Advanced Zero Trust Architecture for automating fine-grained access control with generalized attribute relation extraction

Nakul GHATE[†] [a)], ***Nonmember***, **Shohei MITANI**[†], ***Member***, **Taniya SINGH**[†], ***Nonmember and*** **Hirofumi UEDA** [†], ***Member***

SUMMARY The paper proposes a novel zero trust architecture which aims to achieve fine-grained access control with low cost. Fine-grained network access control is challenging to implement because of lack of information caused by encryption or vendor specific payload as well as the high cost of design and management of access control designed with fine-grained high-level access policy. We propose an architecture leveraging automated policy generation scheme to achieve fine-grained network access control with low design and implementation cost. The core component is attribute relation extraction which enables the execution of access request evaluation. The policy decision is made by a refined network access policy without requiring the explicit definition of high-level policy, saving processing time and storage cost achieving low cost access control.

Keywords: *Zero Trust architecture, high-level attributes, fine-grained network access control, automatic policy, attribute relation extraction*

1. Introduction

The increasing complexity of enterprise network infrastructure has become a security issue to the cybersecurity experts. The onset of remote-connection, bring-your-own-device (BYOD) policy and cloud services render the perimeter based security solutions useless as the current infrastructure has no defined boundaries. An attacker inside the perimeter can easily perform lateral movements to breach security. Thus, while approaching towards a security model in the current scenario, it is important to consider that an attacker is always present in the environment. Hence, in the recent years, the network security trend has shifted more towards models which assume lack of trust in users and devices. One such example is the Zero Trust architecture (ZTA) [1]. ZT security model assumes that any enterprise environment is not trustworthy and an attacker is always present in the environment. The goal of a ZT model is to prevent any unauthorized access to resources and perform decision making by the principle of least privilege for subject(s) over resource(s) by evaluating each access request in a dynamic and granular fashion using an access control mechanism. The access control incorporates the definition of access policies. The privilege permissions for subjects to access resources make up the high-level access policy definition. These policies may be conventionally defined by policy administrators [2] or extracted from Natural Language access control list [3] and are used to make access decisions. A high-level access policy is based on the application layer attributes and any other contextual attributes, also called as high-level attributes. It includes subject attributes such as user authorization and authentication credentials, device states, contextual attributes such as location, time and object attributes such as resource type, resource owner, etc. [2]. The Attribute Based Access Control (ABAC) policy [2] is widely used in enterprises for access control in ZTA. ABAC uses a high-level access policy to evaluate these attributes to achieve fine-grained access control. However, ABAC policy cannot be enforced for network access control because the network access control is based on the evaluation of network attributes such as IP, port, etc. despite of deployability whereas high-level access policy is not applicable on network attributes. The network packets do not carry high-level attributes making it impractical to apply high-level access policy for achieving fine-grained network access control as shown in figure 1.

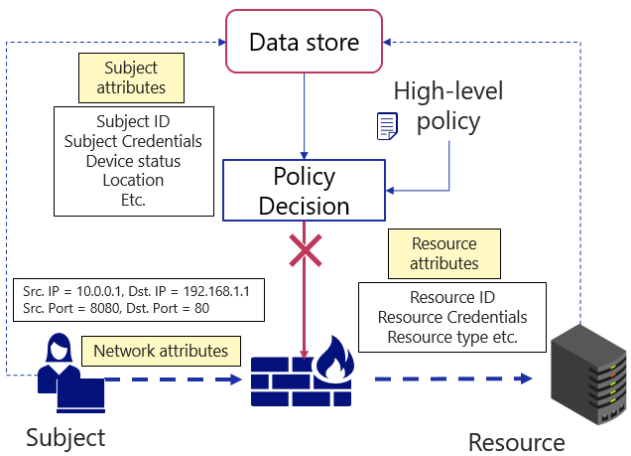

Figure 1. Inconsistency between access request attributes and evaluated attributes in zero trust network.

Berger et al (DynFire) [4] assumed that some binding can be established between IP address and User ID during the login process. Using such a binding, the Policy controller module can know the information about user attributes such as location, time of access, roles assigned to user and their authorization credentials. By evaluating these attributes using a high-level policy such as ABAC policy, it is possible to output the access policies for firewalls. In another study, Jacob et al (FURZE) [5] use two policy languages, one to express high-level policies and other to define generic firewall rules (network-level policies), and they are coordinated with the use of a context handler.

