perimeter	**Size**	Big, enclosing whole network	Small, down to every segment
	Dynamicity	Relatively static	Dynamic / Flexible
Trust Granting		Default trust to intranet address	Trust after verification
Risk Mitigation		Security products like firewalls at perimeter	Micro-segmentation, Software-defined Network (SDN), Access Control, Other security products
Access Method		VPN, intranet	Policy Enforcement Point

A comparison of four common factors of both perimeter-based security and ZT security is presented in Table 1. It shows, the former focuses on preventing the network enclosed by the perimeter from threat penetration, while ZT security provides fine-grained protection down to every workload. It also implies that there are ono direct connections between resources and the requester in ZT security. In conclusion, under current network environment, ZT security provides a more comprehensive protection to data resources than perimeter-based security does.

3.2.3 Zero Trust Architecture

This section will introduce how the ZT concepts are used to build the Zero Trust Architecture (ZTA). More specifically, it first covers the approaches and techniques that will be used to achieve ZTA. Then, it will show the roadmap for an enterprise to transfer from legacy security model to ZTA. Finally, a real-world ZTA example will show the priority and criticality of ZT security.

Zero Trust Architecture Basics

Rose et al. (2020) also offered an operative definition for ZTA. It describes ZTA as the enterprise's overall cybersecurity plans that are specially designed based on ZT concepts. Typical ZTA logical components are shown in Figure 4. The picture shows basic relationships and interactions among logical components of a ZTA. The core components are two separate planes – control plane and data plane. Their functionalities are different from those of Software Defined Network (SDN). In ZTA, the control plane is used to control the accessibility of a request, while the related application data is communicated on the data plane. In control plane, the Policy Engine (PE) makes the ultimate access decision for an access request, and Policy Administrator (PA) controls the Policy Enforcement Point (PEP) in the data plane to establish or shut down the protected communication path between a subject and the target resources. They jointly form the Policy Decision Point (PDP). As for, the workflow in data plane, it reflects that the subject is untrusted, until it passes the PDP verification, then it can access the target resources through the security tunnel that are established by PEP. Besides the core components of a ZTA, other data sources, such as activity log and Continuous Diagnostics and Mitigation (CDM) system, can provide inputs to security policies used by the policy engine when making access decisions.

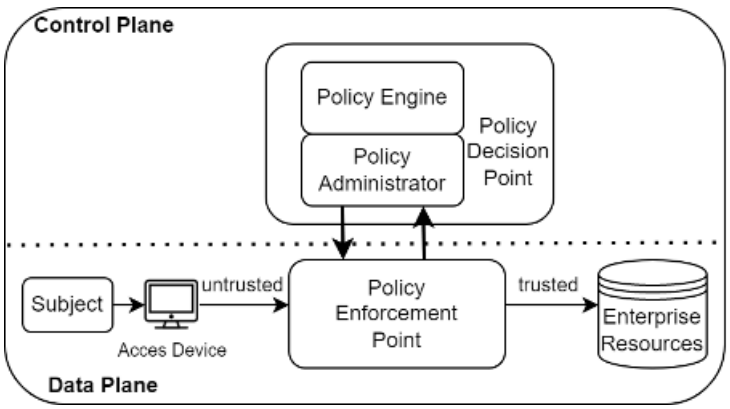

Figure 4 ZTA Logical Components (Rose et al., 2020)

Approaches to Achieve ZTA

There are three mainstreaming approaches to achieve ZTA:

Enhanced Identity Governance. ZT requires identity-centric verification. The most direct way to achieve ZTA is to enhance the identity governance. The idea is much like the enhanced role-based access control (RBAC) (Kulkarni and Tripathi, 2008). As Figure 5 shows, they both have a focus during authentication phase. Enhanced RBAC ensures the role of the subject in the organisation and contextual information will never surpass the permissions inherent to the role's authorised action. ZTA focuses the use of contextual information to verify the identity of the subject. The identity is generally ensured by the available attributes of the subject, such as credentials (username and password), location, device, department, and the position in the company. Access policies are developed according to the permissions about resources, and the permissions will be granted to the identity only if it has been verified. This approach works well with BOYD and visitor access.

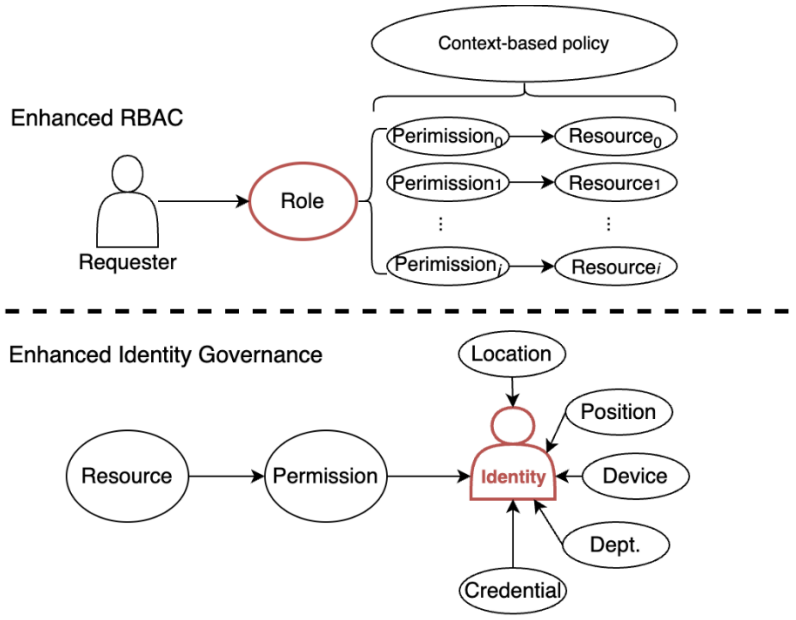

Figure 5 Comparison of Enhanced RBAC and Enhanced Identity Governance

Micro-Segmentation (Vincentis, 2017)**,** (Sheikh et al., 2021)**.** Segmentation occurs between more than two different kinds of network with each deployed security guard at the perimeter. Perimeter-based security is a realisation of segmentation that focuses on controlling the north-south traffic. East-west traffic that happen within the outermost perimeter are also protected through segmentation but beyond inspection. Therefore, the problem is the segments in the security perimeter are too big to

effectively protect the resources within, in other words, the network visibility is not complete in practice. Micro-segmentation therefore aims to provide complete network visibility and fine-grained security by shrinking the size of segments, which means to reduce the number of resources in each segment. It worth mentioning that micro-segmentation can both operate at link and application layer.

The link layer micro-segmentation is achieved through intelligent gateway. According to Kindervag et al. (2010), enterprise can use smart gateway, such as next-generation Firewall (NGF) or intelligent switch, to serve as the network segmentation gateway (SG). Figure 6 shows what a conceptual SG looks like. A single SG integrates the features of several security products, so besides the features of a regular gateway, it can also *i)* properly segment the network according to the functionality of the workloads and allow re-configure; *ii)* enforce efficient security policies with high-speed interfaces; *iii)* uniformly manage the network that complete network visibility.

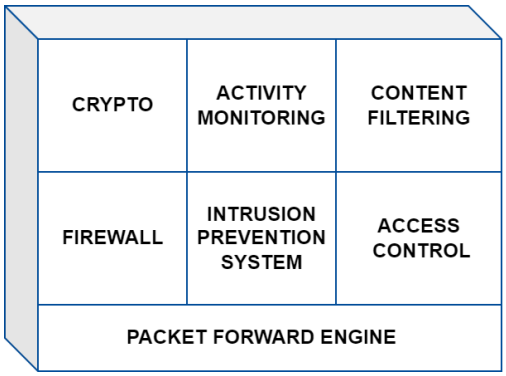

Figure 6 Conceptual Segmentation Gateway (Kindervag et al., 2010)

As for the micro-segmentation achieved in application layer, it is generally achieved using network virtualisation (NV), (Chowdhury and Boutaba, 2010). Morden data centres have dramatically benefited from computer and storage virtualisation. This innovates the network professionals to seek the solutions for NV in order to achieve higher workload mobility and response speed to business requirements. More specifically, NV decouples network hardware from software, which means it can reshape and combine several networks into a software-defined virtual network (VN) to provide Layer2 to Layer7 network, without reconfiguring the network infrastructure. This feature is highly compatible with the goal of micro-segmentation, as NV is easy to reconfigure to suit the security requirements of the organisation.

Software Define Perimeter (SDP) (Moubayed et al., 2019), ("Software-Defined Perimeter (SDP) and Zero Trust," 2020), ("Software-Defined Perimeter (SDP) Specification v2.0 | CSA," 2022). SDP is also known as "Black cloud". The black here means the network components are undetectable under common circumstances. Therefore, the system is free from basic network attacks as the hackers have nowhere to move around after exploiting the system. On the other hand, the connectivity of SDP is established following the need-to-know model, which means the resources are unavailable to the users until the user has been proved that he/she is eligible to access and need to access. This can be achieved by leveraging software-defined network (SDN) to provide the underlying network programmability and flexibility that are required to implement an SDP.

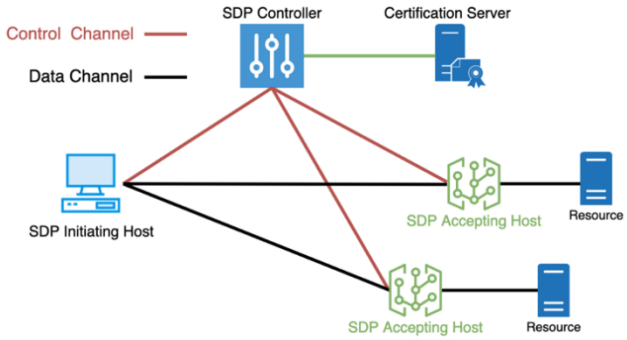

Figure 7 Logical SDP Architecture

The logical architecture of SDP is showed in Figure 7. The SDP controller servers as the brain to control fine-grained connectivity between SDP Initiating Host (IH) (in SDP client) and SDP Accepting Host (AH) (in SDP gateway). The workflow is as following:

1. On receiving request, the controllers go online and establish connection with available certification server.
2. According to the request, the controller establishes Mutual Transport Layer Security (MTLS) tunnels with available SDP AHs. These AHs will only respond to the requests allocated through controller.
3. SDP IH establishes MTLS tunnel after SDP IH passing the verification.
4. SDP controllers determine a list of available AHs and inform them to accept the communication from IH.
5. SDP controllers send the list of determined AHs' IP addresses to IH, and then IH establishes MTLS tunnels with AHs accordingly.

The above three approaches can mix and match according to the security requirements, network, and budget. It worth mentioning that ZT security is not conflict with perimeter-based security, one company can still choose to establish ZTA while reserves the outermost security perimeter that will not violate the ZTA tenets.

While all three approaches can be used to zero trust architecture, they each have their focus and strengths. Enhanced Identity Governance focuses on verifying the identities of users trying to access resources. Micro-segmentation focuses on controlling access to individual or groups of resources placed in different network segments protected by a security component. Software-Defined Perimeter focuses on the dynamic creation and management of network connections based on user authentication and context. For a comprehensive Zero Trust architecture, an organisation might consider implementing all three strategies in a way that works best for its specific needs.

ZTA Maturity Model

ZTA does not happen overnight. Especially for the organisations that have adopted the legacy security strategy, it is unrealistic for them to replace the whole network infrastructure and services. Instead, the organisation needs to consider firstly to deploy ZT principles to protect most valuable and confidential assets, then gradually move to less important ones. This means that for a long time an organisation will run in a hybrid ZT and perimeter-based model. Generally, the journey for a company's security strategy and network architecture to migrate to ZTA is described as ZTA Maturity Model. Listed a group of leading ZTA platform providers such as Microsoft and Cisco. According to Cunningham (2018), there are five pillars of ZTA plus the extra automation and orchestration, and visibility and analytics. Table 2 summarises each pillar's changes in each stage to full ZTA, according to their published their proposed ZTA Maturity Models, ("Evolving Zero Trust," 2021; "Getting Started with Zero Trust Access Management Trust Begins with Secure Identity | Okta," 2021; "Zero Trust Maturity Model | CISA," 2023).

Table 2 ZTA Maturity Model

Stage / Factor	1 (Traditional)	2 (Initial)	3 (Advanced)	4 (Optimal)
Identity	On-premise identity; Numerous passwords; Single factor authentication.	Hybrid on-premise and cloud identity; Using SSO; Using on-premise MFA.	Federated identity with cloud and on-premise; Fine-tunning policies across all user groups.	Automated and continuous detection on identity.
Workload	Access based on network address; Limited integration with transaction flows.	Integrated with transaction flows; Access based on identity.	Strong integration with transaction flow; Access based on context.	Automated integration with transaction flow; Access based on risk.
Network	Large segments; Lack secured communication tunnels; Limited visibility and integration with different network environments.	Hybrid large segments and micro-segments; Secured communication tunnels to high value assets; Detection on suspicious ingress and egress.	Micro-segmentation; Secured communication tunnels to every asset; Detection and log on every ingress and egress.	Automated threat analysis on every encrypted ingress and egress.
Device	Limited visibility to device compliance; Simple inventory of devices.	Registered devices to employ compliance; More specific inventory of devices;	Complete inventory of devices; Verification of device security posture.	Automated and continuous detection on security posture.
Data	Not encrypted; Simple inventory of data.	More specific inventory of data; Valuable data encrypted.	Data encrypted at rest; Complete inventory of data.	Transported data support on-the-fly encryption.

3.3 Context-awareness and Access Control Models

This section will cover 1) context-awareness, 2) access control models, and 3) the combination of them. It will start with the basic concepts required by each, through reviewing and comparing related research works.

3.3.1 Context-awareness

Weiser (1999) came up with a concept called ubiquitous computing indicating that connected devices will be so ubiquitous that people will hardly notice their presence. To achieve ubiquitous environment, the first factor needs to be considered is the number of connected devices, and the adoption of IoT has largely boosted the deployment of connected devices and the speed is continuing climbing. On the other hand, ubiquitous environment also needs to be in line with human beings' instinctive reactions, which implies that the connected devices need to be smart enough to properly respond to the upcoming accidents and plans according to the changing environment and conditions. This kind of ability is described as context-awareness.

Context Related Fundamentals

Before further introduction about context-awareness, it needs to be clarified what is the context in context-awareness. There have been many researchers providing the definition of context, for example:

- Schilit and Theimer (1994) highlighted the importance of location information to enable the adaptivity of software, context is therefore described as the location of use and nearby people and objects.
- Abowd et al. (1999) provided an operational definition to describe as "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*"
- Abowd and Mynatt (2000) concluded "five Ws" of minimal set to represent context: "Who" includes the subject and other related people in the environment; "What" indicates the interpretations of people's activities; "Where" generally indicates the locations or continuous perceiving of

location changing; "When" means the start time or duration of a particular human activity; "Why" describes the reason of doing a thing, which can be conclude through the sensing of other contextual information.

- Kayes et al. (2017) defined context as "*any relevant information about the state of an entity or the state of a relationship between persons (as entities) relevant to access control"*.

The definitions of context vary in different situations, so there is not a right one. The above three definitions are commonly discussed ones, and from their descriptions one may infer the fast and continuous development in hardware abilities and more diverse and fine-grained requirements on software applications. For this project, as ZT security provides data-centric protection, it is better to handle context from a data point of view. Therefore, this project will use the definition from Abowd et al. (1999) for the base of later topics.

Categories of Context

Application designers need to consider which kind of context to use in their application. A clear classification helps the designer to make decisions quickly and precisely. Researches focusing on context-aware system have offered some categorisation methods:

- Schilit et al. (1994) discussed context from three aspects: "where you are" includes related location information, "who you are with" not only describes the identity of the people around the subject but the social situations, and "what resources are nearby" includes all available resources located in a specific distance from the location of the subject.
- Abowd et al. (1999) directly indicated four primary types of context: time, location, identity, activity. These primary contexts not only characterise the situation of a specific entity but also serves as lead to more complex context.
- Henricksen (2003) noticed the previous static/dynamic abstraction of context is less effective for context-aware applications. Therefore, Henricksen provided a more specific classification according to the reliability and frequency of change. The context is classified into following four types:

- **Static:** static context is invariant information that will not change its value overtime. Thus, it is considered high reliable.
- **Profiled:** profiled context is less reliable compared to static one, because it changes with low frequency.
- **Sensed:** sensed context can change every time it is detected, so it is generally unreliable, and its value is stale-prone.
- **Derived:** derived context is calculated through logical operations on above three directly detected or stored context. Its value is updated as each primary context changes, therefore is reliability is on the less reliable primary context.

They are developed for resolving inconsistency and supporting various context management tasks.

- Bunningen et al. (2005) proposed two classification directions, one is classifying context by how the context is operationally acquired, modelled, and manipulated. The other conceptually distinguishes context from each other based on the its meaning and relationship with the subject and other contexts. Perera et al. (2014) also came up with a categorisation of context based on the work of Bunningen et al. (2005) and Abowd et al. (1999). A single context in Perera et al.'s categorisation can be classified to primary or secondary context from operational perspective, or can be counted in one of time, location, identity, and activity from conceptual perspective.

Context-aware describes the ability for a system to discover and respond to the changing context surrounding the subject. Abowd et al. (1999) also provided a widely accepted definition for context-aware:*"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."* Context-aware as an important feature of pervasive system is not only widely researched on providing higher Quality of Service (QoS) but fine-grained access control. More specifically, context-aware is embedded with access control models to provide higher adaptivity and improve access decision accuracy under pervasive environment.

3.3.2 Access Control Models

Access control (AC) aims to protect the confidentiality-integrity-availability (CIA) of data. One can see it as the customs of the information system. AC models are used to systematically formalise the presentation of access policies. However, the complexity and evolutionary speed of cybersecurity paradigm are always climbing, which leads the AC models to adapt according to the new security requirements brought by the highly dynamic pervasive environment. The following contents of this section will first introduce some important original AC models, including their definitions, application scenarios, advantages, and limitations. Then, it will discuss how the above factors will change when the AC models are added with the context-aware feature.

Mandatory Access Control and Discretionary Access Control

The first access control model that is to be introduced is **Mandatory Access Control (MAC)** (US Department of Defense, 1985). MAC is defined to restrict access from subjects to objects based on the sensitivity of the objects and the formal clearance to the subjects to access resources at some level of security. Every subject and object in the system is assigned with a sensitivity label that records the hierarchical security level of the object and non-hierarchical classification.

MAC policies hold the following requirements on every access request based on the properties of Bell-LaPadula (BLP) model (Hansche, 2003):

- Simple Security Property: a subject is only allowed to read the objects whose security level is dominated by the subject’s.
- Star (*) Property: a subject is only allowed to write the objects whose security level dominate the subject’s.

As Figure 8 represents, if a user whose security level is credential, then he/she can read the resources with security level lower than his/her, in the picture are classified and public ones. However, he/she cannot read resources whose security level is higher than his/her. This requirement is intuitive that one cannot read resources have higher security level than he/she has, and is summarised as "no read up". And writing permission can be seemed as "no write down. If one malicious credential agent wants to divulge credential secret, the only method is to reduce the security level of the object through writing the content in credential level object to lower ones recursively.

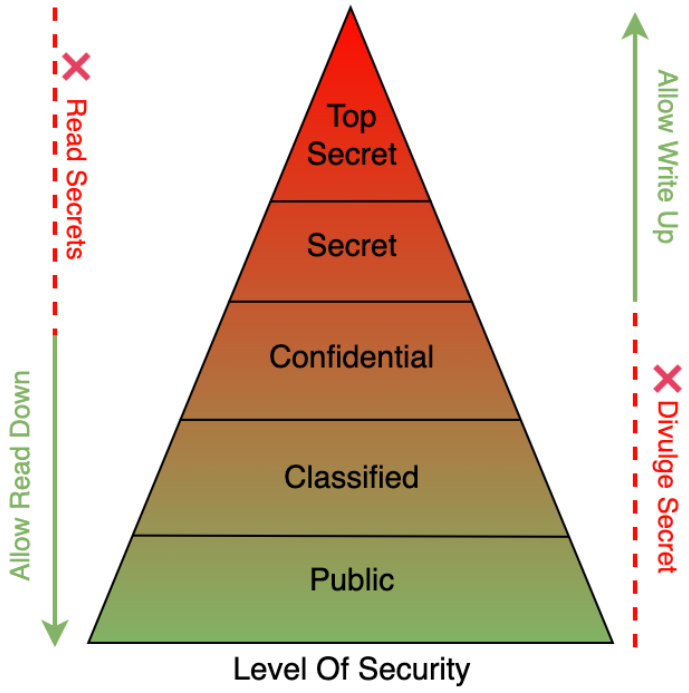

Figure 8 Bell LaPadula Model

Deploying MAC can effectively protect data confidentiality. However, its complexity is subject to the number of sensitivity labels and categories. More sensitivity labels and categories mean more expensive and complex to maintain. MAC therefore is not suitable for the organisations who do not have a clear ranking both on staff and resources, and whose businesses involve too many kinds of services and job functions. Moreover, MAC lacks the dynamicity to respond to frequent personnel turnover. Because every personnel turnover, even the lowest level of position, implies a reconfiguration on sensitivity labels and/or categories.

