Federating Trust: Network Orchestration for Cross-Boundary Zero Trust

Karl Olson
University of Colorado - Boulder
Boulder, CO
karl.olson@colorado.edu

Eric Keller
University of Colorado - Boulder
Boulder, CO
eric.keller@colorado.edu

1 INTRODUCTION

Zero Trust is an emerging security paradigm that does away with implicit zones of trust commonly employed within static, defense-in-depth, enterprise architectures. One of the core tenets of Zero Trust is that resource access is determined by dynamic policy - an intersection of trust in a user, the supporting application or service, the underlying network, and the devices which hold or process data. Establishing this overall assessment of trust serves well for centralized architectures where an administrator can establish and assess each of these trust enablers, such as in an enterprise network. However, shifting workloads to remote access, bring your own device (BYOD), and cloud hosting of collaborative services, to name a few, all challenge the ability of an administrator to effectively establish a complete Zero Trust architecture due to the inability to fully trust each component.

This shift away from centrally managed architectures reveal a significant challenge in achieving complete Zero Trust: security is a function of many interactions, many of which an administer has no control over. Recently the term "Zero Trust 2.0" was coined as an evolution to Zero Trust which establishes identity as the new perimeter via an orchestration layer and machine learning capabilities [1]. However, this functionality still remains tied to centrally controlled architectures where an administrator can link together products and solutions to achieve a desired level of security. We argue that this orchestration needs to expand beyond these common enterprise boundaries in a way that trust can be guaranteed across disparate systems, networks, and servicers. Similar to identity federation, where a user can use credentials from one provider to access another competitors platform, federation of trust should serve as a guarantee for security *across* networks. In the remaining sections we propose what this trust federation mechanism could potentially look like.

2 DESIGN

NIST 800-207 outlines a number of tenets for a Zero Trust deployment such as per-session assessment, use of dynamic policy, and strict policy enforcement prior to access [2]. In designing our distributed trust mechanism, we align with these core tenets of Zero Trust to establish four high-level design objectives which serve as fundamental primitives to guide our design:

(1) Trust across a boundary should be established prior to any follow-on communication or resource access.
(2) A user should be able to specify their security requirements, and the target should be able to respond without revealing any potential negative information about their network (eg. "we do not have the latest security patch installed.")
(3) Similarly, a recipient should also be able to request a set of security requirements from the user in order to maintain a level of trust in their own network.
(4) The trust system should be able to dynamically respond to changes in conditions in order to maintain security of the network (eg. a vulnerability in an agreed upon condition is identified.)

In order to meet these objectives, we therefore require a design that integrates a number of requirements:

(1) A mechanism for both the sender and receiver to specify conditions necessary to establish a level of trust.
(2) A verification process for sender and receiver to respond to trust requirements, and if necessary propose alternatives.
(3) A handshake or agreement process to finalize security requirements before processing further communication
(4) Integration with analytic or security components of a network to dynamically assess conditions of trust within a network
(5) An inform process to inform either sender or receiver of changes in conditions.
(6) A perimeter or DMZ-based negotation mediator to handle trust setup (trust is established external to target host to prevent system exposure prior to establishing trust)

Based on the above core design objectives and requirements, we envision an architecture as presented in Figure 1. Here, a client would first establish a session with the target network's trust proxy, represented by numeral 1. The use of a trust proxy is to serve as a mediator of trust prior to allowing

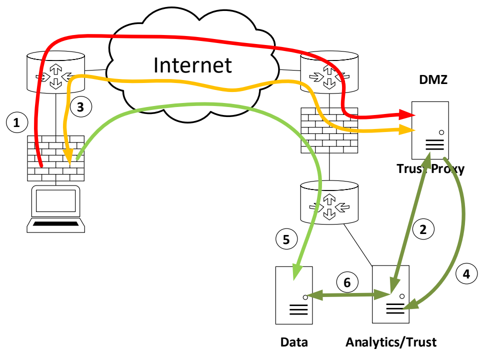

Figure 1: Trust Proxy Architecture. The colored lines represent the degree of trust in each step.

access to a resource. We envision this as an external/DMZ component where trust negotiation would occur, and in the case of failure, does not expose or allow access to a network resource, eg. fail-safe. This proxy would work with an organization's analytic/policy infrastructure to dynamically understand the current security conditions, requirements, and state of trust (numeral 2).