[†]The author is with Secure systems Lab, NEC Corporation
[a)] E-mail: te.nak14@nec.com

While both of them attempts to achieve network access control using high-level policy definition, the access control implemented by both the methods is zone-based access control which is coarse grained. Berger et al used only the binding between source IP and user ID and did not consider any attribute bindings on the resource and/or destination IP and port, etc. This restricted their capability to enforce fine-grained network access control. Secondly, both the architectures do not mention about how the access policy definition is obtained. Instead they work on a manually defined high-level policies obtained from the policy database and focus on the translation of these policies into enforceable firewall rules. Static high-level policies pose design and management challenges as it becomes tedious and error prone to define them manually, and it increases the storage cost and processing time of access control [2]. On the other hand, automatically generated policy based access control suffers from large storage and processing time of access control due to its fine granularity (lots of independent rules) and also has high management cost due to low interpretability of generated policies.

This paper focuses on these limitations and attempts to achieve a fine-grained access control based on attribute evaluation using automatically generated policy definition. We propose an architecture leveraging automated policy generation scheme to achieve fine-grained network access control with low design and implementation cost. In the core of this architecture lies the attribute relation extraction which enables the execution of access request evaluation. In relation extraction module, we present a data structure to represent relations in a general manner which enables the evaluation of network access flow with low storage and processing time. In addition to that, our architecture supports dynamic zoning to reduce the policy management cost and processing time.

2. Proposed Architecture

The proposed architecture is shown in Figure 2. The principal components of our architecture are subject, resource, Data store, the Relation Extraction Module (REM), Policy Generation Module (PGM), Policy Refinement module (PRM), Zone Optimize Module (ZOM), Policy Decision Point (PDP) and the Policy Enforcement Point (PEP).

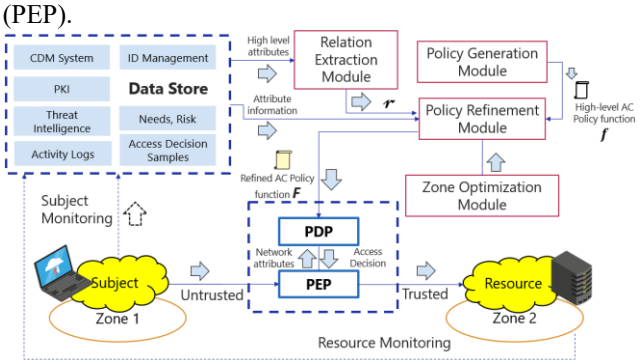

Figure 2. Proposed architecture of ZTA network access control

A subject can be a person or a device acting on behalf of a person [2] acting as requester to access a resource. The access flow is intercepted by the PEP which monitors the network traffic between subject and resource. The PDP is responsible for making access decisions. The Policy generation module (PGM) generates a fine-grained high-level policy in a general form, i.e. a function to output desirable decisions for the set of high-level attributes. It is not necessarily a set of interpretable rules. The refined executable network access policy is obtained with the Policy Refinement Module (PRM) based on the high-level policy. This refinement relies on the Relation extraction module (REM). This module binds the network attributes with high-level attributes. PRM uses a wide variety of information from the data store about these attributes useful for evaluation. Lastly, Zone Optimization Module (ZOM) refines the definition of zones to maximize manageability without reducing fine-granularity of network access control. This architecture is designed on the premise of automated policy generation for finer-granularity and lower policy definition cost than the conventional ZTA. Our architecture aims to achieve this advancement with feasibility, especially on the network access control mechanism. The core component (REM) is capable to accept a fine-grained high-level policy in a general form, and enables the execution of access decisions cooperating with the other component (PRM). The detailed components are described next.

2.1 Data store

The data store is a collection of various information, about the subjects and resources managed by the network. It stores and continuously monitors user activities such as session logs, authentication check, authorization, device state monitoring, etc. In addition, it may also receive data from sources such as threat intelligence, and SIEM systems, collecting information about security, threats, etc. The information inside the data store makes up the high-level attributes which are evaluated in PRM for access decision.