Discretionary Access Control (DAC) (Hu et al., 2017) is more flexible than MAC. DAC restricts the access from a subject to objects based on the identity and/or group of the subject. Every object in DAC has an owner, and the owner have the discretion to:

- grant and retrieve the one or more privileges to other subjects;
- change the access control policies;
- change the security attributes of existing or newly-created objects.

DAC can serve as an extension to MAC to improve system flexibility. However, for a system that has applied both DAC and MAC, MAC naturally dominates DAC. This is because MAC has much stricter security management. Once access granting conflict

happens, the system will take the strategy provided by MAC. For example, from DAC's perspective, the situation is a specific user is provided with the read and write permissions to a document from its owner; however, from the MAC's perspective, the same thing is expressed as a classified level user is granted read and write permissions by the owner of a confidential level file. This situation violates the principles of MAC, so this kind of actions though is allowed by DAC will finally be rejected.

DAC faces the similar limitations as MAC. The cost of management will follow the growing number of subjects joining in the organisation and creating more resources. The longer resource's access permission list further enlarges the complexity of maintenance.

Role-Based Access Control

Role-Based Access Control (RBAC) (Ferraiolo and Kuhn, 2009) is another classic AC model. Considering the fact that the ownership of resources in many organisations do not belong to the end users, the actual "owner" is the organisation itself. Thus, the AC based on data ownership is not applicable to these organisations, instead, AC decisions are subject to the role for an individual end user undertake in the organisation. There are several benefits to make AC decisions based on the role:

- **Roles are easier to maintain.** Roles are relatively static both on numbers and permissions. The number of employees of an organisation grows with the larger business scale and more complex services. The more frequent changes on personnel implies the harder management on access permissions. However, roles are group-oriented, which means there can have a many-to-one relationship between subjects and a role. There can also have a one-to-many relationship between a role and permissions that supports centralised permission control. Therefore, managing roles takes less effort compared with managing subjects and related permissions.
- **Roles are adaptive to different industries.** A role is designed based on corresponding qualifications and specific responsibilities. Roles in different industries therefore can take on their unique responsibilities, for example, roles associated with hospital includes surgeon, nurse, and pharmacist; roles in a university includes professor, student, and librarian.

- **RBAC protect CIA of data.** RBAC supports both PoLP and Separation of Duty (SoD) (Mayfield et al., 1991) through role. In the chain of RBAC user-role-permission, role-permission assignment happens before user-role assignment, therefore, for every role, its permissions have been limited to perform possible tasks to the role. SoD is achieved through mutually exclusive roles constraints when a sensitive task cannot be completed by only one user. Mutually exclusive roles do not have overlap on their permissions and each user can only be in either group in mutually exclusive roles.

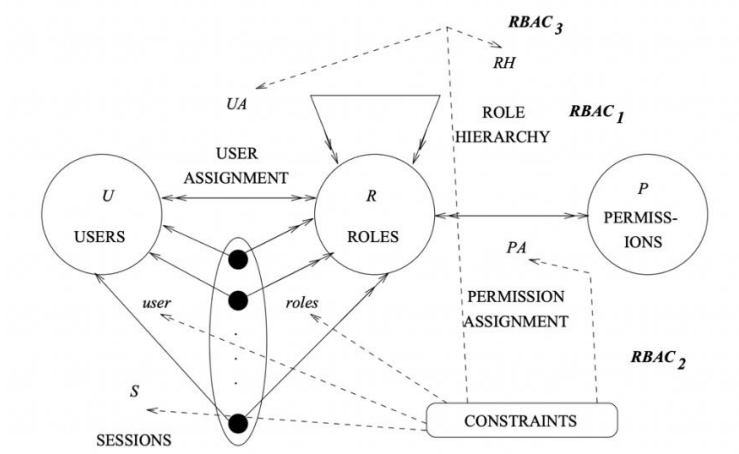

Figure 9 RBAC Models (Sandhu et al., 1996)

Figure 9 contains basic RBAC model and three advanced ones. Basic $RBAC_0$ model contains the four must components and processes respectively to achieve RBAC:

- Users (*U*) is the subject involved in a human-computer interaction.
- Roles (*R*) is the named job function within the organisation that contains the related authority and responsibility.
- Permissions (*P*) is the approval of access request to resources in the system.
- Sessions (*S*) describes the duration that a user is assigned with one or more roles to access resources.
- Permission-role Assignment (PA) describes a many-to-many assignment relationship between role and permission.
- User-role Assignment (UA) describes a many-to-many assignment relationship between user and role.
- *user* maps one session to one user.

- *roles* maps one session to a set of roles.

$RBAC_1$ further introduces role hierarchy to $RBAC_0$ to structure roles according to the hierarchical authority and responsibility in the organisation. For the senior roles in role hierarchy, they naturally inherit the permissions of junior roles, which means that senior roles can do whatever junior roles can do. But junior roles are limited to their assigned permissions because they have relatively more specific working content than the senior ones.

$RBAC_2$ instead add extra constraints to UA, PA, and session control to meet other security requirements. More specifically, in a session, user needs to provide other credentials besides the original username and password in order to be granted a specific role. Permission granting also turns to be stricter. As it is mentioned before, the support of SoD needs RBAC to implement mutually exclusive roles, which requires a single user can only be assigned to either side of mutually exclusive role set. Another example of PA constraint is the cardinality of the role, for example, the number of students in a class cannot exceed 20. $RBAC_3$ combines $RBAC_1$ and $RBAC_2$, which means constrains can also be added on role hierarchy.

Attribute Based Access Control

Attribute Based Access Control (ABAC) is different from above AC models which decides accessibility to objects directly based on the requester's identity or predefined attributes of the requester like roles, (Computer Security Division, 2016). The basic workflow of ABAC model is shown in Figure 10. Firstly, the user sends an access request to ABAC request receiving module, then the request is forwarded to ABAC decision making module. The ABAC decision making module will retrieve the related policy templates according to the request and based on the template ABAC decision making module will collect related environmental conditions and attributes of subject and object. Then these attributes will be evaluated with the rules defined in the policy template. If it passes all of the evaluations, ABAC decision making module will establish a secured communication tunnel between the requester and the requested resources.

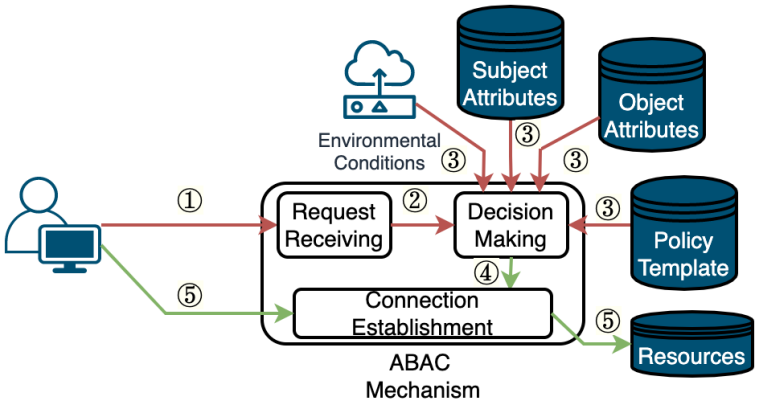

Figure 10 Basic ABAC Workflow

ABAC has some superiorities over non-ABAC models:

- ABAC is more flexible to cope with cross-organisational interactions, as ABAC policies are not identity-centric, and resources are not directly connected to specific roles. For example, if organisation A needs to work with organisation B in a project and an A member wants to access some resources in B, for non-ABAC models, they will need to register an account for A member in organisation B to access the related resources. This action implicitly enlarges the attack surface of organisation B, because B has extended its services but haven't reconfigured the related roles accordingly, which is equal to introduce the potential threat into the system. However, in ABAC, the role of a subject is seemed as a subject's attribute, and an attribute impacts the authentication results but not authorisation.
- Attributes can be recognised across organisations because they are detectable and objective things that can be proven true or false. This helps eliminate the role abuse because of the misuse and improper definition of roles by the administrator. In addition, the growing number of applications and service used throughout the organisation, the higher employee turnover, and the more frequent and deeper contact and cooperation with other organisations can leading to more complex and fine-grained security requirements on AC. This situation forces the administrator to create more roles accordingly, and the roles gradually become too many to maintain, which is described as **role explosion** (Elliott and Knight, 2010).

Implementing ABAC is very hard, regardless of the high scalability it brings. Firstly, it is complex and time-consuming to define and maintain so many attributes and later according to the relationships among attributes and security requirements to specify

related access rules and policies. Furthermore, the source and timeliness of every attribute also need consideration. Attributes that are to be used in access policies need to come from trusted sources within a limited time to promise the correctness of ultimate access decision.

Usage Control

Considering the fact that traditional AC models have limitations dealing with modern dynamic and distributed system, trust management lacks client-side digital information control, and overlaps between Digital Rights Management (DRM) and AC models and policies, Park and Sandhu (2004) came up with a conceptual framework called Usage Control (UCON). UCON systematically merged and enriched the functionalities of above areas to build foundation for achieving well-defined models and policies that provide fine-grained transaction-level controls with respect to mutability and continuity aspects, and privacy issues.

$UCON_{ABC}$ is a core model of UCON. As Figure 11 shows, $UCON_{ABC}$ has eight main components to impact the usage decisions. Except for the subjects and objects and their attributes to represent the existence of themselves, others include:

1) Right describes the privileges of a subject that can exercise on an object. It needs to notice that rights do not exist independent of the access request, instead, rights exist when there is an access request from subjects to objects.

2) **Authorisation** describes the security rules that evaluate whether the subject can exercise some rights on objects or not based on the subject and object attributes. The evaluation can happen before the rights are granted and/or when the rights are exercised. Authorisation rules can also require some attributes to update pre-/on-/post-authorisation.

3) **Obligation** describes the extra mandatory requirements that a subject need to meet before or on exercising a right. Obligation has no overlap with authorisation because attributes do not directly impact the result of obligation, instead attributes are used to decide which kind of obligation to use before or during usage process.

4) **Condition** is the detectable objective environmental and system-oriented attributes that controls which conditional requirements needs to be used in an access request.

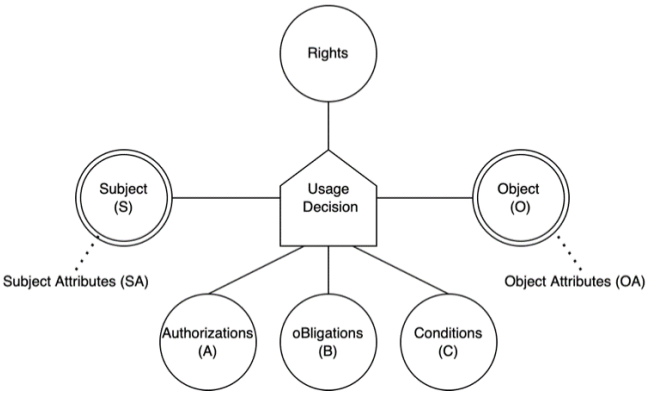

Figure 11 $UCON_{ABC}$ Model (Park and Sandhu, 2004)

UCON provides a more comprehensive perspective to deal with access control in pervasive environment. The benefits of UCON are obvious:

- $UCON_{ABC}$ models and policies further include obligation constraints in usage decision making. Therefore, compared with previous models and policies that only based on one or some of the attributes from subject and/or object, $UCON_{ABC}$ can naturally provide fine-grained access control. Obligations in $UCON_{ABC}$ also implicitly narrow the attack surface of the organisation through defining the prerequisites or ongoing requirements to promise that every transaction is happening under a relatively more secure situation than that without obligation control.
- $UCON_{ABC}$ model supports transaction-based control. This means $UCON_{ABC}$ policies not only evaluate every request based on associated attributes and constraints, but also continually influence if and how the subjects can exercise rights on the target objects based on the update of related attributes. In conclusion, $UCON_{ABC}$ is highly compatible with real-world business workflow because it abstracts the factors that an organisation needs to consider when formulate security policies.
- Privacy issues have rarely been studies in terms of other AC models. However, UCON prevents the privacy from abuse through multi-way control. Previous AC models focus on controlling the communication between the subject and the target object owned by the provider. This situation can be seemed as privacy-agnostic because the requester doesn't have any rights on the object. Reversely, if the subject also holds some rights on an object because his/her private information is contained or related, there

will be three stakeholders in an access request: *1)* **consumer** denoting the requester who want to access an object that doesn't have privacy issues with him/her; *2)* **provider** denoting who holds the requested object; *3)* **identifyee** whose privacy is contained in the requested object and therefore wants to have some control on how the object will be used. To deal with this situation, UCON can provide multi-way control, which requires evaluating the eligibility on the consumer to access and the provider to provide the object under the control of identifyee.

Table 3 compared the mentioned model from four factors. As it shows, the evolution of AC model is accompanied with the growing complexity. In the meantime, the network environment also evolute from static connected computer network scenario to pervasive computing scenario. Pervasive computing is characterised by its huge number of connected devices, highly dynamic and complex network topology, and fast processing speed (Weiser, 1999). The researchers therefore add new features to the original AC models aiming to equip them with the abilities to respond to extra security requirements of pervasive environment. Context-aware feature is one of the most researched features to combine with different kinds of AC models, which is collectively called context-aware access control (CAAC) models. This thesis roughly defines a CAAC model as *the access control model whose ultimate access decision on whether a specific access request is subject to the change of single/multiple contexts.*

Table 3 Comparison on AC Models

AC Model	Basis	Flexibility	Granularity	Impact
MAC	Sensitivity Label of Sub. and Obj.	Low	Coarse	Request-based
DAC	Identity/Group of Sub.	Medium	Coarse	Request-based
RBAC	Sub.'s role in the organisation	Medium	Coarse	Request-based
ABAC	Attribute of Sub. and Obj.	High	Fine	Request-based
UCON	Attribute of Sub. and Obj.; Authorisation; Obligation; Condition;	High	Fine	Transaction-based

3.3.3 Context-aware Access Control Models

The compatibility of context-aware and AC models and applicability of CAAC are apparent from its widespread research. This section will review the state-of-the-art literature about how different AC model uses what context(s) to achieve context awareness.

MAC -Based CAAC

Jafarian and Amini (2009) proposed a Context-Aware MAC (CAMAC), which not only preserves the characteristics of MAC to protect data confidentiality and integrity, but transfer AC policies and security label of entities from static to context-sensitive. To deal with pervasive scenarios, CAMAC can define fine-grained access policies which not only include security labels and discretion, but contextual constraints like time and location or systematic information. Security labels in CAMAC including confidentiality and integrity level also update according to the change of contextual values.

RBAC-Based CAAC

Initially, researchers thought about how time (Bertino et al., 2001) or location (Bertino et al., 2005, p.), or combined spatial-temporal information (Aich et al., 2007; Ray and Toahchoodee, 2007) can make UA and PA in RBAC be more adaptive to location/time-sensitive services. Kulkarni and Tripathi (2008) specified a general-purpose CA-RBAC for pervasive computing system. The model is composed of two layers: Context Management Layer ensures the accessibility, integrity, and authenticity of context data that will be used in access decision making; Access Control Layer is responsible for adaptively control the availability of specific role membership and permission binding among roles, permissions, and objects.

Then with the development of network infrastructure and smart devices, more kinds of context are included to expand RBAC functionality on handling different domain-specific scenarios. For example, in order to respond to dynamic Virtual Organisation (VO) environments, Hilbert et al. (2010) applied the ideas from ABAC where dynamic properties of subjects and objects as well as environmental information are interpreted as attributes to impact RBAC UA and PA. Attribute-dependent authorisations not only limit UA because a user's attribute(s) need to satisfy pre-defined requirements to be assigned a role, but support PoLP because a role is not automatically granted all privileges on an object. Schefer-Wenzl and Strembeck (2012) proposed a CA-RBAC model for IT-supported business process. The goal for every business process stays unchanged while the tasks within can vary under different situation. Every task is impacted by the corresponding causes in the cause-and-effect chain. These causes are

interpreted to process-related context constraints, and the model then can assign the specific role to subject for achieving required tasks.

ABAC-Based CAAC

Different from CA-RBAC models where context information is generally used as supporting constraints to achieve fine-grained user-role and role-permission granting, CA-ABAC no long uses role to represent the user's identity, instead, role is seemed as an attribute of the user and it can contain different meaning depending on the scenario, such as in the enterprise role is used as the position of employee, while in social media, role can be the relationship between an account. CA-ABAC uses dynamic and static context information to profile the "security posture" of the user to decide the accessibility to resources.

Covington and Sastry (2006) pointed out that policies designed based solely on users and object are too rigid to introduce necessary administrative complexities, and dynamic context can bring computer systems with the capabilities inherent to human's perception and reasoning about the current situation. Therefore, Covington et al. propose a contextual ABAC model for mobile environment whose authorisation policies are defined using dynamic contextual information existing in the user's operating environment, including attributes associated with subjects, object, transactions, and environment. And these attributes are used to authenticate whether the user is permitted to do the action or not. Picard et al. (2018b) considered that IoT-based systems have the ability to supply lots of diverse attributes collected from real world in a fast manner and describes an enhanced ABAC model using this property of IoT-based systems. The model follows the XACML architecture and is added with a proactive engine which can collect raw data flow derived from the sensors and feed the processed secondary data to PDP. Also, proactive engine could keep PDP stay tuned for new events and react to context changes as quickly as possible. Gupta et al. (2019) noticed that smart vehicles are vulnerable to cyber-attacks as they expose original isolated car system to external environments. Gupta et al. define a two-level ABAC model to protect external interfaces from unauthorised access and prevent in-vehicle communications from overwriting or controlling by adversaries. Also, the model can dynamically cluster smart vehicles into groups based on their location and time and provide group-specific services, such as warning signals to a blind spot or gas discount notifications.

Picard et al. (2018a) examined how secondary context can be derived from sensors for IoT-based ABAC-based CAAC. He proposes the "Proactive Engine," a rule-based engine for access context acquisition, modelling, and reasoning and embeds the engine in an ABAC framework to implement access control expression and enforcement. Chukkapalli et al. (2020) created a smart farming ontology to represent various physical entities like sensors, workers on the farm, and their interactions with each other, as well as context. They develop an ontology-based context-aware ABAC system. Dutta et al. (2020) also proposed a cloud-based semantic web-based ABAC system captures physical context collected from sensed data (attributes) and performs dynamic reasoning over these attributes and context driven policies to execute access control decisions for IoT-based cyber-physical systems.

UCON-Based CAAC

UCON can be seemed as extended ABAC to some degree. Because UCON extends the policy categories of ABAC to not only authorisation, but obligation and condition; the ability to allow attributes to update its value during the session; and impact of decisions from simply returning allow/deny to a request, to continuous monitoring and responding to attribute values change. Actually, these extensions make UCON compatible with context-awareness. Similar to context-aware systems which change the services and notifications to the user when the context is changing, UCON changes its access decisions to resources once detecting attributes change.

Bai et al. (2010) proposed a CA-UCON model called Con-UCON, for Android to provide additional privacy protection and resource usage control. Based on the basic UCON model components, Con-UCON introduced extra *Permission Label* (*PL*) and *Context*. PL is defined similar to the security labels in the MAC that comprise the confidentiality level and integrity level of the data objects, while context is defined as the property of computing system and environment, especially spatial and temporal, that will be used to specify the context constraints in the access policies. Almutairi and Siewe (2011) introduced a CA-UCON model which used subject context, object context, physical environments context, and ICT context to equip the UCON model with the ability to adapt to the environment changes during the execution of an access request. Dimitrakos et al, (2020) described a hybrid architecture which integrated the core components of ABAC to support capture characteristics of things existing in

heterogeneous environments, and UCON to enable fine-tuning the granularity of access policies to respond to the continuous context changing during the session. The architecture is further extended with a Trust Level Evaluation Engine (TLEE) to (re-)evaluate the related trust parameters in the policies.