With an updated assessment of security conditions, the trust proxy would enter into trust establishment process with the requesting system. We envision this negotiation process as a four-way handshake to agree on security conditions, shown in Figure 2 and discussed in detail with our poster.

If both the sender and proxy agree to a set of security requirements (such as patch compliance, chosen encryption methods, hardware or software versioning, etc.) , a security token is generated by the proxy for the session and provided to both the requester and the organizations internal trust/analytics platform. Security mechanisms, such as a firewall, are also updated to recognize the generated token and any traffic tied to the session. The purpose of the security token is to: 1) identify a requester as having completed a negotiation process based on a set of agreed upon conditions, 2) identify all traffic associated with the session and its operating conditions, such that a change in network policy or status could quickly block or prevent further communication easily, 3) serve as a secondary verification for a target resource to verify access is authorized. We describe this token process in more detail within our poster. With this security token, a requester would then be given access to a resource (numeral 3). In turn, the resource would verify the token (numeral 4) before fulfilling any requirement.

3 DISCUSSION/FUTURE WORK

One assumption we make in this design is that the response provided by both the requester or the trust proxy are true and not malicious in intent. To ensure this, we envision a few potential options:

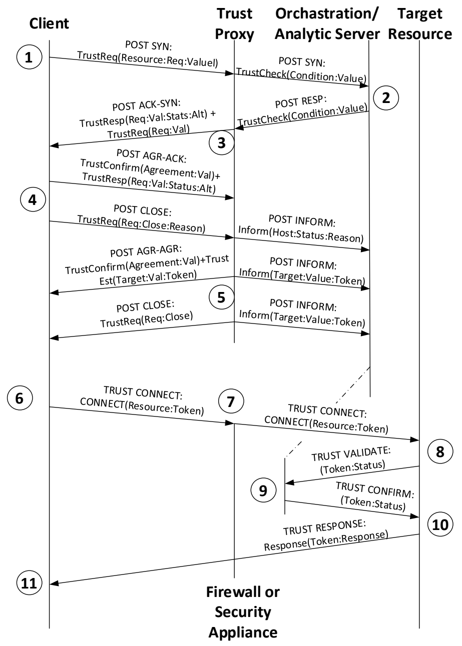
Client
Trust Proxy
Orchastration/ Analytic Server
Target Resource
1
POST SYN: TrustReq(Resource:Req:Valuel)
POST SYN: TrustCheck(Condition:Value)
2
POST RESP: TrustCheck(Condition:Value)
POST ACK-SYN: TrustResp(Req:Val:Stats:Alt) + TrustReq(Req:Val)
3
POST AGR-ACK: TrustConfirm(Agreement:Val)+ TrustResp(Req:Val:Status:Alt)
4
POST CLOSE: TrustReq(Req:Close:Reason)
POST INFORM: Inform(Host:Status:Reason)
POST AGR-AGR: TrustConfirm(Agreement:Val)+Trust Est(Target:Val:Token)
POST INFORM: Inform(Target:Value:Token)
5
POST CLOSE: TrustReq(Req:Close)
POST INFORM: Inform(Target:Value:Token)
6
TRUST CONNECT: CONNECT(Resource:Token)
7
TRUST CONNECT: CONNECT(Resource:Token)
8
TRUST VALIDATE: (Token:Status)
9
TRUST CONFIRM: (Token:Status)
10
TRUST RESPONSE: Response(Token:Response)
11
Firewall or Security Appliance

Figure 2: Trust Federation Negotiation Process. Complete walkthrough will be presented in poster.

(1) for a host, metrics could be tied to remote attestation via the trusted platform module (TPM). This could be useful for enterprise systems where configuration could be verified against local policy requirements.
(2) a standardized hierarchical trust architecture, similar to that afforded public certificates. Here a network could tie trust to a certificate system which embeds security metrics within the certificate.
(3) 3rd party negotiation - here networks could attest to their clients status based on compliance with local security policy. This would work best in environments where both parties have a mature architecture that can assure status of systems, whether local or remote.

Other potential considerations for federating trust are the potential processing overheads and efficiency of negotiation. While the proposal in current form introduces overhead and additional processing latency, the costs should be offset by the need for assured trust in data and systems first and foremost. This assumption would need to be revisited for high-throughput environments.

REFERENCES

[1] Amir Nooriala. 2020. Zero Trust 2.0: The Perfect Balance Between Convenience and Security. (2020).
[2] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. 2019. *Zero trust architecture*. Technical Report. National Institute of Standards and Technology.