2.2 Generalized bindings

In Berger et al [4], a binding between user and IP address allowed policy controller to know about the related attributes to the user for evaluation. However, no such attributes related to resource are known for evaluation resulting in coarse grained access control. The notion of generalized binding is to obtain the set of all high-level attributes which can affect the decision of access control corresponding to a given access request. As such, our proposed relation extraction module translates the set of network attributes which represent the access attributes to the high-level attributes which impact the decision.

2.2.1 Relation Extraction module

In this section we formulate the relations between the attributes. The purpose is to bind the network attributes with high-level attributes for fine-grained access control using a high-level policy. The relation extraction module outputs the set of attributes to be evaluated for access decision. It uses attribute relations to obtain the set of high-level attributes. A relation between two attributes imply the logical binding of one attribute with the other. For instance, which user ID is using which device, or which application is hosted at which port. These relations are called direct relations. Let A be the set of all attributes in the system and for each attribute $a_i \in A$, v_{a_i} be the value of a_i which varies in the set V_{a_i}. Then for a pair of attributes $a_i, a_j \in A$, $V_{a_i} \times V_{a_j}$ will be the set of all possible "attribute:value" pairs written in the Cartesian product. A direct relation $R_{a_i \to a_j} \in \mathcal{R}$ is defined as $R_{a_i \to a_j} \subset V_{a_i} \times V_{a_j}$, the subset for which there exists some logical bindings between the pair of values $\left(v_{a_i}, v_{a_j}\right) \in V_{a_i} \times V_{a_j}$. Here, $\mathcal{R}$ is the set of all such direct relations. The attribute set A is divided into the high-level attributes A_H and the network attributes A_N. Therefore, two types of direct relations exist: $R_{a_{H_i} \to a_{H_j}} \in \mathcal{R}$ which define the direct relations between the two high-level attributes $a_{H_i}, a_{H_j} \in A_H$ and $R_{a_{N_i} \to a_{H_j}} \in \mathcal{R}$ which define the direct relations between network-level attribute and high-level attribute $a_{N_i} \in A_N, a_{H_j} \in A_H$. The attributes form the vertex V and the direct relation form the edges E of the graph $G = (V, E)$. Each graph G_i comprises of a direct relation $R_{a_{N_i} \to a_{H_1}}$, $a_{N_i} \in A_N, a_{H_1} \in A_H$, and several direct relations $\left\{R_{a_{H_1} \to a_{H_2}}, R_{a_{H_2} \to a_{H_3}}, \dots, R_{a_{H_j} \to a_{H_{j+1}}}, \dots\right\}$ between high-level attributes, $a_{H_j} \in A_H$. Such graphs are shown in figure 3. A path in the graph G_i is the one connecting related attribute values, defined as $path: \left(v_{a_{N_i}}, v_{a_{H_1}}, v_{a_{H_2}}, \dots\right)$ $s.t.$ $(v_{a_{N_i}}, v_{a_{H_1}}) \in E, (v_{a_{H_j}}, v_{a_{H_{j+1}}}) \in E$. The indirect relation is proposed to connect several such graphs in order to infer meaningful relationships between source and destination attributes. The indirect relation is the combined path in a collection of such graphs $\{G_1, G_2, ..\}$. The relation extraction module proposed in our architecture generates a function which outputs the combined path implying the relationships among the high-level attributes at the source and destination. The relation extraction module therefore outputs a function r which takes the source and destination network attribute values as inputs and outputs a combined path as the set of related high-level attributes values. r: $V_{a_{N_1}} \times V_{a_{N_2}} \times \dots V_{a_{N_i}} \times \dots \to V_{a_{H_1}} \times V_{a_{H_2}} \times \dots V_{a_{H_j}} \times \dots$ $s.t.$ $v_{a_{H_1}}, v_{a_{H_2}}, \dots \in r\left(\dots, v_{a_{N_i}}, \dots\right) if \ \exists \ path: \left(v_{a_{N_i}}, v_{a_{H_1}}, v_{a_{H_2}}, \dots\right) in \ G_i$.