Chapter 4

Context-aware Access Control in Zero-trust Architecture

4.1 Introduction

The core of ZT model is no default trust, which indicates both resources and requesters need verify trustworthiness of the requester before building connections. Generally, the trustworthiness of resources is manifested as the quality of the resources of the organisations owning the resources, while the trustworthiness of the requester is evaluated based on the risk within the request. Therefore, trust within the ZT model is not inherent attribute but requires ongoing risk assessment during each request session.

For each access request to ZT system, the subject identity, request action and context are subject to continuous evaluation against a comprehensive set of security policies. Traditional access control models often fall short in capturing the dynamic complexity of modern digital environments. The interactions between users, devices, and applications vary significantly across different contexts, which requires more refined approach to security (Kim and Lee, 2017). Therefore, context-awareness are starting to be added as an important feature to access control models in ZTA to deal with uncertainty or probabilistic scenarios.

This chapter explores the integration of context-aware access control and ZT model by first introducing about different context schemes that can be used in ZT system, then it discusses how dynamic context can influence the pivot factors of ZT model - trust and risk, and how these two complementary factors in turn impact the access control decision making. **The aim of this investigation is to determine what modifications, if any, need to be added to the PAROLE language to accommodate ZT context awareness**. Finally, this chapter specifies some main patterns from the CAAC and risk/trust analysis and consider how these may apply to ZT.

4.2 Understanding Context-Awareness in Zero Trust

This section discusses the application of context-awareness access control in the ZT paradigm. Firstly, it talks about different categories of context that can be leveraged within the ZT framework. Then, the discussion extends to the integration of trust and risk within access control, further enriching the understanding of their functional roles within ZT.

4.2.1 User-centric Context

User-centric context focuses on the attributes, behaviours, and credentials of the user that will be used in decision-making process of access control. This context is particularly relevant for Enhanced Identity Governance (Buck et al., 2021), where the system needs to evaluate the role and authentication of users. Some typical user-centric context attributes are listed below:

- **Identity:** This context includes the user's unique identifier (e.g., username or user ID), authentication factors (e.g., password, token, or biometric data), and affiliation (e.g., department or team within the organisation) (Ferraiolo et al., 1999). Identity attributes are essential for determining the user's role and authorisation level in the system.
- **Behavioural:** This context includes the attributes that provide information about the user's past actions, interactions, and access patterns within the system. By analysing the behavioural attributes, such as login frequency, resource access history, and access duration, the system can evaluate the trustworthiness of users and identify potential anomalies or security risks (Su, 2010).
- **Trust and Reputation:** This context includes the attributes that quantify the level of confidence the system has in the user, based on factors such as user's historical compliance with security policies, peer reviews, and external trust assertions (e.g., PKI certificates). Trust and reputation attributes can help make more informed access control decisions, particularly in cases where the user's identity or behaviours may be ambiguous or insufficient (Yan et al., 2015).

- **Spatial:** This context includes the attributes that provide information about the user's physical or virtual location during the access request (Yan et al., 2015). Spatial attributes can help identify potential security risks (e.g., access from unfamiliar or high-risk locations) and enforce location-based access policies.
- **Temporal:** This context includes the attributes that capture the timing of the access request, including the date, time, and duration of the session. Temporal attributes can help enforce time-based access policies (e.g., restricting access to certain resources during non-working hours) and detect potential anomalies in the user's access patterns (Yang et al., 2012).
- **Device:** This context includes the attributes that provide information about the user's device, such as the device type, operating system, security configurations, and the presence of security software (Banks et al., 2021). Device attributes can help evaluate the security posture of the user's device and enforce device-specific access policies.

The utilisation of user-centric context attributes fortifies ZT framework, underlining the paradigm shift from a static, perimeter-based security to a dynamic, context-aware one. User-centric contexts ensure that the access control decisions are not solely dependent on static attributes, but also consider the dynamic attributes. These dynamic attributes together encapsulate the situational context of a user's access request, thus offering a fine-granular assessment of potential security risks. By continuously monitoring and evaluating these user-centric context attributes, anomalous behaviours can be detected promptly, thereby enabling swift response to potential security threats. Embedding user-centric context into security policies not only enhances the adaptability of the ZT model to dynamic risk environments, but also augments its capacity to offer specialised security measures for different application scenarios.

4.2.2 Environment-centric *Context*

Environment-centric context contains the characteristics of the environment where the access request occurs. This kind of context is crucial for Micro-Segmentation based approaches (Sheikh et al., 2021), which aims to divide the network into smaller segments and avoid direct connection between two segments. Environment-centric context include:

- **Network Conditions:** This context includes the attributes that describe the current state of the network, including the network traffic load, latency, and available bandwidth (Ahmed et al., 2016). By considering network conditions, ZT access control can make more informed decisions about granting access to resources, such as limiting access during periods of high network load or prioritizing access for critical applications and services.
- **Network Topology:** This context includes the attributes that describe the structure and organisation of the network, including the distribution of network nodes, connectivity between nodes, and network segments. By including network topology, ZT access control can enforce fine-granular policies that take into account the relationships between different network elements, such as restricting access to specific subnets or isolated segments (Yao et al., 2017).
- **Service Dependency:** This context includes the attributes that describe the relationships and dependencies between various services and applications within the network environment (Shameli-Sendi et al., 2018). Understanding service dependencies can help ZT access control make better risk-aware decisions when granting access, as it can consider the potential cascading effects of allowing access to one service that may depend on others. This information can also be used to prioritise access to critical services during periods of high demand.
- **Business Continuity:** This context includes the attributes that provide information about the dependencies between business process and software/hardware assets (services), as well as their criticality to the organisation's business continuity. By considering these dependencies, ZT access control can enforce access policies that prioritise the continuity of critical business services and minimise the potential impact of security incidents on the organisation's operations (Sarkar et al., 2022).
- **Security Posture:** This context includes the attributes that provide information about the current security level and states of the environment, including the presence of security controls (e.g., firewalls, intrusion detection systems), security policies, and known vulnerabilities. Security attributes can help ZT access control make more informed decisions about

granting or denying access based on the current risk level and the security consequence (posture changes) on the overall environment (Rose et al., 2020).

- **Resource:** This context includes the attributes that offer a global perspective on the characteristics of resources being accessed. Resource encompasses the resource type (e.g., file, database, application), sensitivity level, and ownership, viewed collectively rather than individually. These resource attributes can support the enforcement of resource-specific access policies at a macro level (Abbasi et al., 2020), ensuring that users gain access only to those resources necessary for their roles, in accordance with overarching security constraints.

Incorporating environmental-centric context attributes with ZT can greatly enhance the effectiveness of the ZT framework by providing a system-level perspective on security. These contexts can provide a holistic graph of the overall system's state and its potential vulnerabilities. For instance, understanding network conditions and topology can help identify weak points or unusual traffic patterns, while knowledge about service dependencies can help prevent cascading failures in case of a breach. By continuously monitoring and integrating these environmental-centric context into access control policies, ZT model can adapt and respond to varying system conditions and evolving threats, subsequently enhancing its robustness and reliability.

4.2.3 Object-centric Context

Object-centric context relates to attributes of accessed resources or objects, which helps ascertain resource sensitivity and potential access-related risks (Hu et al., 2013). This section introduces various object-centric context that can be used in ZT model:

- **Sensitivity level:** This context includes the attributes that describes the confidentiality-integrity-availability (CIA) requirements of the resource being accessed. Understanding the sensitivity level of objects helps ZT framework to enforce access policies based on the potential risks associated with granting access to specific resources, such as limiting access to sensitive data or critical systems (Ferraiolo and Kuhn, 1992).
- **Label:** Resources may be labelled based on their importance, criticality, or sensitivity level within the organisation. These labels can be used by ZT

access control to enforce access policies tailored to the specific requirements of each resource category, ensuring that only authorised users can access the appropriate resources (Hu et al., 2013).

- **Ownership:** The owner of a resource can be an individual, a group, or an organisational role that is responsible for managing and maintaining the resource. Ownership can help ZT access control enforce policies that restrict access to resources based on the user's relationship with the resource owner or their role within the organisation (Sandhu et al., 1996).
- **Access history:** This context includes historical access patterns of a resource, such as the frequency, duration, and user access history. By analysing access history, the system can detect anomalies in access patterns, prioritise access to frequently used resources, or enforce access policies based on historical usage trends (Hu et al., 2013).

Integrating object-centric context attributes into a ZT framework can augment the security and adaptive decision-making of the system. Object-centric attributes refer to properties and characteristics associated with the resources or data that are being accessed. For instance, data sensitivity and criticality attributes can influence the level of trust required for access, potentially necessitating additional authentication steps for highly sensitive or critical data. These attributes provide crucial information for making fine-grained access control decisions, thereby minimizing the risk of unauthorised data access, and ensuring compliance with relevant regulations.

4.2.4 Summary

Understanding the concept of context and its application is paramount for ZT. Although interpretations of context may differ among different access control models and application scenarios, the majority of works depict a significant similarity in its representation. Conventionally, context is viewed as characteristics or amalgamations of features defining the state of one or more entities, and possibly, their interrelations. This definition extends to the domain of access control, where context is information used to delineate the state of access control-specific entities like a user, a resource, and an environment, along with the relationships amongst them.

A variety of entities such as user, object, subject, role, and environment are part of context-aware access control models, varying based on the model. The context in these

models is articulated via contextual conditions or expressions, defined as relational operations on entity attributes, or logical combinations of these conditions. This is a recurring theme in the surveyed works. Furthermore, the term ’situation’ can be seen as equivalent to context and can be represented similarly. Situations are often linked with a mission or purpose and can be represented by data tuples or aggregates. Both context and situation can be primary, deriving directly from the entities, or secondary, inferred or retrieved using primary context data.

4.3 Context-aware Access Control and Trust

While ZT model operates on the principle of "never trust, always verify," the concept of trust remains integral to this model. In ZT model, trust is not assumed or given freely; rather, it’s earned and continually validated. Trust is still required, but it is built based on verified actions, behaviours, and the context in which users, devices, and systems operate. Context awareness is a key element in facilitating this trust formulation. By understanding the situational circumstances of a user or device, such as location, time, device type, behaviour patterns, and data sensitivity, context-aware systems can make more informed decisions about trust (Ceccarelli et al., 2012). This real-time assessment allows for dynamic trust levels, continually adjusting access rights and privileges based on changing circumstances. Thus, context awareness enhances the security posture of ZT by adding an additional layer of intelligence to trust formulation. Generally, there are two methods of using contextual information to calculate trust (Armando et al., 2015). They are listed below.

4.3.1 Trust Indicators

Trust indicators (TIs) represent a wide range of measurable and observable features that contribute to calculate the trustworthiness of an entity, whether it is a user, a resource, or a system. These indicators include a variety of contextual attributes such as reputation scores, roles, security postures, historical behaviours which are highly overlapping with the entity attributes. The diversity of these indicators demonstrates that trust itself encompasses not only reliability and competence but also a range of access control decision making context factors.

Winkler et al. (2007) proposed a model of trust indicators that are based on existing operational risk categories. Risks surging in the operational processes of a firm is

categorised to **operational TIs**, risks caused by human behaviours are denoted as **organisational TIs**, risks stem from external source of the organisation is referred as **external TIs**, risks impact the reliability of the organisations are **financial TIs**, and trust-related information from third-party is integrated to **third-part TIs**. Karthik and Dhulipala (2011) identified a series of parameters and metrics as indicators to calculate trust in wireless sensor network including latency, packet loss and hop count. While in social semantic web, there are several social factors that affect trust judgements (Sacco et al., 2013), they are: identity of the requester; similarity between the use and the requester; relationship between the user and the requester; reputation of the user; and historical interactions.

The specific indicators and their relative importance may vary depending on the context, the nature of the relationship, and individual preferences. Trust calculations often involve a combination of multiple indicators, and their cumulative effect contributes to the overall assessment of trustworthiness. Nevertheless, the implementation of trust indicators requires careful deliberation about privacy, data protection, user experience, and system performance.

4.3.2 Reputation Model

Reputation models play a significant role in calculating trust by providing a means to assess the trustworthiness of entities. Reputation models aggregate, process, and analyse information about past behaviours, experiences, and interactions to generate reputation scores or ratings (Resnick et al., 2000). The reputation of an entity serves as a valuable signal for others to determine the level of trust they can place in that entity.

Reputation models contribute to building trust by providing a basis for initial trust judgments. When encountering a new entity, individuals often rely on reputation information to guess its reliability and credibility. Positive reputation signals can increase the willingness to trust, while negative or absent reputation can raise scepticism. Additionally, reputation models facilitate ongoing trust assessment. As trust is dynamic and can evolve over time, reputation systems continuously update and refine reputation scores based on new interactions and feedback. This enables individuals to adapt their level of trust based on the entity's demonstrated behaviours.

Xiong and Liu (2004) proposed a reputation-based trust supporting framework called PeerTrust to use community-based reputations to estimate the trustworthiness of peers in electronic communities. There are five important parameters of reputations that are identified when evaluating the trustworthiness of a peer: the amount of satisfaction within the feedback ; the number of transactions; credibility of feedback; transaction context factor; and community context factor.

Overall, reputation models significantly influence trust by providing valuable information, social validation, ongoing assessment, and fostering accountability. They enable individuals and systems to make informed trust decisions, reducing uncertainty, and facilitating trustworthy interactions.

4.3.3 Trust-based CAAC

In the realm of context-aware access control, trust is considered as an extra level of assurance an entity administering resources pertaining to the appropriate use of requested resources by a user trust (Armando et al., 2015). This notion has been used in an amount of trust-based context-aware access control (CAAC) strategies.

Bernal Bernabe et al. (2016) posited a multi-dimensional trust-based CAAC model tailored for the Internet of Things (IoT), encompassing parameters like reputation, service quality, security, and interpersonal relationships among IoT devices, such as those under the ownership of a single individual. Utilizing fuzzy logic, the model calculates trust and distinguishes four tiers of trust: **Distrust:** the device will act against the best interests of another; **Untrust:** corresponds to the space between distrust and trust, in which a device is positively trusted, but perhaps not sufficiently to cooperate with it; **Trust:** represents the range where the device ensures a minimum of reliability and acts as expected; **HighTrust:** corresponds to the space where the evaluated entity can be confidently trusted.

Ouechtati and Azzouna (2017) laid the foundation of his Trust-Attribute-Based Access Control (Trust-ABAC) on reputation, incorporating a trust metric from a reputation manager for each subject requesting access. This trust metric is integrated into an ABAC-based CAAC system via a Trust Management Broker framework, which collates and disseminates transaction feedback from each CAAC system. Post every access control request, the access control enforcement (namely a Policy Enforcement

Point, or PEP) assigns a trust rating for each subject, which is relayed to the local broker for trust estimation.

Yaici et al. (2019) advocated for a model for pervasive computing environments. The user's location, time, and device used for access are considered, while trust is ascertained by past behaviours after completing the operations and updating the reports. The model employs a set of pre-defined rules to evaluate different requests and accordingly make access control decisions.

4.3.4 Summary

Trust functions as a crucial determinant in CAAC fundamentally denotes the extent to which one party is willing to rely on another within a specific situation, thereby illustrating the inherent link between trust and context. Trust can be quantified through various measures, encompassing reputational trust, along with the analysis of trust indicators such as security metrics or the employment of trust assertions like PKI certificates. Within the framework of CAAC, trust signifies the resource controller's degree of confidence in the user's responsible usage of the resources in question. It's often integrated with Attribute-Based Access Control (ABAC) through the incorporation of a unique 'Trust' attribute. These insights suggest the increasing importance of trust-based mechanisms in the evolving landscape of access control methodologies.

4.4 Context-aware Access Control and Risk

Risk is defined as "*A measure of the extent to which an entity is threatened by a potential ... event and is a function of ..the impact that would arise … and the likelihood of occurrence*" (Ross, 2018). Risk is material when the value of a transaction is high, or when the transaction has a critical role in the security or the safety of a system (Jøsang and Presti, 2004).

Risk is introduced to CAAC to address challenges of allowing access to resources and information in dynamic environments. The access control system estimates the costs and benefits of giving access for each particular transaction and grants access if the risk is below a certain level. Risk-based access control is more permissive than traditional policy-based systems which do not consider contextual risk in making a decision. Risk-based access control system may include a risk mitigation mechanism

(Armando et al., 2015; Cheng et al., 2007a; Ni et al., 2010) to reduce the risk associated with a transaction to an acceptable level—generally an obligatory action to be taken before or after access is granted (Kandala et al., 2011). Many factors can be included in the risk calculation including contextual, situational/mission, and environmental factors as well as trustworthiness of the subject making the request.

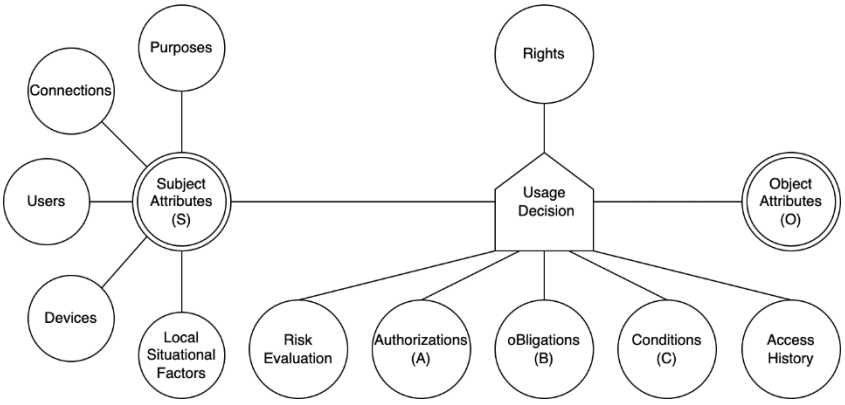

Figure 12 UCON-based RadAC (Kandala et al., 2011)

Kandala et al. (2011) proposed an ABAC framework for risk-adaptive access control based on the UCON access control approach — shown in Figure 12. UCON is an extended access control approach that seeks to unify both traditional access control, i.e., access at the start of the transaction with the need for ongoing control of access to the object during the transaction — what UCON terms *decision continuity*. This latter property is a significant addition to RAdAC (Risk-Adaptive Access Control) as it allows adaptation to changing environment conditions. In Figure 12, the subject concept has been decomposed into a number of components, i.e., users, devices, connections, and purposes. The usage/access control decision process is shown to include Risk Evaluation component as well predicate/rule-based components for *Authorisation* (based on the attributes), *Obligations*, and *Conditions*. Obligations are functional predicates that verify mandatory requirements that a subject has to perform before or during a usage session—in RAdAC obligations can be used for risk mitigation as proposed in other risk-aware research works reviewed earlier. Conditions are environmental or system-oriented decision factors. Kandala incorporates situation related to a particular user or a group of users such as location, as *Local Situational Factors* which are defined as functional predicates that can be evaluated to be true or false. Usage access decision-making is based on all three

rule/predicate components, i.e., authorisation, obligation, and conditions and all can be evaluated pre, during, or post the session.

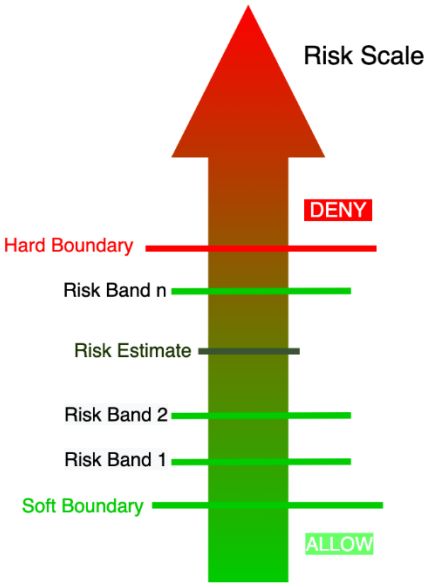

Figure 13 QRAAC (Cheng et al., 2007)

Cheng et al. (2007a) developed QRACC—see Figure 13, a fuzzy logic-based *quantified risk adaptive access control* model for a multilevel security system. QRACC defines multiple bands of risk between the normal binary "allow" and "deny." The quantified risk estimates for any access falls into one of these risk bands. Each band is associated with a decision and a risk mitigation action, e.g., such as increased auditing, application sandboxing, charging the risk to the user; the decision, the action, and band boundaries are all determined according to risk tolerance and can be changed when risk tolerance changes. Risk is estimated on the sensitivity of the information and the trustworthiness of the subject. Ni et al. (2010) investigated the applicability of fuzzy inference for risk-based access control for multilevel security with a banded risk gradation similar to Cheng. However, their focus is more on the design choices of the fuzzy inference systems than defining a model for risk.