For a given access request ($Src.IP =$ 192.168.1.1, $Dst\ IP =$ 192.168.1.10, $Dst\ Port =$ 443), several paths are generated. The source path $R_1 =$ ($Src.IP =$ 192.168.1.1, $Device =$ Device0, $User =$ Alice, $Role =$ Finance Manager), the destination paths $R_2 =$ ($Resource =$ Financial records, $Dst\ IP =$ 192.168.1.10) and ($Storage\ Device =$ Finance server, $Dst\ Port =$ 443) are combined to obtain the indirect relation (192.168.1.1, 192.168.1.10, 443, Device0, Alice, Finance Manager, Financial records, Finance server). Such indirect relations enable attribute feature evaluation based on a general-form of high-level access policy described next.

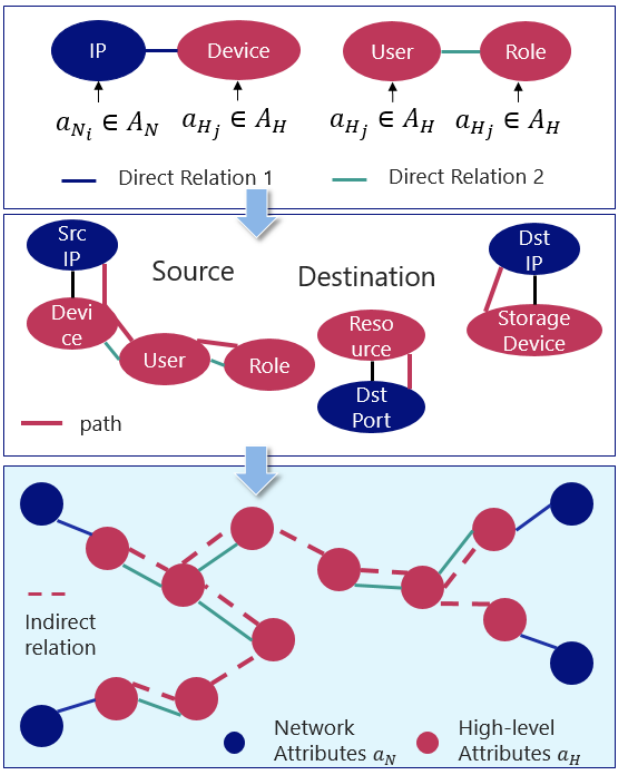

Figure 3. Relation extraction method to output set of related high-level attributes

2.3 Attribute evaluation and access decision

The access decision is made using evaluation of attributes. The information about the attribute features is evaluated to decide if the access should be accepted. There are both positive and negative information on their monitored status to decide an action. The high-level policy can be represented by a function instead of rules, i.e. it can be modeled as a function explicitly describing the trade-off between positivity and negativity. For e.g., trust scores may be assigned to attributes such as users and devices based on their health status, such as whether authorization is performed, or whether the device has latest security system installed. High trust score has a positive effect on decision. The set of attributes whose feature information is to be evaluated is the output of the relation extraction module.

2.3.1 Policy Generator Module

The Policy Generator Module (PGM) outputs a function $\boldsymbol{f}$ to represent high-level access policies. In our proposed architecture the $\boldsymbol{f}$ maps the set of values of related high-level attributes in the indirect relation to an access decision action y. Thus the PGM outputs the function $\boldsymbol{f}: V_{A_{H_1}} \times V_{A_{H_2}} \times \ldots \rightarrow Y$. There can be numerous ways to define such a function $\boldsymbol{f}$ which represents a high-level policy. One example of such function is described in [4] which allows access to certain resource only if the value of the subject credentials is larger than a pre-defined threshold.

2.3.2 Policy Refinement Module and policy decision

The Policy Refinement Module (PRM) updates the network-level access policies using the high-level access policy, attribute information and the indirect relations as inputs. It outputs the function $\boldsymbol{F}$ which maps the set of network-level attributes to the access control action $y \in Y$, $\boldsymbol{F}: V_{A_{N_1}} \times V_{A_{N_2}} \times \ldots \rightarrow Y \;\; s.t. \;\; \boldsymbol{F} = \boldsymbol{f} \circ \boldsymbol{\mathcal{r}}$, where y can be a set of access control decisions, e.g. {allow : 1, deny : 0, etc.}. It evaluates the positive and negative effects of the attribute features information. The evaluation is performed according to the high-level policy $\boldsymbol{f}$ obtained from the PGM. The resulting function $\boldsymbol{F}$ outputs network-level policies. The PDP receives the set of network attributes $a_N \in A_N$ and its values from the PEP for a given access request. It then uses the refined Policies $\boldsymbol{F}$ obtained from the PRM to decide the action 'y' by evaluating the set of network attributes against the refined policy as $y = \boldsymbol{F}(v_{a_{N_1}}, v_{a_{N_2}}, \ldots)$. The PDP is logically connected to the PEP at the network end points in order to reduce the latency in access control.