4.4.1 Summary

Risk-based access control consists of the systematic process of identifying, analysing, and evaluating risks associated with access requests. This involves considering various factors such as the nature of the request, the user/object role, historical behaviours, and other context. Then the outcomes of the risk assessment are used to determine whether to grant or deny the access request. The combination of both processes

enables risk-based AC to provide an informed understanding of the potential threats and their impact. The adaptive approach enhances threat response by continuously re-calibrating assessment and access decisions to mirror the current risk landscape, particularly when anomalies like new location or device access requests emerge.

4.5 Expressing Context in ZT Access Control Policies

Based on the previous investigation on the integration of CAAC and ZT system, this section describes summarises the main lessons learned with respect to the current. Practise of context and risk awareness for access control. It subsequently describes how these lessons can be applied to accommodate context and risk awareness in ZT systems.

Generally contextual information can be classified as Kayes et al. (2017):

(i) Simple context—a context fact, i.e., an attribute of an entity that specifies the state of the entity based on a single information source, e.g., user identity.

(ii) Complex context—a combination of the values of attributes that characterise the state of one or more entities, based on one or more context information sources, e.g., an interpersonal relationship between two users.

In order to enable Parole with the ability to express context, it needs to support definition of simple and complex contextual expressions. For example,

```
entity.simple-context rel_op value
```

where `rel_op` is a relational operator and the value is some value from the type domain of the context attribute, e.g.,

```
iphone_2.version = 8.
```

A complex contextual expression is a logical composition (`AND`, `OR`, etc.) of a number of simple or complex expressions, e.g.,

```
(iphone_2.version >7) AND (iphone_2.patch_level == up-to-date).
```

The UbiCOSM context model (Corrad et al., 2004) suggested a context can be logical, which identify the logical states of "entities composing a ubiquitous service deployment. Logical states of context depend on logical properties of relevant attributes. Therefore, the context in the policy language needs:

(i) context name that uniquely identifies the context, e.g., Tourist defines a role.

(ii) context type that identifies the context type, i.e., string or int.

(iii) context location reference that specifies the source of context, i.e., provided by the requester, or stored in database.

(iv) a context invocation or activation method, i.e., find("role").

4.6 Expressing Context and Risk/Trust Aware Zero Trust Architecture

Based on the comprehensive studies described in this chapter, this section identifies some of the main patterns from the previous CAAC and Risk/Trust analysis and consider how these may apply to ZTA:

4.6.1 Context

1. Context aware access control models contain several different entities which vary depending on the model. User, object, subject, role, environment are typical entities.
2. Context is generally defined as an attribute or combination of attributes that characterise the state of one or more entities. Context may also include the state of the relationship between entities.
3. Context is expressed in access control policies by contextual conditions/expressions/predicates which are defined as relational operations (`=`, `!=`, `>` etc.) over entity attributes (simple conditions) and/or a logical combination of either simple conditions (complex conditions) or other complex conditions. This mode of expressing context occurs repeatedly throughout the surveyed works.
4. Situation can be considered as equivalent to or a form of context and as such may be expressed by context expressions also.
5. Situation is often associated with *mission or purpose*. In such cases situation may be represented by a *data tuple* or *data aggregate*.
6. Context and situation may be *primary* i.e., emanating directly from the entities or *secondary* i.e., inferred or retrieved using the primary context information.
7. RBAC based context models adapt to dynamically changing context by

varying *role* assignment and/or permissions to match the context.

8. ABAC based context models adapt to dynamically changing context by invoking appropriate rules/policies based on the values of entity *attributes*.

9. Context and situation have some well know examples e.g., spatial, temporal etc. More generally contexts/situations are domain specific and require definition of domain attributes. The most common example found in the literature is the medical/health domain.

4.6.2 Risk and Trust

1. Trust is linked to context by its definition as the degree of willingness of one party to depend on someone or something in a given situation.

2. Trust can be calculated in several ways including reputational trust and behavioural trust as well as from assessing trust indicators such as security metrics and from trust assertions such as PKI certificates.

3. In context-aware access control trust expresses the level of confidence the resource controller has in the user not to misuse the resources being accessed.

4. Trust many is often combined with ABAC through the use of a specific *Trust* attribute.

5. Blockchain based trust systems are being extensively researched to manage trust for decentralised access control for IoT systems. Typically, these also use ABAC.

6. Risk-aware access control balances the trade-off between benefits and potential costs (downsides) of giving a user access to a resource.

7. The benefit (or need) from giving access is often related to the purpose or mission of an entity/situations. Cost is related to misuse of the resource e.g., breach of confidentiality, integrity or availability. Cost may therefore be related to the sensitivity or priority of the resource to the organisation.

8. Risk may be calculated based on many factors (entity attributes) including context. The trustworthiness of the user is often a particularly important factor.

9. Risk-aware access decisions may be binary (allow/deny) but more often are scaled in some way i.e., may give different degrees of access or enforce some form of risk mitigation/trust enhancement actions before giving access. Such approaches attempt to deal with the intrinsic uncertainty and probabilistic nature of dynamic contexts.

10. This ‘scaled’ decision making model is a form of context-reasoning and many

of those proposed technologies (e.g., ontologies, fuzzy logic) have been proposed for risk-based access control.

11. Risk mitigation or trust enhancement actions may be ongoing during the access session. These are often referred to as *obligations* and are included in most risk-aware access control schemes.

4.6.3 Application to ZTA

1. The CAAC formalisms and models explored above provide a rich canvas to capture and express the dynamic policies anticipated by ZTA. Different entity contexts may be appropriate for the different ZTA approaches described in the work of Rose et al. (2020). For example, the Enhanced Identity Governance approach may benefit from an emphasis on User-centric context, while the Micro-Segmentation based approach may benefit from an Environment-centric context focus.

2. In principle either ABAC or RBAC based schemes could be used to provide ZTA access control. In practise ABAC gives more fine-grained control and is the dominant access control approach explored in the literature and in commercial systems. However, it is very likely that RBAC systems will be used to provide primary or secondary context information sources for user centric-contexts.

3. ZTA places very strong focus on user credentials and device state when making access control decision. As a result, the main ZTA trust mechanisms proposed in the literature are trust indicators and trust assertions (Rose et al., 2020), (Osborn et al., 2016a), (Ward and Beyer, 2014). Behavioural trust based on users historical access is also strongly suggested for use – (as alluded to earlier Rose et al. (2020) refers to behavioural trust as "contextual trust assessment). Although not explicitly described reputation-based trust systems could in principle also apply.

4. The line between ZTA trust assessment and risk-aware access control risk assessment is very imprecise. Calculation of access benefit through mission or purpose is not explicitly referenced in the ZTA literature – rather the potential damage or cost arising from subject, object or environment entity sensitivities is the main factor considered in making the access decision. However, as the range of dynamic context increase as ZTA is more widely deployed, it is reasonable to predict the increasing convergence between the two.

5. Real-world ZTA systems access decisions may be binary or scaled. NIST (Rose et al., 2020) defined this as "criteria" vs "score" based where the former permits

or denies access based on the values of a set of attributes while the latter assigns a confidence level based on the values of different attributes and grants access if the confidence value is higher than a given threshold. Access maybe either denied or restricted if the confidence level is too low. Google employed a tiered-trust scheme (Osborn et al., 2016a). In order to access a given resource, a device's trust tier assignment must be equal to or greater than the resource's minimum trust tier requirement.

6. The calculation of ZTA trust confidence levels can be based on any of the context reasoning techniques outlined in the work of Perera et al. (2014) such as fuzzy logic, probabilistic logic, ontology-based or machine learning. The numerous fuzzy logic risk-aware access control approaches could be adapted for ZTA trust calculation e.g., Manchala's (Manchala, 2000) fuzzy trust matrix approach could provide a comprehensive access control approach that would map well to both NIST and Google's BeyondCorp ZTA architectures. Moreover Armando's the trust and risk aware access control framework (Armando et al., 2015) pointed out an approach to implementing a trust/risk evaluation system.

7. Continual monitoring of the access control decision is a key ZTA tenet and thus the use of risk-aware access control obligation type mechanisms will be required as part of ZTA access control.

8. Since ZTA is essentially a set of concepts and ideas rather than a functional architecture it can be applied to many different enterprise information systems configurations including IoT, cloud, remote working etc. Moreover, diverse technologies such as blockchain could be used in the implementation. The focus is traditional enterprise – which may include cloud and remote working component but does not explicitly consider IoT and edge computing. These latter two may require consideration of extra details such as those outlined by Kayes et al. (2020).

4.7 Conclusion

In conclusion, this chapter focuses on the merge of context-aware access control and the Zero Trust (ZT) model. Initially, it provides an overview of various context schemes that can be implemented in a ZT system. Following this, it elaborates on how the dynamic nature of context influences the two pivotal factors of the ZT model - trust and risk. Further, it clarifies on how these mutually supportive elements in turn

affect the decision-making process of access control. It then identifies key patterns derived from the context-aware access control and risk/trust analysis and describes how these patterns are applicable to the ZT model and outlines requirements that must be satisfied by a context and risk aware ZT access control policy language, upon in the remainder of the thesis.

Chapter 5

Specification of PAROLE Policy Language

The CAAC of ZTA needs identity-centric access control policy language that is specially designed for the fine-grained access control. However, currently there only exist policy languages for access control models without the ability to continuous monitor dynamic context changing. Considering the features of CAAC and the "no default trust" principle of ZT network environment, this chapter specifically introduces the specification of policy language – PAROLE policy language – used to realise CAAC model in ZTA. This chapter describes an update of the previous work of Vanickis et al. (2018) in the group. Firstly, it discusses about the requirements that a policy language needs to satisfy for it to process access request in ZT system in a fine-grained manner. Then, it comes to the detailed introduction of the main language constructs of Parole, including their syntactical representation, semantical meaning, and the sequential processes to handle each access request. Finally, it represents a whole Parole script that will be used to handle the access to some resources in the organisation.

5.1 Requirements of a Policy Language

The general requirements for the policy language are proposed based on the analysis of the work of Damianou et al. (2001), Claudio Agostino Ardagna et al. (2008), Seamons et al. (2002), and characteristics of ZTN. The requirements are roughly summarised into following three aspects:

1) For rules and policies:

- Rules and policies are applicable to every requester trying to access the specific resources and making decisions through the ontology and context

of the requester, at the same time, offer service-oriented security with decentralised policy components.

- Rules and policies should have well-defined structure that clearly indicate the propositions to individual or collection of attributes and, in the meantime, obey the principles of least privilege and separation of duty.
- Rules and policies are extensible to add new policy templates for other actions on resources or combined through flexible procedures to apply to a specific decision request. In addition, they are expressive to describe permissions and restrictions, and descriptive to analyse conflicts or inconsistencies existing in the rules and policies.

2) For contextual attributes:

- Considering the nature of access control policies that require a precise, robust, and easily understandable structure (Hu et al., 2017), PAROLE is designed to be a statically and strongly typed policy language instead of a dynamically and weakly typed one. Compared to a dynamic and weak typing language, a statically and strongly typed language provides compile-time checks and enforces type constraint rules at runtime. This means that changes to policies are explicit and require conscious effort to implement, which reduces the potential runtime errors (Gao et al., 2017). Moreover, strong typing also has implications for efficiency and speed, as the system does not need to spend resources at runtime deciding what type a variable is, which is crucial for real-time access control decisions in ZT systems. On the other hand, using a dynamic and weak typing in ZT system must be accompanied by additional safeguards to prevent security vulnerabilities, and it would likely necessitate more rigorous testing to catch any runtime errors (Seixas et al., 2009). In PAROLE, an attribute can only have one specific data type that cannot be modified during runtime, a unique formatted name and single- or multi-value, based upon which the decision of one specific policy is made.
- • PAROLE provides logical and arithmetical operations that can be acted on the attributes for providing necessary evidences in authentication phase.
- PAROLE provides interfaces to fetch attribute’s value from distributed data sources that stores the authentic contextual information ready for used in

access control; to invoke the risk calculation within the request; and to enable continuous monitoring of the requesters' context.

3) For Constraints:

- The constraints should be clearly expressed in rules and policies which describe the extra permissive or preventive requirements that must be satisfied before, during, or after the authorisation is granted.
- The execution of constraints is independent from the policy enforcement.

5.2 Main Language Constructs

Parole language is expressive and extendable enough to handle dynamic context when doing access control decisions and, in the meantime, ensures continuous security during the entire session. There are mainly four functioning components in PAROLE:

5.2.1 Namespace

Namespaces are used to organise the collection of policies, attributes, events, and other syntactic artefacts that make up the policy system for an organisation. Namespaces may e.g., be defined along organisational lines such as business units etc.. Namespaces can be nested, so the programmatic entities that are defined within other namespaces need to be imported before they are used in another namespace. It needs to be noticed that inner namespace does not represent higher priority, it only represents the inclusion relationship between the two namespaces. Syntactically, a namespace is defined as:

```
namespace <<namespace_id>> { … }
```

A namespace is identified by keyword "namespace" followed by a unique identifier in the current namespace scope (same name is allowed in different hierarchical namespace). The curly brace contains other programmatic language constructs.

5.2.2 Attribute

"attribute" block is used to declare the attributes of the policy entities i.e., subject, object/resource, context etc. These attributes can then be directly used in access policies to evaluate the overall context of the requester. It needs to be noticed that the attributes and their values that are used to do authentication come from two sources: one is from the request, which holds basic authentication credentials of the requester

and contextual information that can be provided by the requester side such as date and device information; the other source is the Policy Information Point (PIP) which holds the attributes of the resources and authentication information about the identity that is claimed by the requester. Generally, the declaration of attribute is the same as how a variable is declared in general-purpose language:

```
<<attribute_type>> attribute_name₁, attribute_name₂, …
attribute_nameₙ;
```

Currently, Parole supports four primitive data type: integer (`int`), float number (`real`), string (`string`), bool (`true`/`false`).

5.2.3 AuthRule

AuthRule are language constructs defined in the namespace that contains the specification of the access control policies. Generally, the goal of an authRule is to ensure the identity of the requester which means it complies with the pre-authorisation stage in Usage Control model. There can be several authRules defined in one namespace. Within the authRule, there are programmatic predicates that can determine if the credentials provided by the requester satisfy the requirements. The default value of the authRule is deny, which complies with the principles of ZTA. While if the risk is affordable and requirements are reached, authRule will return allow. This implies that the requester is authenticated and authorised to access the resources. An authRule block is identified by keywork ‘authRule’ and followed by the like:

```
authRule role_name { … }
```

5.2.4 Session

As it is described in the $UCON_{ABC}$ model, `authRule` block has ensured the authorisation of the request before the access request is allowed. The ‘session’ block, function is to ensure the decision continuity of the request during the whole session i.e., during the period the requester has access to the resource. This means extra obligations, or say requirements of context, need to be achieved before granting access and constraints are monitored throughout the session. Contextual requirements mean the specific situation the authenticated requester should be in before access the resources, such as the time to access the resources should be in weekdays and the device used by the requester should be managed by the organisation. Then after the action of accessing the resources has been launched, session block also ensures the

requester keeps in a safe situation when the connection between the user and the resource is still alive. To achieve this goal, the system will check the situation of the requester based on a time interval defined in the session. Once the system detects the situational requirements are not reached, the connection will be cut off.

Basically, a `session` is an intermediate construct between control plane and data plane. Above contents of a `session` all happen in the control plane of the ZTA. While `session` also controls the PEP to build communication channel between the resources and the requester in data plane, after making an `allow` access decision. A session block is defined as:

```
session <<action_name>> { … }
```

A session is identified by keyword `session` and followed by an action name. A series of sessions defined in the same namespace indicates that the allowed actions that can be done on the current resource.

Action indicates the allowed action that can be performed on a resource. Parole supports four kinds of actions:

- Execute: allow the user to execute actions on the resource.
- Read: allow the user to read the resource.
- Write: allow the user to create and edit the resource.
- Delete: allow the user to delete the resource.

It is mentioned above that the risk within the request is also calculated in session. PAROLE uses fuzzy logic to define risks, specifically it uses a Java implementation of the IEC 1331 Fuzzy Control Language, FCL- i.e. jFuzzyLogic (Cingolani and Alcalá-Fdez, 2013). Risks are defined as FCL function block and are invoked by the PAROLE runtime to evaluate policy risks.

This function is invoked by referencing `riskFB` (a risk functioning block). This interface uses fuzzy logic ("Fuzzy Control Programming," 1997) to evaluate the risk against dedicated parameters as inputs. Rather than dealing strictly with the usual "true" and "false" values of classical logic, fuzzy logic uses varying degrees of truth. These degrees are often represented as values between 0 and 1, where 0 represents absolute falseness and 1 represents absolute truth. This is useful when problems that do not have precise, well-defined solutions nor where the data is not strictly binary. The risk assessment in ZT excludes the judgement on crisp sets like: if the user is using

the managed device or not, or if the user is access during worktime or not. These kinds of context can be directly evaluated using logical operators. Instead, the parameters that are used have continuous value or can be discretised into multiple levels based on human perception. Typically, it includes vulnerability severity (low, medium, high), or impact level (low, medium, high) (Jeff Williams, n.d.).

5.2.5 Request Processing Workflow

This section represents how a request is processed in Policy Decision Point (PDP) controlled by Parole. Figure 14 shows the whole processing workflow of a successful request. As it is shown in the figure, the whole process starts with a user sending an access request to some resource in ZT system. The request is interfered with Policy Enforcement Point (PEP) to avoid direct connection between the user and resources. Once PEP has received the request it then dispatches this request to context handler. The request that context handler received contains other information besides the contextual information. Then context handler starts to extract useful context information and output them into a Json file. Now the context is ready to be sent to PDP for further use.

Basically, the required attributes within a request include user credentials, user's intended action, and user's target resource. PDP will first find the related namespace of target resource according to the request. Then based on the user credentials, PDP will invoke the related `authRule` within the target namespace. During processing `authRule`, PDP will ask PIP for authentic data value according to the rules within the `authRule`. After the identity of the user is ensured, then PDP will invoke the related session based on the user's intended action. Generally, at this stage, PDP will also ask for data from PIP, while it may also ask for extra contextual information from the user for risk calculations and continuous monitoring. Finally, if the context evaluations also pass, then PDP will let PEP know its access decision, PEP then will allow data transfer between the user and resource, and the data are further transferred through PEP.

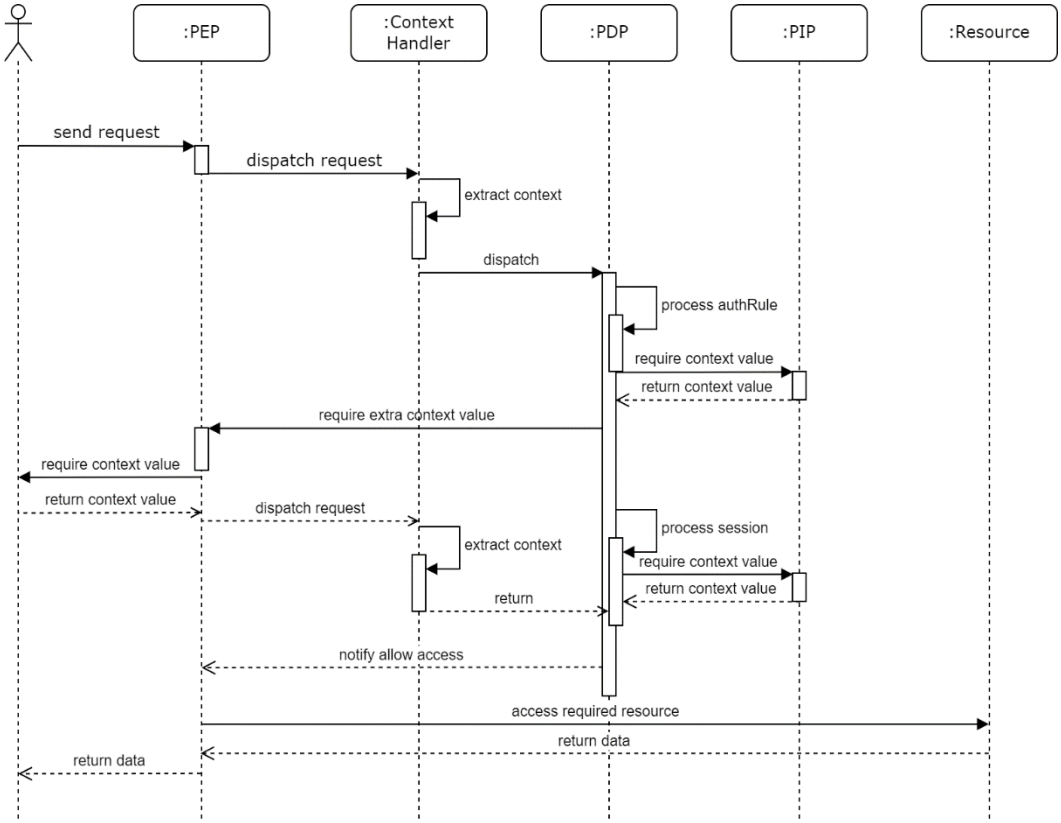

Figure 14 Request Processing Workflow

5.3 Example: Lab Assets Access

This section gives a specific example about how to write Parole script to control access to lab assets. The whole code is attached in Appendix B. Firstly, the programmer needs to identify the distribution of the assets within the lab. Figure 15 shows a sample assets distribution within the lab. From top to bottom, the first layer is the logical groups within the lab, which hold a bunch of physical entities. `enclave_01` includes the computing properties of the lab such as GPUs, robots, and databases. The SRI part includes the member information within the software research institute (SRI).

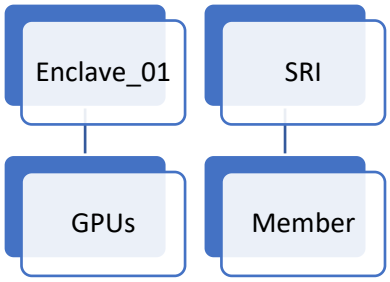

Figure 15 Assets Distribution for Lab Assets Access Example

Using Parole to describe such distribution is straight-forward - using the namespace. The structure showed in above figure are described as below. As you can see in the code snippets, the namespaces are nested which intuitively mirrors the relationships among different assets just like that in the figure.

```
namespace enclave_01{
    namespace gpu{…}
}
namespace db {
    namespace dataset{…}
}
namespace sri {
    namespace member{…}
}
```

The code snippet below shows the inside of the namespace. Basically, it starts with some import statements, so that current namespace can use the attributes defined in other namespace in later `authRule` and session. Import statements are the same as java does. In this example, `gpu` in `enclave_01` imports attributes declared in namespace `sri` and `sri.member` for using later.

There are also attributes declarations. These declarations define some attributes that are only available within the current namespace. Another namespace which wants to use these namespace needs import the namespace holding these attributes before using them.

Then it comes to `authRule`, which evaluates if the user has the related identity. In the example, there are two `authRule` to evaluate if the user is a member in `sri` or a `gpu` administrator respectively. Specifically, `authRule` uses the extracted value from the request, which is explicitly indicated with `REQ`, to compare with the authentic values stored in the PIP. For higher-prioritised roles, the programmer can add more verifications. The default return value of an `authRule` is `deny` if the user's identity fails to be authenticated. Once `deny` is returned, the system terminates this request.

The final parts within the namespace are a set of sessions. There is a session in the example to control user to execute on resources. In the session, there are identity-centric rules to ensure the requester's context satisfies the requirements and the risk of the request does not go over the threshold. It needs to be noticed that besides risk calculations, the session also allows evaluating the context repeatedly during the

session. In this example, the session for `gpu_admin` also checks if the user's location is in `lab` per minute (6000 milliseconds).

```
namespace enclave_01{
    namespace gpu{
        import sri.*;
        import sri.member.*;
        string name, addr, manufacturer, status, value;
        authRule sri_member {
            REQ.name in sri.member;
        }
        authRule gpu_admin {
            REQ.name in find(sri.member, (role="gpu-admin"))
                 && REQ.device.name in sri.member.device;
        }
        session execute{            //calculate risks
            sri_member:
                RiskFB(sri_member.fcl, value, history) <= "medium";
            gpu_admin:
                RiskFB(gpu_admin.fcl, history) <= "low";
                REQ.location == "lab", freq = 6000;
        }
    }
}
namespace sri {
    namespace member{
        string name, role, password, expireDate;
        namespace device{
            string name, ip, manufacturer;
        }
    }
}
```

Chapter 6

PAROLE Language Implementation

This chapter describes: 1) how the context and risk aware requirements outlined in the last section of Chapter 4 will be realised in the PAROLE language -section 6.1 and 6.2.; 2) the design and implementation of the PAROLE interpreter – section 6.3.

6.1 Representing Context in PAROLE

Context can be represented in PAROLE by several language elements. In the first place, PAROLE supports an explicit `Attribute` element which allows the definition of attributes for use in context specification. Secondly, PAROLE has simple but expressive statement and expression syntactical grammar. This enables the easy formulation of contextual expressions/conditions/predicates as described in the discussion on context in Chapter 5. This includes the formulation of both simplex and complex conditions. Moreover, PAROLE provides a data aggregation capability to support context and situation definition should this be required. A situation is sometimes defined as the combination of purpose or mission and a set of related situation attributes. While this combination of attributes does not *require* direct aggregation (e.g., via a class or struct type construct) it is often convenient for policy authoring, management, and readability to have an explicit structure. In PAROLE, data aggregation is provided by the *namespace* construct which has been adapted from the ALFA language ("Tutorial," n.d.). It is valid to ask if this construct is sufficient to support the complexity or richness of the various models or if the construct is overloaded? E.g., is it valid to define an `entity` or `context` constructor, and leave namespace to define the higher-level domain? Because arguments can be made for both approaches, a better option is to go with just a single aggregator construct and retain `namespace`. Finally, PAROLE provides support for both primary and secondary

context definition – though in the latter case via information retrieval from secondary sources rather than via inference.

In summary:

1. Context is defined in PAROLE as logical assertions over one or more attributes of one or more namespace objects. Context is implicit i.e., there is no formal language support for it. This usage is sufficient to define both general (e.g. spatial/temporal approaches) as well as domain specific (e.g. medical (Ye Tian et al., 2017) or physical context as described by UbiCOSM (Corrad et al., 2004)) notions of context.
2. Situation is defined as a namespace construct to capture the operational need semantics. The situation object must contain an attribute defining the purpose. Again, this is implicit – no direct language support is given. PAROLE does not directly support the specification of `SituationalAssertions` such as "User B is in Building A". However, this is easily supported by defining a User namespace with Location attribute.
3. The evidence above shows that PAROLE provides sufficient support for the definition of context awareness in access control according to the various definitions from the reviewed literature without requiring any specific extensions to the language.

6.2 Representing Risk Awareness in PAROLE

Trust based access in ZTA can be granted in either a binary or scaled manner. In the case of binary (or criteria) based access control, decisions are typically based on the use of trust indicators including security metrics (e.g., authentication) or trust assertions (e.g., PKI certificates). A number of "*qualified attributes*" may be used to evaluate whether or not to grant access. PAROLE can easily implement a binary trust algorithm based on the use of languages constructs such as `AuthRule`, Attributes and actors as as well as formulation of contextual expressions/conditions/predicates as described in the discussion above.

Scaled or score based trust algorithms may also be calculated deterministically over a number of attributes, such as operating system patch level and enterprise asset's current state, from different sources with the possibility to specify different weightings

for each data source as detailed by Rose et al. (2019). Moreover, a scale of such scoring points may exist corresponding to different levels of trustworthiness and allowing different levels of access, (Cheng et al., 2007b), (Osborn et al., 2016b), (Rose et al., 2019) with the possible accompaniment of a trust enhancement or risk mitigation action for each level. In the same way as outlined for CAAC and criteria-based decision making, PAROLE provides a broad range of constructs to enable the specification of discrete scale-based risk awareness approaches for ZTA.

Scaled based trust approaches are the base of many of the risk aware approaches described in the literature reviewed in previous chapters. However, scale based approaches do not deal easily with the uncertainty inherent in the dynamic contexts of the complex current computing landscape. Many researchers have therefore adopted probabilistic *context reasoning* approaches (Perera et al., 2013) based on machine learning, fuzzy logic, ontology based etc. PAROLE adopts fuzzy logic as the mechanism to express uncertainty associated with risk. More specifically, it adopts the Fuzzy Control Language (FCL) ("Fuzzy Control Programming," 1997)-a language for implementing fuzzy logic, especially fuzzy control standardised by IEC 61131-7. PAROLE embeds a Java library implementation of FCL, *jFuzzyLogic*, (Cingolani and Alcalá-Fdez, 2013), as its fuzzy logic engine.

6.3 Language Implementation

Generally, this project designs and implement a domain-specific language (DSL) that is used to control the access to resources in ZTA. The tool that is used to build interpreter is ANTLR (ANother Tool for Language Recognition) (Parr, 2013), which is a powerful parser generator. A more specific introduction to ANTLR4 and how it is used to build the interpreter will be presented.

After the implementation of the Parole interpreter, it needs to set up a network environment to simulate the function of ZTA, where related contextual information of the request is evaluated in the policy engine. The function of the policy engine is achieved by the interpreter. Specific experimental network environment setup will then be described. Pre-mentioned use cases are used to evaluate the correctness of the access decision given by the interpreter. The evaluation result shows that the interpreter can give the right access decision correctly based on the evaluation of the context.

6.3.1 Domain-Specific Language

General-purpose language (GPL) that the programmers use to code are broadly applicable across different application domains, such as java and python. However, such languages lack the ability to handle specialised features of a particular domain, because of the complex lengthy code to achieve a particular function, or lack of expressiveness making people hard to read. So here comes to the domain-specific language (DSL). A DSL has a very clear intent that only focuses on solving the problems of one specific domain. For example, Structured Query Language (SQL) is an example of DSL that is used only to manage and operate on relational database. SQL's statements are easy to understand which looks like the short sentences that people would understand. SQL largely lowers the barriers to manage the database because the functions have been well encapsulated within the language constructions. Users only need one SQL statement to read data from database, compared with java which may require dozens line of code.

DSL syntax is concise, specially designed for one domain. There are generally two kinds of DSL based on how it is implemented. One is internal DSL, which can be seen as the subset of the hosting language. When the user is using the internal DSL, it makes them feel like using the hosting language because the syntax and programming model is similar. The other kind of DSL is external DSL, which has its own customised syntax. Customised syntax means the hosting language's compiler or interpreter cannot directly handle the scripts written in DSL. So, building an external DSL is like creating a compiler or interpreter. The main function of compiler and interpreter is to convert the code from one format to another. More specifically, compiler converts human-readable/high-level language code to machine-readable/low-level language code and then stores the converted code in memory. While an interpreter also translates the code like compiler but then it then directly executes on the code.

Parole policy language is an external DSL, and it requires giving on-the-fly access decisions. So, what it needs is an interpreter that reads and interprets Parole-format script. And then the server is equipped with the ability to resolve the request and give the corresponding access decision based on the values in the request.

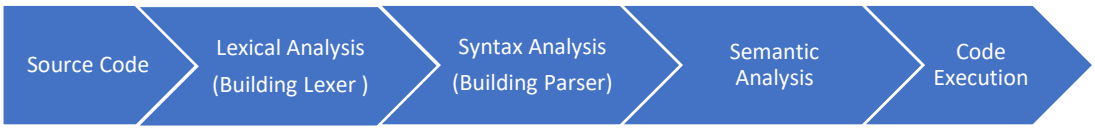

Figure 16 Process to Build Interpreter

As it is indicated in Figure 16, there are generally four phases to build an interpreter. First is to design the language, which is specifically introduced in the previous chapter. With basic language constructions determined, then it is time to do lexical analysis. A lexer is used to convert a sentence into a sequence of lexical tokens, this process is called tokenization. Lexer sequentially scans every word in the code, and then pairs each word to its specific token type. The result of tokenization (Guo, 1997) is a sequence of strings which record the token type and token value pair, and the order keeps the same as the original code.

The next step is to do syntax analysis or parsing. A parser is used at this stage to check if the expressions and statements composed of tokens are syntactically correct. A parser takes in the token sequence generated by the lexer and then refactors the structure of the words from a sequence of string to tree structure, based on the production rules defined by the user. There are generally two types of techniques to build the parse tree: top-down parsing and bottom-up parsing. Top-down parsing is a parsing starts with the grammar's start symbol, and recursively predicts the structure by applying grammar rules, working its way towards matching the input tokens. Bottom-up parsing constructs the parse tree from the input tokens, gradually combining them into higher-level structures until the start symbol is reached, assembling the tree through a data-driven approach.

The generated parse tree can identify the syntactical errors in the code; however, it does not know about the semantics about language constructs. Semantics of programming language indicate how code’s syntax structure can ultimately derive the language’s operational or functional characteristics. So, the next step of building interpreter is to do semantic analysis, which is to determine the meaning of the text by evaluating its syntax against context using type checking, symbol resolution, and scope management to ensure logical coherence and adherence to formal language rules. After the syntax and semantics of the source code are both evaluated and no error

occurs, the source code is then to be executed. This phase is done by using the host language to implement the semantics of the code.

Figure 17 represents the whole process how an interpreter handles the language using a snippet of code. Firstly, the code is transformed into a sequence of tokens. As it is represented, every word (in blue) keeps the same order and is assigned token type (in red). The generated tokens are then processed by the parser to check syntax correctness and build the parse tree[12] accordingly. Finally, the parse tree is walked through, and the symbol table is generated to relate the lexeme with its location in the source code for further scope resolution and type checking.

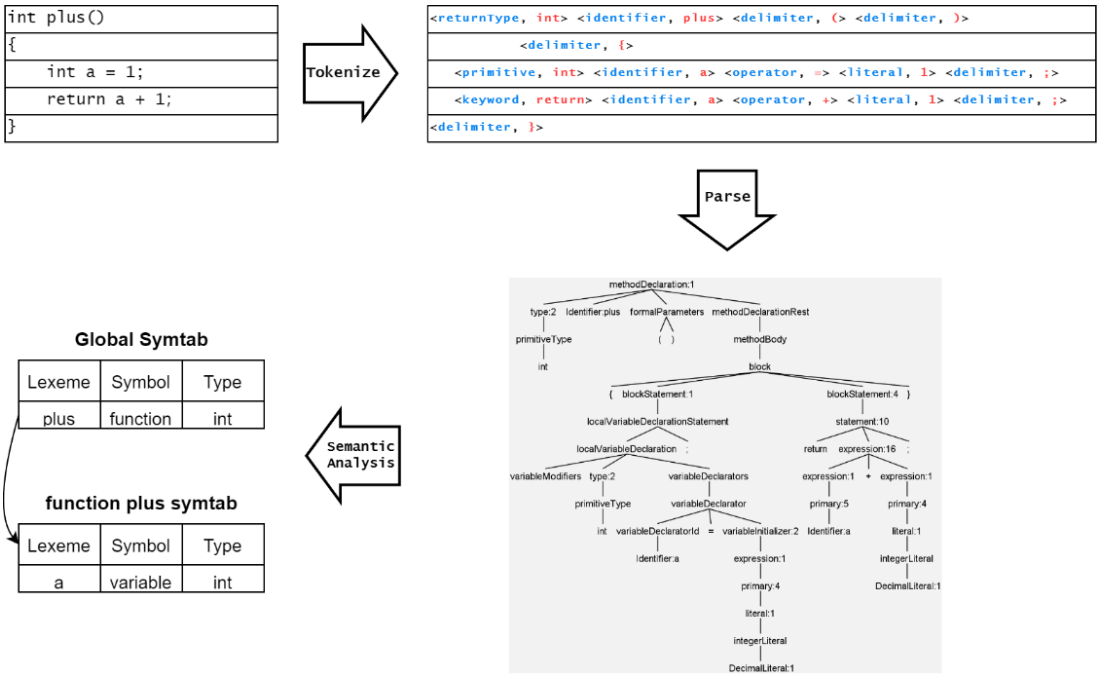

Figure 17 Source Code Handling Process

6.3.2 Tools: ANTLR4

The tool that is used to implement the interpreter is ANTLR[3] (Another Tool for Language Recognition). ANTLR is a powerful parser generator for building DSLs that allows you to define the grammar for a language in a formal and human-readable way,

[1] The parse tree is generated by ANTLR4 IntelliJ IDEA plugin: https://github.com/antlr/intellij-plugin-v4

[2] The grammar is http://media.pragprog.com/titles/tpantlr2/code/tour/Java.g4

[3] https://www.antlr.org/

using an Extended Backus-Naur Form (EBNF) notation. The grammar compiled by ANTLR needs to specify the syntax and structure of the language, including the rules for valid tokens (lexer rules) and the way they can be combined to form valid sentences or expressions (parser rules). Parser rules start with lowercase letters, and lexer rules with uppercase. Once the grammar is defined and successfully compiled, ANTLR4 automatically generates the lexer and the parser in the hosting language (e.g., Java, python, or JavaScript).

ANTLR supports four language patterns:

1) **Sequence** consists of a sequence of elements or subphrases.
2) **Choice** includes a set of phrase alternatives.
3) **Token dependency** indicates the presence of one token requires the presence of other tokens.
4) **Nested phrase** defines a language structure that can contain self-similar language constructs.

Here's a table summarizing ANTLR's core grammar notations:

Table 4 ANTLR4 Core Notation (Parr, 2013)

Syntax	Description
`x`	Match token, rule reference, or subrule `x`.
`x y ... z`	Match a sequence of rule elements.
`(... | ... | ...)`	Subrule with multiple alternatives.
`x ?`	Match `x` or skip it.
`x *`	Match `x` zero or more times.
`x +`	Match `x` one or more times.
`r : ... ;`	Define rule `r`.
`r : ... | ... | ... ;`	Define rule `r` with multiple alternatives.

6.3.3 Implementation

This section specifically introduces how the language is implemented using ANTLR4 and Java. The whole grammar is in Appendix A

. There are mainly three phases to implementing the language. It starts with writing the syntax of Parole using ANTLR grammar. The grammar of Parole includes parser rules and lexer rules. Then it talks about the symbol table that used to associate the semantic with the token in the grammar. Finally, it uses the ANTLR to build parse tree

listener to walk through the parse tree twice. The first pass defines the symbols and scopes and defined the symbol in the related scope. The second pass resolves the symbol which means it figures out which definition is of the current symbol by finding the symbol in the current scope, and then the definition is used in the hosting language for further manipulation.

Parser Rule

The grammar of Parole follows ANTLR4 grammar writing pattern, which starts with the header that names the grammar followed by a set of rules that can invoke other rules:

```
grammar <<grammar-name>>;
<<rule-name₁>>: <<rule-content₁>>;
<<rule-name₂>>: <<rule-content₂>>;
…
```

Designing the grammar of a programming language is a process of top-down functional decomposition. So, the starting rule can describe the overall input pattern of the language. For Parole, it can be described in English pseudocode that it is "a program that consists of a sequence of `namespaces`", because a Parole script is composed of a series of nested namespaces holding other language constructs. In ANTLR4 grammar, it is represented as:

```
paroleProgram: namespace*;
```

As it is described in Chapter 5, there are four kinds of language constructs: `namespace`, `attribute`, `authRule`, and `session`. These constructs are all included in the rule `paroleStruct`.

```
paroleStruct: attrDecl | authRule | session | namespace;
```

Table 5 Parole Language Constructs

Language Construct	Specification
`namespace`	`namespace: PE_NAMESPACE ID '{' (importStmt)* (paroleStruct)* '}';`
`attrDecl`	`attrDecl: attrType ID (',' ID)* ';' ;`
`authRule`	`authRule: PE_AUTHRULE ID '{' statement* '}';`
`session`	`session: PE_SESSION action '{'role_session* '}';`

Table 5 gives out the specification of each language construct:

- Rule `namespace`: namespace is identified by the keyword `namespace` (represented as `PE_NAMESPACE`). The identifier of namespace is stored in lexer

rule `ID`. Then the namespace body is enclosed by curly brace. Inside the namespace body, it first declares the imported namespace if it has any. Then there can be several definitions of Parole language construct.