2.3.3 Zone Optimization Module

The ZOM (described in [6]) represents the network policies in a set of group C to reduce the complexity of large number of independent rules. For the network attributes $x \in X$, it proposes a grouping function $\zeta: X \rightarrow C$. The zone access control policy function can be written as $F_Z: C_s \times C_D \rightarrow Y$ where C_s, C_D are source and destination zones respectively. It optimizes the balance between fine-granularity and manageability for the feasibility of our advanced ZTA.

3. Use-case scenario

Two users in an organization, Alice, a "Finance Manager" and Bob, an "IT technician" log into the system. The Data store monitors the attributes of both the users and notify about their values to the PRM. The PRM utilizes the relation extraction function and the high-level access policy function to create network-level refined policies. A high-level policy can be "allow access if the sum of operational need (x_1) and trust score (x_2) is greater than 1" which can be represented as $y = \{1 \;\; if \;\; x_1 + x_2 > 1 \;\; else \;\; 0\}$. This is generated by PGM without manual design. The formula $x_1 + x_2$ and the threshold "1" are automatically defined. The PRM quickly updates the refined access control policy if the values of attribute vary with time. ZOM reduces the complexity of the policy refined by PRM at each time by grouping the network attributes, e.g. IP, port. The grouped attributes in a zone share a common rule. It makes it easy for operators to check adequacy and modify the rules as needed. Based on the assigned zones, Alice and Bob can only access certain resources described by their attributes as shown in Figure 4.

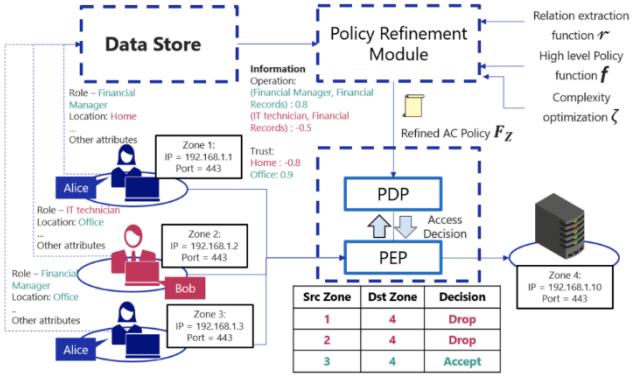

Figure 4. Representation of a Use-case scenario

5. Conclusion and Future work

We present a new architecture to realize the advanced zero trust feasible to utilize the automatic policy definition for fine-grained access control and low-cost of policy creation. Our proposed architecture is aimed to accept general shaped policy with low storage, small processing time, and high manageability. The evaluation of performance, scalability and security on a real enterprise network are left for future work.

References

[1] S. W. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture." *NIST Special Publication* 800: 207, 2020

[2] V.C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, ... and K. Scarfone, "Guide to attribute based access control (abac) definition and considerations (draft)." *NIST special publication* 800.162 pp. 1-54, 2013

[3] M. Alohaly, H. Takabi, H., and E. Blanco, "Automated extraction of attributes from natural language attribute-based access control (ABAC) policies." *Cybersecurity* 2.1, pp. 1-25, 2019

[4] S. Berger, A. Vensmer, and S. Kiesel, "An abac-based policy framework for dynamic firewalling." *International Conference on Systems and Network Communications (ICSNC 2012)*. Vol. 2012, pp. 118-123, 2012.

[5] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access control policy enforcement for zero-trust-networking." *2018 29th Irish Signals and Systems Conference (ISSC)*. IEEE, pp. 1-6, 2018.

[6] S. Mitani, T. Singh, N. Ghate, and H. Ueda, "Attribute-based low-complexity network access control policy with optimal grouping algorithm." *IEICE Communications Express,* 2021.