- Rule `attrDecl`: attribute declaration works like conventional programming languages. It starts with attribute type and followed by one or several attribute names.
- Rule `authRule`: it is identified by the keyword `authRule` and its name. In its body, it contains neither or several statements.
- Rule `Session`: `session` is identified by the keyword `session` and followed by the action name. In the session body, it contains neither or several `role_session` rules.

Parole also supports a series of expressions, including invoking risk function block, `find` function, logical operations, assignments and so on. As it is showed in Figure 18, every alternative in the `expression` rule end with a label identified by `#` operator.

```
expression: findExpr #find |
            riskFB #risk |
            expression '.' expression #reference |
            expression 'in' expression #exist |
            expression op = ('<=' | '>=' | '>' | '<') expression #comparison |
            expression op=('!=' | '==') expression #equality |
            expression op=('&&' | '||') expression #logic |
            expression '=' expression #equal|
            expression ',' expression #list|
            LITERAL #literal | STRING #string | BOOLEAN #boolean | NULL #null | ID #id
            ;
```

Figure 18 Parole Expression

ANTLR will create a separate listener method for each alternative with label. Here is the description of each alternative:

- **Find:** This expression refers to `findExpr` rule which specifies the pattern to invoke `find` function, which aims to find documents that match a set of selection criteria. If so, the document will be returned and stored, otherwise, error message will be printed. The first parameter of `find` function is pip's name. If there is no expression follows, then it means find everything in the pip. If there are following expressions, the contain the selection criteria. The following expressions are generally instance of rule `equal` and rule `comparison`.

- **Risk:** This expression refers to `riskFB` rule, which specifies the pattern to invoke `riskFB` function. `riskFB` function has several parameters including the required `fcl` file, and the inputs specified in the `fcl` file.
- **Reference:** This expression handles four kinds of situations 1) `REQ.<field> :` this is the situation means it the value of indicated field; 2) reference in `FindExpr:` this situation means the specific attribute in corresponding namespace and makes sure it exists; 3) in `import` statement: this mean the name of the imported namespace; 4) `.fcl` file: this situation specifies the name of the `fcl` file.
- **Exist:** this expression means it checks the value of left expression can be find in the right expression.
- **Comparison, Equality, Logic:** These rules use mathematical operators to figure out the relationship between the left expression and the right expression.
- **Equal:** This rule is used in the rule `findExpr` as the selection criteria, not the assignment of some attribute.
- **List:** rule `list` represents a list of expressions. This expression often occurs in the rule `findExpr` and rule `riskFB`.
- **Literal, String, Boolean, Null:** These alternatives all represent the values they store.
- **ID:** This rule is the identifier of the parole language constructs.

Lexer Rule

The lexer rules are used in following ways:

- Lexer rules reserve the keywords. For example: `PE_NAMESPACE` represents `namespace`, `T_STRING` represents `string`.
- Lexer rules specify the patterns of identifiers in Parole. The allowed identifier in Parole starts with a letter and follows by arbitrary length of letter or digit or underscore (_). `ID : LETTER ( LETTER | Digit | '_' )*;`
- Lexer rules help skip the compilation of whitespace and comments including single-line and multi-line comments.

6.3.4 Symbol Table

After the grammar is successfully compiled by ANTLR, and the lexer and the parser are generated, then it is time to build a symbol table to record and track the semantics of tokens. Figure 19 shows the overall structure of the Parole symbol table. Basically, the symbol table is built based on the meta-attributes of a symbol – type, symbol, and scope.

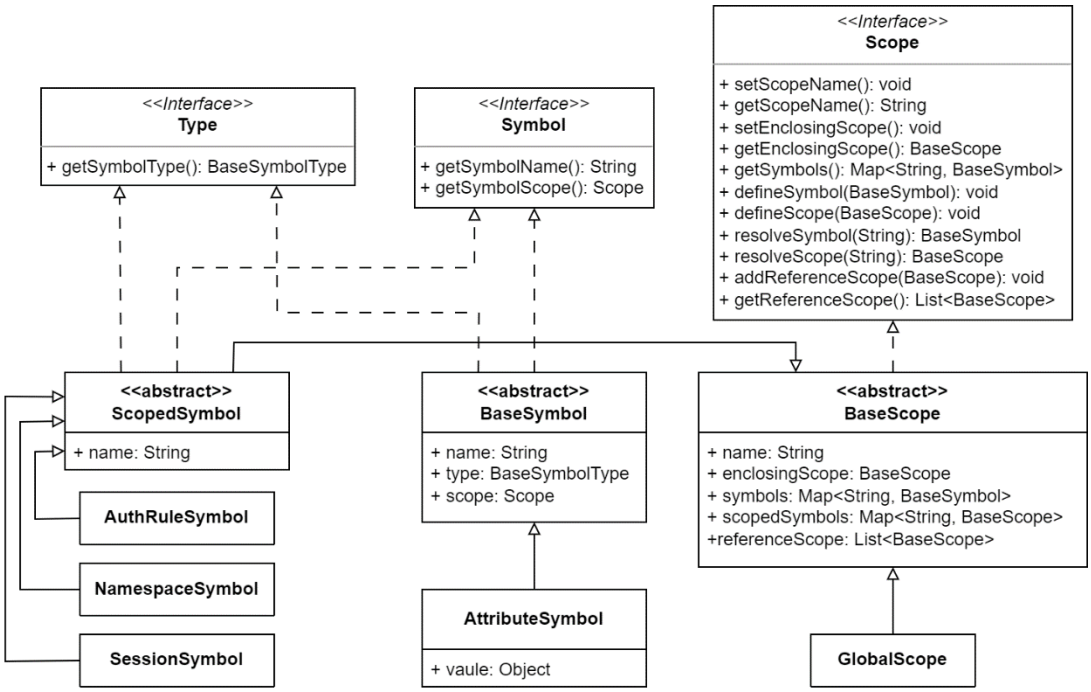

Figure 19 Parole Symbol Table

Interface Symbol declares the methods to get symbol's name and scope. Interface type declares the method to get the type of a symbol. There are four types of symbols in Parole, and Table 6 lists the supported type for each kind of symbol. As for interface Scope, it declares a series of methods that are used to enable linking and switching among nested scopes, define and resolve symbols within the scopes, and add reference scopes if there is import statement within.

Table 6 Symbol Types

Symbol	**Type**
attribute	`tINT, tREAL, tSTRING, tBOOLEAN`
namespace	`peNAMESPACE`
authRule	`peAUTHRULE`
session	`peSESSION`

On starting the interpretation, it first appends a global scope to scope stack. In the global scope, other constructs are defined. As for the language constructs that include a functioning body enclosed by a pair of curly braces, it actually creates a scope that is identified by its name. While at the same time, they are also a symbol defined within the current scope. These kinds of symbols are grouped into scoped symbols which not only implement both the methods of type and symbol, but also extend the methods of base scope. As for attribute symbols, they are the instantiated base symbols that can be defined and resolved in the scoped symbols and associate themselves with the values that comply with their symbol type.

6.3.5 Interpretation

After building the symbol table, it is time to start walking through the parse tree. Parole uses the ANTLR built-in parse tree walker mechanism called listener, which generates enter and exit methods for each rule. As the walker encounters some rule, say rule `namespace`, it triggers `enterNamespace()` method with the `NamespaceContext` parse tree node. After the walker has finished visiting every child of the `namespace` node, it then triggers `exitNamespace()` method with the `NamespaceContext` parse tree node.

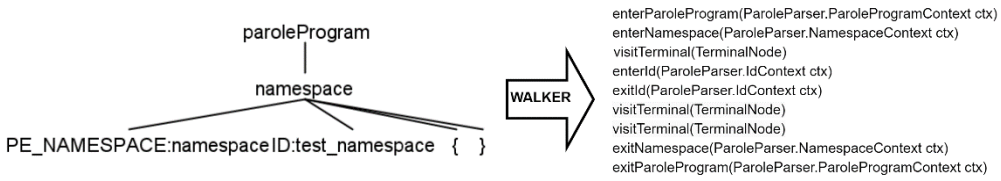

Figure 20 Listener Call Sequence

After all the listeners are created, then it needs to write specific actions in the required listeners. Generally, if there is no import statement, then after a walk through the parse tree marks the end of the program, because the symbols are all defined before they are resolved. To support using a symbol before it is defined in the source file, or say forward reference, it needs to make two walks through the parse tree. In the first walk, it defines the scopes and ensures the relationships between the scopes, in the meantime, it defines the symbols within the scopes and assigns value to the symbol if the symbol is initialised. At the time of the second walk, it has known the scope relationships and symbol definitions, so it can start resolve symbols.

Definition Phase

Definition phase starts with the `currentScope` point to the `globalScope` with no symbol defined. While as the walker enter and exit each node, scopes and symbols will be defined. Specifically, the processing logic during the definition phase is described in Algorithm 1. When entering a node, it first checks if the current scope has had other symbols using the name of current scope, if so, it prints error message and exit the program. If there are no duplicate names, then current node is defined in the current scope using `defineSymbol(BaseSymbol)` method. At the same time, if the current node is an instance of scoped symbol, then the scope identified by the current node is defined in the current scope using `defineScope(BaseScope)` method and the current scope is pointed to the scope identified by the current node. When exiting the node, if the current node is an instance of the scoped symbol, current scope then be pointed to its enclosing scope.

Handling the nodes that are literals, strings, bool (`true` and `false`) and null (`null`) are different from the symbols, because after exiting these nodes, the value of the node is known. At this time, their values are stored into the instance a data structure provided by ANTLR to associate a property with a parse tree node, called `ParseTreeProperty`. The values stored will then be directly used in the resolution phase.

ALGORITHM 1: DEFINITION PHASE PROCESSING LOGIC

```
    Input: ctx: ParoleParser.RuleContext, BaseScope currentScope,
    PareTreePropery<Basescope> scopes, ParseTreeProperty<Object> values
1   If ctx instance of AttrDeclContext; then name ← ctx's identifier
2     If currentScope resolve symbol name is null;
      then get symbol type
      currentScope define symbol with name and type
3   If ctx instance of NamespaceContext or AuthRuleContext or
    SessionContext;
     then name <- ctx's identifier
4     If currentScope resolve scope name is null;
      then get symbol type;
      currentScope define scoped symbol with name and type;
      save symbol into scopes
      currentScope changed to scope of Scoped Symbol
5   If ctx instance of LiteralContext or StringContext or BooleanContext
    or NullContext; then save ctx's value into values
```

Resolution Phase

Resolution phase is the core of Parole, during which how each language construct handles context is implemented. When entering the `ParoleProgram` node, it first gets

the target resource name in the request, and check if the resource is in the related PIP, if so, the program continues, otherwise an error is printed, and program stops.

Basically, it does not need to do extra things only change current the scope pointer when entering and exiting the scoped symbols like namespace. However, as the listener always walks through every node of the parse tree, and the processing logic is defined per rule, so that every functioning block - `authRule` and session is always functioning. While the only namespace that should take effect is the one that hosts the target resource, and the only `authRule` that should be evaluated is the one in the target namespace whose identifier is the same as the value of the field `role` in the request, and the only session that should be activated is based on the values of the filed `action` and the value of the filed `role` in the request. This requirement needs extra control in three places: `paroleStruct` node, `authRule` node, and `session` node.

ALGORITHM 2: REMOVE REDUNDANT LANGUAGE CONSTRUCTS

```
Input: ctx: ParoleParser.RuleContext, BaseScope currentScope,
JSONObject request
If ctx instance of ParoleStructContext; then targetScope <- full
path to target in request
    If full path to currentScope != targetScope AND (ctx's child is
    instance of AuthRuleContext or SessionContext);
    then remove child
If ctx instance of AuthRuleContext; then role <- value of field role
in request
    If identifier of ctx != role; then remove the children of ctx
If ctx instance of SessionContext; then action <- value of field
action in request
    If identifier of ctx != action; then remove the children of ctx
```

When entering `paroleStruct` node, it checks if the current namespace is the one holding the targe resource, if not, it removes the `authRule` node or the `session` node it holds. This action is done at the `paroleStruct` node instead of at the `namespace` node is because `attrDecl` nodes defined within the current namespace need to be reserved in case the current namespace is imported by the target namespace, while `authRule` node and `session` node will be removed to avoid them from taking effect and impacting the evaluation result. While in the target namespace, there can exist `authRule` nodes that are not for the role claimed in the request and `session` nodes that are not for the action claimed in the request. Therefore, on entering the `authRule` node and `session` node, it needs to check first if the current `authRule` node or `session` node

is the one for the request. If not, the children of current node need to be removed completely. The whole handling procedure is showed in Algorithm 2.

As it is mentioned above, Parole support import statement, which mean the attributes declared in other namespaces will be available in the current namespace. This function is achieved through adding the scope specified in the import statement to current scope's reference scope list. To use the attribute defined in the reference scope, the attribute needs to explicitly specify the namespace it belongs to using the member access (dot) operator (`.`). For example, if the current namespace is `enclave_01`, and `enclave_01` has imported another namespace call `sri`. In namespace `sri`, there is an attribute `role`, then using this attribute in `enclave_01` is like `sri.role`.

A critical feature of Parole is to calculate risk per request using fuzzy logic. This function is achieved through rule `riskFB`. The use of `riskFB` function requires pre-defined file written in fuzzy control language (`fcl`), and the input parameters defined in the `.fcl` file. These parameters are then evaluated using the API provided by jFuzzyLogic package (Cingolani and Alcalá-Fdez, 2013).

Chapter 7

Experimental Evaluation

7.1 Setup Experimental Environment

The experiment environment is demonstrated in Figure 21.This environment is used to test the use case described in 5.3 Example: Lab Assets Access. There are three separate networks identified using different colors. As it is indicated in the figure, the access request is directly sent to PEP using 10.0.0.1 network. Resources are isolated in 20.0.0.1 network and can be accessed through PEP once it has got the "allow" signal from PDP.

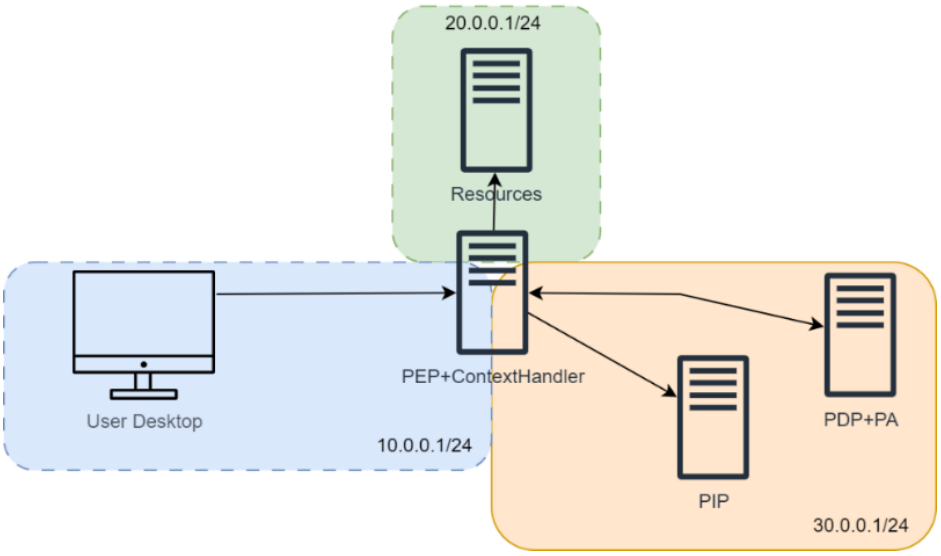

Figure 21 Experimental Environment

In PIP, it uses MongoDB[4] to store the related contextual attributes. These attributes will then be retrieved by PDP and used in policy execution. Table 7 lists the contextual attributes used in the experiment. Attributes generally come from two sources: from the request, and from PIP.

[4] https://www.mongodb.com/docs/

Table 7 Contextual Attributes For Experiment

Context Source	**Context**	**Value**
Request/User Provided	Role	sri_member
		gpu_admin
	Name	Shiyu Xiao
	Device.Name	DESKTOP - IJEQPSD
	Target	enclave_01.gpu.t4
	Action	execute
PIP.enclave_01	name	V100
		A100
		T4
	address	192.168.200.4
		192.168.200.5
		192.168.200.6
	manufacturer	NVIDIA
		NVIDIA
		NVIDIA
	status	active
		active
		active
	value	100000
		160000

In PAROLE, the attributes from the request will be identified by prefix `REQ.`. As for the context stored in the PIP, they are will first be stored in the database which is PIP in this experiment. Then, it needs two MongoDB collections: `enclave_01` and `sri`. Collection `enclave_01` is used to store the attributes of GPU resources, and collection `sri` is used to store the information about the users in organisation `sri`. Then these context are declared in the corresponding namespaces, see following code snippet. The absolute path to the namespace holding the attribute is the same with the path to finding it in the database. It needs to be noticed these attributes should not be initialised with default value, as the values of these attributes are retrieved from PIP and cannot changed using PAROLE script.

```
namespace enclave_01{
    namespace gpu{
        string name, addr, manufacturer, status;
        real value;
        …
    }
}
namespace sri {
    namespace member{
        string name, role, password, expireDate;
        namespace device{
            string name, ip, manufacturer;
        }
}
```

		60000
PIP.sri.member	name	Shiyu Xiao
		Yuhang Ye
		Xiangyu Liu
	role	Research Assistant, gpu_admin
		Lecturer
		Student
	device.name	DESKTOP - IJEQPSD
		yuhang - aw - x14
		xyliu - xps - sri
	device.ip	192.168.201.3
		192.168.201.4
		192.168.201.5
	device.manufacturer	Dell Inc.
		Intel Corporate
		Rivet Networks
	expireDate	Never
		Never
		Never

As for PAROLE itself, it is deployed in PDP+PA server. This server has OpenJDK 11.0.18 installed. The parsing of PAROLE language is using "antlr4-4.9.2.jar[5]" package. The connection between PDP and PIP (MongoDB) is supported by "mongo-java-driver-3.12.13.jar[6]" package. And the risk calculation in PAROLE is through the api provided by "jFuzzyLogic-1.0.jar[7]" package to evaluated pre-defined `fcl` files.

7.2 Experimental Process

Firstly, `User Desktop` the host sends http request to the PEP. The https request sent by the user contains some contextual information. As it is showed in Figure 22, there are two kinds of requests with different value of the field `role`, one being `sri_member`, the other being `gpu_admin`. These two requests aim to test the functionality of the main language constructs of PAROLE to see if they work fine with different context. There is an http server running in PEP to receive the request and extract context data within the request. Then the context data is forwarded the request to PDP in the JSON format.

[5] https://repo1.maven.org/maven2/org/antlr/antlr4/4.9.2/

[6] https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver/3.12.13

[7] https://sourceforge.net/projects/jfuzzylogic/files/jfuzzylogic/jFuzzyLogic.jar/download

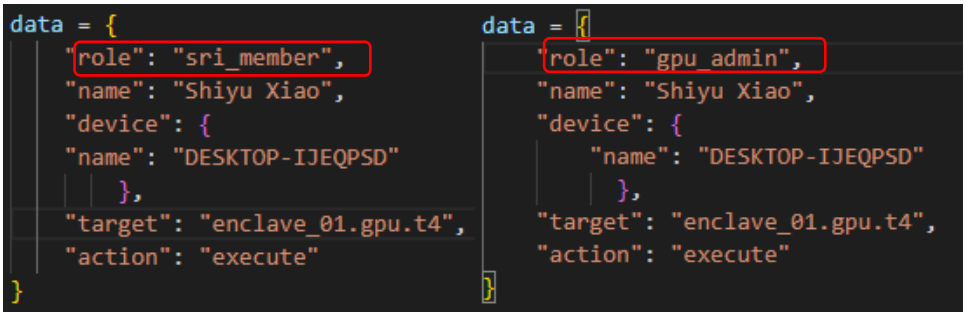

Figure 22 Context Data in Request

Then PAROLE retrieves the request and starts evaluation. It first ensures the `target` in the request does exist in the corresponding namespace. In the experiment, the use aims to access the resource `t4` in `enclave_01.gpu`. Then it starts to use `authRule` to check the role of the user. As it is indicated in the following code snippet, the first `authRule` checks if the user's role is `sri_member` by checking the value of field `name` in the request is in the collection `sri_member`, and the second one checks if the user's role is `gpu_admin` by checking the value of field `name` in the request is in the collection `sri_member` and has the role of `gpu-admin`. As the role `gpu_admin` has more priority than the `sri_member`, its `authRule` is more complex and thus defines more constraints. Successful evaluation of `authRule` will return the message `Authentication passed!`, otherwise the program will terminate and output error message.

```
authRule sri_member {
    REQ.name in sri.member;
}
authRule gpu_admin {
    REQ.name in find(sri.member, (role="gpu-admin"))
        && REQ.device.name in sri.member.device;
}
```

After a successful `authRule` authentication, it then starts `session` authentication. It first checks the value of `action` field in the request, and then uses the related rules based on the role. As it is indicated in the code snippet below, the user of role `sri_member` only needs to ensure the risk of his/her request not higher than `medium` level, while a `gpu_admin` needs lower risk score to gain access permission. Risk calculation of a request is invoked by using `RiskFB` which takes in pre-defined `fcl` file and followed with the parameters required in the `fcl` file. In the experiment, to calculate the risk of the request launched by a `sri_member` requires the value of the resource and this user's historical score, see Appendix C

while the risk of request launched by `gpu_admin` only needs to evaluate the historical score, see Appendix D. Then based on the risk level, it can decide if the request is allowed or not.

```
session execute{
            //calculate risks
            sri_member:
                RiskFB(sri_member.fcl, value, history) <= "medium";
            gpu_admin:
                RiskFB(gpu_admin.fcl, history) <= "low";
                //REQ.location == "sri", freq = 6000;
        }
```

The success of passing the `session` evaluation will return an `allow`, which remarks the user is allowed to access the required resource.

7.3 Experimental Result

Here are the experimental results to the request with the value of field `role` is `sri_member` and `gpu_admin`. Figure 23 and Figure 25 display the failed situation and output where the error occurs. Both situations fail because of a low history score making the risk score over the threshold defined in the script. The calculation of a requester's historical score, while an important factor in some systems, is considered a separate task typically associated with risk assessment. The rationale behind excluding the calculation of the history score within the scope of this project is based on the assumption that such information, i.e., the historical score, would be readily available from another component within the system. Therefore, in the program it is substituted with a random integer between [1,10]. It needs to be noticed they both have a `Authentication passed!` output, which means the evaluation on `authRule` is passed. Figure 24 and Figure 26 are the successful situations. Both situations have passed the `authRule` evaluation and `session` evaluation, so there is an `allow` returned in the end.

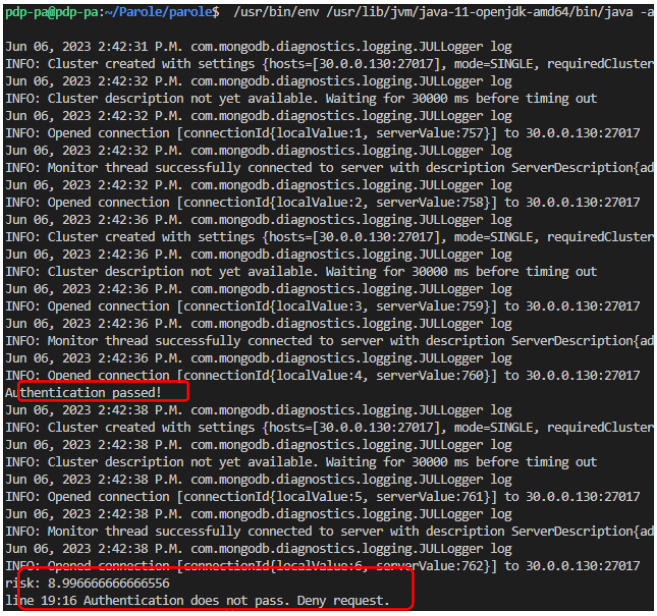

Figure 23 User Fail to Access Resource using role `sri_member`

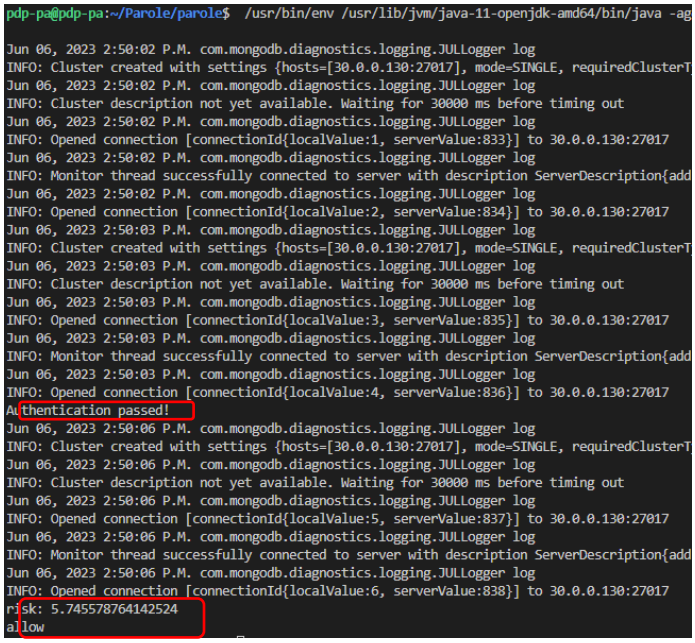

Figure 24 User Succeed to Access Resource using role `sri_member`

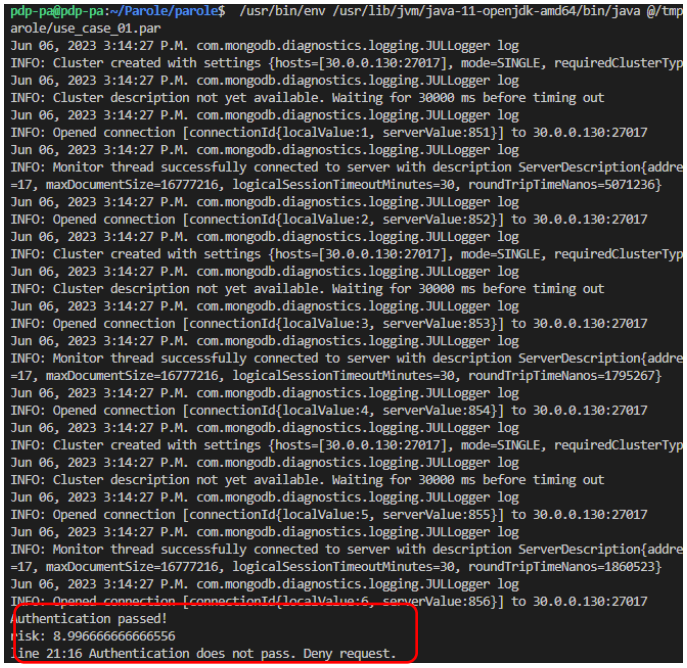

Figure 25 User Fail to Access Resource using role `gpu_admin`

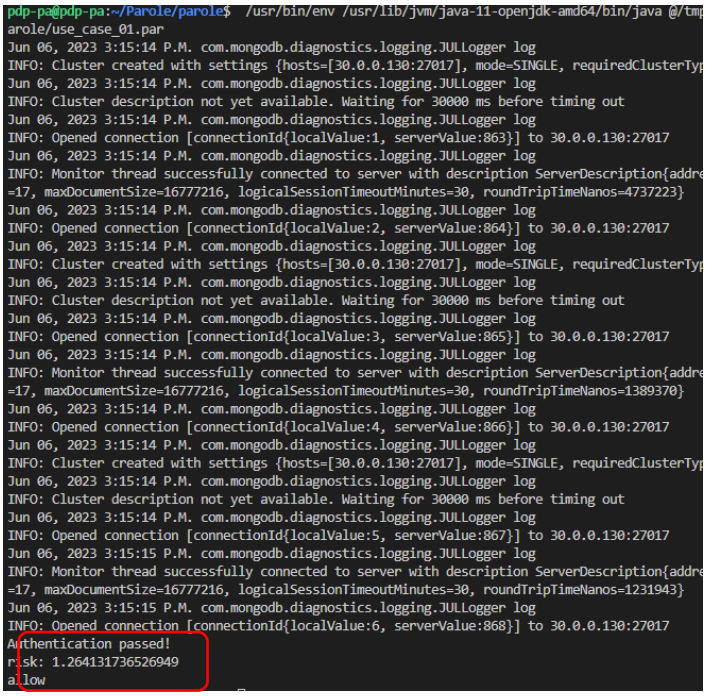

Figure 26 User Succeed to Access Resource using role `gpu_admin`

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This thesis has reviewed previous research in CAAC and risk and trust in access control with a goal to identify common concepts and themes in these fields and to examine their potential use in ZT security models. It is found that there are indeed many underlying commonalities across the various research works and that many of these ideas can be, and in some cases, and are being applied to ZT models and deployments.

Based on the findings, this thesis presents the specification of a policy language called PAROLE that is specially designed to handle the context in the zero-trust based system. PAROLE's interpreter is implemented using ANTLR parser generator and Java programming language. It is also verified in the thesis that PAROLE can give correct access decision based on the context of the requester.

8.2 Future Work

The future work of this project includes:

- Using the framework like Xtext[8] to create an IDE for PAROLE. The IDE will include the features e.g., code completion and code insight, so that the developers program faster and easier.
- The experiment in the project tests the functions of PAROLE to handle context in simple network environment. In the future, it needs to be further tested and validated in a more complex network setting, involving various IoT devices, fuzzy logic, and a more comprehensive zero-trust environment.

[8] https://www.eclipse.org/Xtext/

Glossary

ABAC Attribute Based Access Control

AC Access Control

AH Accepting Host

ANTLR ANother Tool for Language Recognition

APT Advanced Persistent Threats

BLP Bell-LaPadula

BYOD Bring Your Own Device

CAAC Context-Aware Access Control

CDM Continuous Diagnostics and Mitigation

CIA Confidentiality-Integrity-Availability

DAC Discretionary Access Control

DMZ Demilitarised Zones

DRM Digital Rights Management

DSL Domain-Specific Language

EBNF Extended Backus-Naur Form

GLP General-purpose Language

IH Initiating Host

IoT Internet of Things

MAC Mandatory Access Control

MTLS Mutual Transport Layer Security

NGF Next-Generation Firewall

NGF Next-Generation Firewall

NV Network Virtualisation

OT Operational Technology

PA Policy Administrator

PDP Policy Decision Point

PE Policy Engine

PEP Policy Enforcement Point

PL Permission Label

PoLP Principle of Least Privilege

QoS Quality of Service

RAdAC Risk-Adaptive Access Control

RBAC Role-Based Access Control

SDN Software Defined Network

SDP Software Defined Perimeter

SG Segmentation Gateway

SoD Separation of Duty

SoK Systematisation of Knowledge

SQL Structured Query Language

TI Trust Indicator

TLEE Trust Level Evaluation Engine

UCON Usage Control

VN Virtual Network

VO Virtual Organisation

ZT Zero Trust

ZTN Zero Trust Networking

ZTA Zero Trust Architecture

Reference

Abbasi, A.A., Shamshirband, S., Al-qaness, M.A., Abbasi, A., AL-Jallad, N.T., Mosavi, A., 2020. Resource-aware network topology management framework. ArXiv Prepr. ArXiv200300860.

Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P., 1999. Towards a Better Understanding of Context and Context-Awareness, in: Gellersen, H.-W. (Ed.), Handheld and Ubiquitous Computing, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 304–307. https://doi.org/10.1007/3-540-48157-5_29

Abowd, G.D., Mynatt, E.D., 2000. Charting past, present, and future research in ubiquitous computing. ACM Trans. Comput.-Hum. Interact. 7, 29–58. https://doi.org/10.1145/344949.344988

Ahmed, M., Mahmood, A.N., Hu, J., 2016. A survey of network anomaly detection techniques. J. Netw. Comput. Appl. 60, 19–31.

Aich, S., Sural, S., Majumdar, A.K., 2007. STARBAC: Spatiotemporal Role Based Access Control, in: Meersman, R., Tari, Z. (Eds.), On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 1567–1582. https://doi.org/10.1007/978-3-540-76843-2_32

Almutairi, A., Siewe, F., 2011. CA-UCON: a context-aware usage control model, in: Proceedings of the 5th ACM International Workshop on Context-Awareness for Self-Managing Systems. pp. 38–43.

Armando, A., Bezzi, M., Di Cerbo, F., Metoui, N., 2015. Balancing trust and risk in access control, in: On the Move to Meaningful Internet Systems: OTM 2015 Conferences: Confederated International Conferences: CoopIS, ODBASE, and C&TC 2015, Rhodes, Greece, October 26-30, 2015. Proceedings. Springer, pp. 660–676.

Bai, G., Gu, L., Feng, T., Guo, Y., Chen, X., 2010. Context-Aware Usage Control for Android, in: SecureComm. https://doi.org/10.1007/978-3-642-16161-2_19

Banks, A.S., Kisiel, M., Korsholm, P., 2021. Remote attestation: a literature review. ArXiv Prepr. ArXiv210502466.

Bernal Bernabe, J., Hernandez Ramos, J.L., Skarmeta Gomez, A.F., 2016. TACIoT: multidimensional trust-aware access control system for the Internet of Things. Soft Comput. 20, 1763–1779.

Bertino, E., Bonatti, P.A., Ferrari, E., 2001. TRBAC: A temporal role-based access control model. ACM Trans. Inf. Syst. Secur. 4, 191–233. https://doi.org/10.1145/501978.501979

Bertino, E., Catania, B., Damiani, M.L., Perlasca, P., 2005. GEO-RBAC: a spatially aware RBAC, in: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies - SACMAT '05. Presented at the the tenth ACM symposium, ACM Press, Stockholm, Sweden, p. 29. https://doi.org/10.1145/1063979.1063985

Brent, E., Leedy, P.D., 2016. Practical Research: Planning and Design. Teach. Sociol. 18, 248. https://doi.org/10.2307/1318509

Buck, C., Olenberger, C., Schweizer, A., Völter, F., Eymann, T., 2021. Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust. Comput. Secur. 110, 102436.

Bunningen, A., Feng, L., Apers, P., 2005. Context for Ubiquitous Data Management. pp. 17–24. https://doi.org/10.1109/UDM.2005.7

Ceccarelli, A., Bondavalli, A., Brancati, F., La Mattina, E., 2012. Improving security of internet services through continuous and transparent user identity verification, in: 2012 IEEE 31st Symposium on Reliable Distributed Systems. IEEE, pp. 201–206.

Cheng, P.-C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Reninger, A.S., 2007a. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control, in: 2007 IEEE Symposium on Security and Privacy (SP'07). IEEE, pp. 222–230.

Cheng, P.-C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Reninger, A.S., 2007b. Fuzzy Multi-Level Security: An Experiment on Quantified Risk-

Adaptive Access Control, in: 2007 IEEE Symposium on Security and Privacy (SP ’07). pp. 222–230. https://doi.org/10.1109/SP.2007.21

Chowdhury, N.M.M.K., Boutaba, R., 2010. A survey of network virtualization. Comput. Netw. 54, 862–876. https://doi.org/10.1016/j.comnet.2009.10.017

Chukkapalli, S.S.L., Piplai, A., Mittal, S., Gupta, M., Joshi, A., 2020. A Smart-Farming Ontology for Attribute Based Access Control, in: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). Presented at the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), pp. 29–34. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00017

Cingolani, P., Alcalá-Fdez, J., 2013. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming: Int. J. Comput. Intell. Syst. 6, 61. https://doi.org/10.1080/18756891.2013.818190

Claudio Agostino Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, Pierangela Samarati, 2008. A privacy-aware access control system [WWW Document]. ResearchGate. https://doi.org/10.3233/JCS-2008-0328

Computer Security Division, I.T.L., 2016. SP 800-162, Guide to ABAC Definition and Considerations | CSRC [WWW Document]. CSRC NIST. URL https://content.csrc.e1a.nist.gov/News/2014/SP-800-162,-Guide-to-ABAC-Definition-and-Considera (accessed 2.8.21).

Corrad, A., Montanari, R., Tibaldi, D., 2004. Context-based access control management in ubiquitous environments, in: Third IEEE International Symposium on Network Computing and Applications, 2004.(NCA 2004). Proceedings. IEEE, pp. 253–260.

Covington, M.J., Sastry, M.R., 2006. A Contextual Attribute-Based Access Control Model, in: Meersman, R., Tari, Z., Herrero, P. (Eds.), On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, Lecture Notes in

Computer Science. Springer, Berlin, Heidelberg, pp. 1996–2006. https://doi.org/10.1007/11915072_108

Creswell, J., Guetterman, T., 2018. Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research, 6th Edition.

Creswell, J.W., 2009. Research design: qualitative, quantitative, and mixed methods approaches, 3rd ed. ed. Sage Publications, Thousand Oaks, Calif.

Cunningham, C., 2018. The Zero Trust eXtended (ZTX) Ecosystem 15.

Cunningham, C., Holmes, D., Pollard, J., 2019. The Eight Business And Security Benefits Of Zero Trust.

Damianou, N., Dulay, N., Lupu, E., Sloman, M., 2001. The Ponder Policy Specification Language, in: Sloman, M., Lupu, E.C., Lobo, J. (Eds.), Policies for Distributed Systems and Networks, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 18–38. https://doi.org/10.1007/3-540-44569-2_2

Dimitrakos, T., Dilshener, T., Kravtsov, A., La Marra, A., Martinelli, F., Rizos, A., Rosetti, A., Saracino, A., 2020. 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). Presented at the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1801–1812. https://doi.org/10.1109/TrustCom50675.2020.00247

Dutta, S., Chukkapalli, S.S.L., Sulgekar, M., Krithivasan, S., Das, P.K., Joshi, A., 2020. Context Sensitive Access Control in Smart Home Environments, in: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). Presented at the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), IEEE, Baltimore, MD, USA, pp. 35–41. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00018

Elliott, A., Knight, S., 2010. Role Explosion: Acknowledging the Problem. Presented at the Software Engineering Research and Practice.

Equation Group: Questions And Answers, 2015. . Kaspersky Lab HQ 44.

Evolving Zero Trust, 2021. . Microsoft Corporation, p. 9.

eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01, 2017.

Ferraiolo, D., Kuhn, D., 1992. Role Based Access Control. 15th National Computer Security Conference: 554-563.

Ferraiolo, D.F., Barkley, J.F., Kuhn, D.R., 1999. A role-based access control model and reference implementation within a corporate intranet. ACM Trans. Inf. Syst. Secur. TISSEC 2, 34–64.

Ferraiolo, D.F., Kuhn, D.R., 2009. Role-Based Access Controls. ArXiv09032171 Cs.

Fuzzy Control Programming, 1997.

Gao, Z., Bird, C., Barr, E.T., 2017. To Type or Not to Type: Quantifying Detectable Bugs in JavaScript, in: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). Presented at the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), pp. 758–769. https://doi.org/10.1109/ICSE.2017.75

Getting Started with Zero Trust Access Management Trust Begins with Secure Identity | Okta [WWW Document], 2021. URL https://www.okta.com/resources/whitepaper/zero-trust-with-okta-modern-approach-to-secure-access/ (accessed 1.13.22).

Gilman, E., Barth, D., 2017. Zero Trust Networks [Book], 1st ed. O'Reilly Media.

Guo, J., 1997. Critical Tokenization and its Properties. Comput. Linguist. 23, 569–596.

Gupta, M., Benson, J., Patwa, F., Sandhu, R., 2019. Dynamic Groups and Attribute-Based Access Control for Next-Generation Smart Cars. https://doi.org/10.1145/3292006.3300048

Hansche, S., 2003. Official (ISC)®ø guide to the CISSP exam. Boca Raton, Fl : Auerbach Publications.

Harris, L., 2003. Home-based teleworking and the employment relationship: Managerial challenges and dilemmas. Pers. Rev. 32, 422–437. https://doi.org/10.1108/00483480310477515

Henricksen, K., 2003. A Framework for Context-Aware Pervasive Computing Applications. The University of Queensland.

Hilbert, F., Katranuschkov, P., Scherer, R.J., 2010. Dynamic context-aware information access in virtual organizations 6.

Hu, V., Kuhn, R., Yaga, D., 2017. Verification and Test Methods for Access Control Policies/Models (No. NIST Special Publication (SP) 800-192). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-192

Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., others, 2013. Guide to attribute based access control (abac) definition and considerations (draft). NIST Spec. Publ. 800, 1–54.

Jafarian, J.H., Amini, M., 2009. CAMAC: a context-aware mandatory access control model. ISC Int. J. Inf. Secur. 1, 35–54. https://doi.org/10.22042/isecure.2015.1.1.5

Jeff Williams, n.d. OWASP Risk Rating Methodology | OWASP Foundation [WWW Document]. URL https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (accessed 5.22.23).

Joint Task Force Transformation Initiative, 2011. Managing information security risk :: organization, mission, and information system view (No. NIST SP 800-39). National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-39

Jøsang, A., Presti, S.L., 2004. Analysing the relationship between risk and trust, in: Trust Management: Second International Conference, ITrust 2004, Oxford, UK, March 29-April 1, 2004. Proceedings 2. Springer, pp. 135–145.

Kagal, L., 2002. Rei : A Policy Language for the Me-Centric Project.

Kandala, S., Sandhu, R., Bhamidipati, V., 2011. An Attribute Based Framework for Risk-Adaptive Access Control Models, in: 2011 Sixth International

Conference on Availability, Reliability and Security. Presented at the 2011 Sixth International Conference on Availability, Reliability and Security, pp. 236–241. https://doi.org/10.1109/ARES.2011.41

Karthik, N., Dhulipala, V.R.S., 2011. Trust calculation in wireless sensor networks, in: 2011 3rd International Conference on Electronics Computer Technology. Presented at the 2011 3rd International Conference on Electronics Computer Technology, pp. 376–380. https://doi.org/10.1109/ICECTECH.2011.5941924

Kayes, A.S.M., Han, J., Rahayu, W., Islam, M.S., Colman, A., 2017. A Policy Model and Framework for Context-Aware Access Control to Information Resources. ArXiv170302162 Cs.

Kayes, A.S.M., Kalaria, R., Sarker, I.H., Islam, M., Watters, P.A., Ng, A., Hammoudeh, M., Badsha, S., Kumara, I., 2020. A survey of context-aware access control mechanisms for cloud and fog networks: Taxonomy and open research issues. Sensors 20, 2464.

Kim, H., Lee, E.A., 2017. Authentication and Authorization for the Internet of Things. IT Prof. 19, 27–33.

Kindervag, J., 2010. No More Chewy Centers: Introducing The Zero Trust Model Of Information Security 15.

Kindervag, J., Balaouras, S., Coit, L., 2010. Build Security Into Your Network's DNA: The Zero Trust Network Architecture 27.

Kulkarni, D., Tripathi, A., 2008. Context-aware role-based access control in pervasive computing systems, in: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08. Association for Computing Machinery, New York, NY, USA, pp. 113–122. https://doi.org/10.1145/1377836.1377854

Manchala, D.W., 2000. E-commerce trust metrics and models. IEEE Internet Comput. 4, 36–44. https://doi.org/10.1109/4236.832944

Mayfield, T., Roskos, J.E., Welke, S., Boone, J., McDonald, C., 1991. Integrity in Automated Information Systems. https://doi.org/10.21236/ada245555

Moubayed, A., Refaey, A., Shami, A., 2019. Software-Defined Perimeter (SDP): State of the Art Secure Solution for Modern Networks. IEEE Netw. 33, 226–233. https://doi.org/10.1109/MNET.2019.1800324

Ni, Q., Bertino, E., Lobo, J., 2010. Risk-based access control systems built on fuzzy inferences, in: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10. Association for Computing Machinery, Beijing, China, pp. 250–260. https://doi.org/10.1145/1755688.1755719

Okoli, C., 2015. A Guide to Conducting a Standalone Systematic Literature Review. Commun. Assoc. Inf. Syst. 37. https://doi.org/10.17705/1CAIS.03743

Osborn, B., McWilliams, J., Beyer, B., Saltonstall, M., 2016a. BeyondCorp: Design to deployment at Google.

Osborn, B., McWilliams, J., Beyer, B., Saltonstall, M., 2016b. BeyondCorp: Design to deployment at Google.

Ouechtati, H., Azzouna, N.B., 2017. Trust-abac towards an access control system for the internet of things, in: Green, Pervasive, and Cloud Computing: 12th International Conference, GPC 2017, Cetara, Italy, May 11-14, 2017, Proceedings 12. Springer, pp. 75–89.

Park, J., Sandhu, R., 2004. The UCON $_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur. TISSEC 7, 128–174. https://doi.org/10.1145/984334.984339

Parr, T., 2013. The Definitive ANTLR 4 Reference, 2nd ed. Pragmatic Bookshelf.

Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S., 2007. A Design Science Research Methodology for Information Systems Research. J. Manag. Inf. Syst. 24, 45–77. https://doi.org/10.2753/MIS0742-1222240302

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., 2014. Context Aware Computing for The Internet of Things: A Survey. IEEE Commun. Surv. Tutor. 16, 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., 2013. Context aware computing for the internet of things: A survey. IEEE Commun. Surv. Tutor. 16, 414–454.

Picard, N., Colin, J., Zampuniéris, D., 2018a. Context-aware and Attribute-based Access Control Applying Proactive Computing to IoT System, in: IoTBDS. https://doi.org/10.5220/0006815803330339

Picard, N., Colin, J.-N., Zampunieris, D., 2018b. Context-aware and Attribute-based Access Control Applying Proactive Computing to IoT System:, in: Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security. Presented at the Special Session on Recent Advances on Security, Privacy, Big Data and Internet of Things, SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, pp. 333–339. https://doi.org/10.5220/0006815803330339

Ray, I., Toahchoodee, M., 2007. A Spatio-temporal Role-Based Access Control Model. pp. 211–226. https://doi.org/10.1007/978-3-540-73538-0_16

Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E., 2000. Reputation systems. Commun. ACM 43, 45–48. https://doi.org/10.1145/355112.355122

Rose, S., Borchert, O., Mitchell, S., Connelly, S., 2020. Zero Trust Architecture (No. NIST Special Publication (SP) 800-207). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

Rose, S., Borchert, O., Mitchell, S., Connelly, S., 2019. Zero trust architecture. National Institute of Standards and Technology.

Ross, R.S., 2018. Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. NIST.

Sacco, O., Breslin, J.G., Decker, S., 2013. Fine-Grained Trust Assertions for Privacy Management in the Social Semantic Web, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Presented at the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 218–225. https://doi.org/10.1109/TrustCom.2013.30

Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E., 1996. Role-based access control models. Computer 29, 38–47. https://doi.org/10.1109/2.485845

Sarkar, S., Choudhary, G., Shandilya, S.K., Hussain, A., Kim, H., 2022. Security of zero trust networks in cloud computing: A comparative review. Sustainability 14, 11213.

Schefer-Wenzl, S., Strembeck, M., 2012. Modeling Context-Aware RBAC Models for Business Processes in Ubiquitous Computing Environments. Presented at the Proceedings - 2012 3rd FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing, MUSIC 2012. https://doi.org/10.1109/MUSIC.2012.29

Schilit, B., Adams, N., Want, R., 1994. Context-Aware Computing Applications, in: 1994 First Workshop on Mobile Computing Systems and Applications. Presented at the 1994 First Workshop on Mobile Computing Systems and Applications, pp. 85–90. https://doi.org/10.1109/WMCSA.1994.16

Schilit, B.N., Theimer, M.M., 1994. Disseminating active map information to mobile hosts. IEEE Netw. 8, 22–32. https://doi.org/10.1109/65.313011

Seamons, K.E., Winslett, M., Ting Yu, Smith, B., Child, E., Jacobson, J., Mills, H., Lina Yu, 2002. Requirements for policy languages for trust negotiation, in: Proceedings Third International Workshop on Policies for Distributed Systems and Networks. Presented at the Proceedings Third International Workshop on Policies for Distributed Systems and Networks, pp. 68–79. https://doi.org/10.1109/POLICY.2002.1011295

Seixas, N., Fonseca, J., Vieira, M., Madeira, H., 2009. Looking at Web Security Vulnerabilities from the Programming Language Perspective: A Field Study, in: 2009 20th International Symposium on Software Reliability Engineering. Presented at the 2009 20th International Symposium on Software Reliability Engineering, pp. 129–135. https://doi.org/10.1109/ISSRE.2009.30

Shameli-Sendi, A., Dagenais, M., Wang, L., 2018. Realtime intrusion risk assessment model based on attack and service dependency graphs. Comput. Commun. 116, 253–272.

Sheikh, N., Pawar, M., Lawrence, V., 2021. Zero trust using network micro segmentation, in: IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 1–6.

Software-Defined Perimeter (SDP) and Zero Trust [WWW Document], 2020. . CSA. URL https://cloudsecurityalliance.org/artifacts/software-defined-perimeter-and-zero-trust/ (accessed 5.31.23).

Software-Defined Perimeter (SDP) Specification v2.0 | CSA [WWW Document], 2022. URL https://cloudsecurityalliance.org/artifacts/software-defined-perimeter-zero-trust-specification-v2/ (accessed 5.31.23).

Su, L., 2010. User behaviour based access control decision, in: 2010 International Conference on E-Business and E-Government. IEEE, pp. 1372–1376.

Tutorial: A Beginner's Guide to XACML (Part 2) Getting Started with ALFA [WWW Document], n.d. . Axiomatics. URL https://www.axiomatics.com/resources/tutorial-a-beginers-guide-to-xacml-part-2-getting-started-with-alfa/ (accessed 11.19.21).

US Department of Defense, 1985. Department of Defense Trusted Computer System Evaluation Criteria, in: US Department of Defense (Ed.), The 'Orange Book' Series. Palgrave Macmillan UK, London, pp. 1–129. https://doi.org/10.1007/978-1-349-12020-8_1

Vanickis, R., Jacob, P., Dehghanzadeh, S., Lee, B., 2018. Access Control Policy Enforcement for Zero-Trust-Networking, in: 2018 29th Irish Signals and Systems Conference (ISSC). Presented at the 2018 29th Irish Signals and Systems Conference (ISSC), pp. 1–6. https://doi.org/10.1109/ISSC.2018.8585365

Vincentis, M.D., 2017. Micro-segmentation For Dummies®, 2nd VMware Special Edition, 2nd ed. John Wiley & Sons, Inc., 111 River St. Hoboken, NJ 07030-5774.

Ward, R., Beyer, B., 2014. Beyondcorp: A new approach to enterprise security.

Weiser, M., 1999. The computer for the 21st century. ACM SIGMOBILE Mob. Comput. Commun. Rev. 3, 3–11. https://doi.org/10.1145/329124.329126

Williams, C., 2007. Research Methods. J. Bus. Econ. Res. JBER 5. https://doi.org/10.19030/jber.v5i3.2532

Winkler, T., Haller, J., Gimpel, H., Weinhardt, C., 2007. Trust indicator modeling for a reputation service in virtual organizations.

Xiong, L., Liu, L., 2004. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. IEEE Trans. Knowl. Data Eng. 16, 843–857. https://doi.org/10.1109/TKDE.2004.1318566

Yaici, M., Ainennas, F., Zidi, N., 2019. Context-Aware Trust-Based Access Control for Ubiquitous Systems, in: CS & IT Conference Proceedings. CS & IT Conference Proceedings.

Yan, Z., Li, X., Wang, M., Vasilakos, A.V., 2015. Flexible data access control based on trust and reputation in cloud computing. IEEE Trans. Cloud Comput. 5, 485–498.

Yang, K., Liu, Z., Cao, Z., Jia, X., Wong, D.S., Ren, K., 2012. Taac: Temporal attribute-based access control for multi-authority cloud storage systems. Cryptol. EPrint Arch.

Yao, J., Venkitasubramaniam, P., Kishore, S., Snyder, L.V., Blum, R.S., 2017. Network topology risk assessment of stealthy cyber attacks on advanced metering infrastructure networks, in: 2017 51st Annual Conference on Information Sciences and Systems (CISS). IEEE, pp. 1–6.

Ye Tian, Yanbin Peng, Gaimei Gao, Xinguang Peng, 2017. Role-based Access Control for Body Area Networks Using Attribute-based Encryption in Cloud Storage. Int. J. Netw. Secur. 19. https://doi.org/10.6633/IJNS.201709.19(5).09

Zero Trust Maturity Model | CISA [WWW Document], 2023. URL https://www.cisa.gov/publication/zero-trust-maturity-model (accessed 1.13.22).

Appendix A

PRAOLE Policy Language Syntax

```
grammar Parole;

paroleProgram: namespace*;
namespace: PE_NAMESPACE ID '{' (importStmt | paroleStruct)* '}';

importStmt: IMPORT composed_id ('.*')? ';' ;
composed_id: ID (.ID)*;

paroleStruct: attrDecl | authRule | session | namespace;

attrDecl: attrType ID (',' ID)* ';' ;
attrType: T_REAL | T_INT | T_STRING | T_BOOLEAN;

authRule: PE_AUTHRULE ID '{' statement* '}';

session: PE_SESSION action '{'role_session* '}';
action: EXECUTE | READ | WRITE | DELETE;
role_session: ID ':' statement*;

statement: expression';';
expression: findExpr |
            riskFB |
            expression '.' expression |
            expression 'in' expression |
            expression op = ('<=' | '>=' | '>' | '<') expression |
            expression op=('!=' | '==') expression |
            expression op=('&&' | '||') expression |
            expression '=' expression |
            expression ',' expression |
            LITERAL | STRING| BOOLEAN | NULL | ID
            ;
findExpr: FIND '(' pip_name (',' '(' expression (','
expression)*')' )*  ')';
pip_name: composed_id;
riskFB: RISKFB '('composed_id (',' composed_id)* ')' ;
/********** Lexer **********/
/* Type */
T_REAL : 'real';
T_INT   : 'int';
T_STRING: 'string';
```

```
T_BOOLEAN: 'boolean';
/* Keyword*/
PE_NAMESPACE: 'namespace';
PE_AUTHRULE: 'authRule';
PE_SESSION: 'session';
IMPORT: 'import';
EXECUTE: 'execute';
READ: 'read';
WRITE: 'write';
DELETE: 'delete';
FIND: 'find';
RISKFB: 'RiskFB';
LITERAL: INTLITERAL | REALITERAL ;
STRING:  '"' (ESC|.)*? '"';
BOOLEAN: 'true' | 'false';
NULL: 'null';
ID : LETTER ( LETTER | Digit | '_' )*;
INTLITERAL: '-'? Digit+;
REALITERAL: '-'? Digit+'.'Digit+;
Digit : JavaIDDigit;
fragment ESC : '\\"' | '\\\\'  ;
fragment LETTER
    :  '\u0024' |
       '\u0041'..'\u005a' |
       '\u005f' |
       '\u0061'..'\u007a' |
       '\u00c0'..'\u00d6' |
       '\u00d8'..'\u00f6' |
       '\u00f8'..'\u00ff' |
       '\u0100'..'\u1fff' |
       '\u3040'..'\u318f' |
       '\u3300'..'\u337f' |
       '\u3400'..'\u3d2d' |
       '\u4e00'..'\u9fff' |
       '\uf900'..'\ufaff'
    ;

fragment JavaIDDigit
    :  '\u0030'..'\u0039' |
       '\u0660'..'\u0669' |
       '\u06f0'..'\u06f9' |
       '\u0966'..'\u096f' |
       '\u09e6'..'\u09ef' |
       '\u0a66'..'\u0a6f' |
       '\u0ae6'..'\u0aef' |
       '\u0b66'..'\u0b6f' |
       '\u0be7'..'\u0bef' |
       '\u0c66'..'\u0c6f' |
       '\u0ce6'..'\u0cef' |
       '\u0d66'..'\u0d6f' |
       '\u0e50'..'\u0e59' |
```

```
       '\u0ed0'..'\u0ed9' |
       '\u1040'..'\u1049'
   ;

COMMENT : '/*' .*? '*/' -> skip // match anything between /* and */
      ;
WS : [ \r\t\u000C\n]+ -> skip
   ;

LINE_COMMENT : '//' ~[\r\n]* '\r'? '\n' -> skip
          ;
```

Appendix B

Parole Script Example

```
namespace enclave_01{
    namespace gpu{
        string name, addr, manufacturer, status;
        real value;

        import sri.*;
        import sri.member.*;

        authRule sri_member {
            REQ.name in sri.member;
        }
        authRule gpu_admin {
            REQ.name in find(sri.member, (role="gpu_admin")) &&
REQ.device.name in sri.member;
        }

        session execute{
            //calculate risks
            sri_member:
                RiskFB(sri_member.fcl, value, history) <= "medium";
            gpu_admin:
                RiskFB(gpu_admin.fcl, history) <= "low";
                //REQ.location == "sri", freq = 6000;
        }
    }
}

/*
namespace db {
    namespace dataset{
        string name, category, sec_level;
        authRule db_admin{
            REQ.name in find(sri.member, (role="db_admin")) &&
REQ.device in sri.member.device;
            find(sri.member, (name=REQ.name, role="db_admin",
status="active")) != null;
        }

        session execute{
            db_admin:
```

```
                RiskFB(db_admin.fcl) <= "medium";
        }
        session write{
            db_admin:
                RiskFB(db_admin.fcl, history, sec_level) <= "low";
                REQ.location == "sri", freq = 60000;
        }
        session read{
            db_admin:
                RiskFB(db_admin.fcl, history) <= "low";
                REQ.location == "sri", freq = 60000;
        }
        session delete{
            db_admin:
                RiskFB(db_admin.fcl, history) <= "low";
                REQ.location == "sri", freq = 60000;
        }
    }
}
*/

namespace sri {
    namespace member{
        string name, role, password, expireDate;
        namespace device{
            string name, ip, manufacturer;
        }
    }
}
```

Appendix C

sri_member.fcl

```
FUNCTION_BLOCK sri_member

VAR_INPUT
    value : REAL;
    historical_access_records : REAL;
END_VAR

VAR_OUTPUT
    risk : REAL;
END_VAR

FUZZIFY value
    TERM low := (0, 1) (1000, 0);
    TERM medium := (900, 0) (1000, 1) (2500, 1) (10000, 0);
    TERM high := (2000, 0) (3000, 1) (30000, 1);
END_FUZZIFY

FUZZIFY historical_access_records
    TERM bad := (0, 1) (3, 0);
    TERM average := (2, 0) (5, 1) (8, 0);
    TERM good := (7, 0) (10, 1);
END_FUZZIFY

DEFUZZIFY risk
    TERM low := (0, 1) (3, 0);
    TERM medium := (2, 0) (5, 1) (8, 0);
    TERM high := (7, 0) (10, 1);
    METHOD : COG; // Center of Gravity defuzzification method
END_DEFUZZIFY

RULEBLOCK No1
    AND : MIN; // Use 'min' for 'and' (also implicit use 'max' for
'or' to fulfill DeMorgan's Law)
    ACT : MIN; // Use 'min' activation method
    ACCU: MAX;

    RULE 1 : IF historical_access_records IS bad THEN risk IS high;
    RULE 2 : IF value IS high AND historical_access_records IS good
THEN risk IS medium;
    RULE 3 : IF value IS high OR historical_access_records IS
average THEN risk IS high;
```

```
    RULE 4 : IF value IS medium AND historical_access_records IS
good THEN risk IS low;
    RULE 5 : IF value IS medium OR historical_access_records IS bad
THEN risk IS high;
    RULE 6 : IF value IS low AND historical_access_records IS
average THEN risk IS low;
    RULE 7 : IF value IS low AND historical_access_records IS bad
THEN risk IS medium;
     RULE 8 : IF value IS low OR historical_access_records IS good
THEN risk IS low;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Appendix D

gpu_admin.fcl

```
FUNCTION_BLOCK gpu_admin

VAR_INPUT
    historical_access_records : REAL;
END_VAR

VAR_OUTPUT
    risk: REAL;
END_VAR


FUZZIFY historical_access_records
    TERM bad := (0, 1) (3, 0);
    TERM average := (2, 0) (5, 1) (8, 0);
    TERM good := (7, 0) (10, 1);
END_FUZZIFY

DEFUZZIFY risk
    TERM low := (0, 1) (3, 0);
    TERM medium := (2, 0) (5, 1) (8, 0);
    TERM high := (7, 0) (10, 1);
    METHOD : COG; // Center of Gravity defuzzification method
END_DEFUZZIFY

RULEBLOCK No1
    AND : MIN; // Use 'min' for 'and' (also implicit use 'max' for
'or' to fulfill DeMorgan's Law)
    ACT : MIN; // Use 'min' activation method
    ACCU: MAX;

    RULE 1 : IF historical_access_records IS bad THEN risk IS high;
    RULE 2 : IF historical_access_records IS average THEN risk IS
medium;
    RULE 4 : IF historical_access_records IS good THEN risk IS low;
END_RULEBLOCK

END_FUNCTION_BLOCK
```