bzt: Berkeley Zero Trust

Developing a zero-trust networking solution using common primitives

Authors: Max Russell, Bill Froemming

Implicitly trusting network traffic is no longer a tenable posture for internet connected organizations. The overhead of maintaining access for users to resources over time when the users and their services span the globe has forced pattern shifts for organizations like the United States Department of Defense. Zero trust networking (ZT) is a security model that fundamentally changes how organizations view network security. Legacy architectures often use perimeters of trust to segment friend from foe into zones, where being in a friendly zone means that actors or traffic have already passed some check and are implicitly trusted. The essence of ZT is that no user, device, or network connection should inherently be trusted, regardless of location, history, or previous access rights. ZT has become essential in modern networking due to dissolving network perimeters, sophisticated cyber attacks, cloud adoption, and the proliferation of internet of things (IoT) devices. *bzt* constructs a ZT architecture with simple, common primitives to form an effective environment absent of the pitfalls of traditional environments.

Due to its difficulty of adoption, age, and high complexity, many network environments have not adopted ZT, or even fully understand the tenants for proper implementation. It is possible to distill ZT's goals into several principles; continuous verification, where every access attempt is authenticated and authorized before granting access to resources, is the first. Under the principle of least privilege, users are given minimum levels of security to perform their job function. ZT networks manifest as micro-segmented, small, isolated segments in order to limit potential damage in the event of a breach. Finally**,** access policies that are continuously updated based on risk-assessments occurring in real-time are examples of dynamic policy reinforcement (Mjcaparas). Successful ZT patterns are able to take the described elements and produce a cohesive whole, such as off the shelf tools like Cisco Systems' Cisco Secure Client, or Fortinet's FortiClient.

When evaluating ZT patterns, the seeming complexity may at first appear daunting, but using existing and common primitives it is possible to construct and implement ZT quickly and easily. *bzt* is built with IPsec, Netfilter rules, REST API's and small portable programs that allow users start their ZT trivially. *bzt* enforces ZT by setting up a secure communication channel between a client and server via the *bzt-agent* (the agent). The trusted channel from the client to agent endpoint is established through the following process:

1. A secured channel exists between an endpoint (running *bzt-agent*) and the control plane (*bzt-server*).
2. The client requests access to a resource on the endpoint from the control plane. The server takes available information about the state of the requested client into account when determining approval. Once authenticated, details of the encrypted connection channel are saved on the control plane.
3. The *bzt-agent* checks into the server to determine approved connections. New connections are implemented, and expired connections are revoked. The channels are created by explicit accepts in IPsec connections and Netfilter rules, with default and implicit drop statements stopping other traffic.
4. The client can now connect to the authorized resource over a tightly scoped secured medium with mutual authentication in the IP layer.

5. At the conclusion of the allowed session, the *bzt-agent* will close the channel, requiring the client to re-authenticate itself to the control plane to reestablish.

While there are no new protocols used in the *bzt* implementation, the less common IPsec transport mode is used for the secure channel between endpoints. In transport mode, IPsec protects only the payload of the IP packet, leaving the IP header itself intact. As a result, the original IP header remains in place, while the content inside the packet is encrypted and authenticated. The IPsec headers are added between the original IP header and the IP payload. Transport mode provides an efficient use of bandwidth, as unlike in tunnel mode only the payload is encrypted, resulting in less bandwidth used in communications. Transport mode was selected for its ability to provide end-to-end communications, such as securing traffic between two hosts or applications, rather than between networks or gateways.

IPsec transport mode plays a significant role in securing network traffic by allowing devices to encrypt data at the IP layer (Layer 3) without relying on tunneling or virtual IPs. This approach enables regular IP addresses to send all traffic in a secure, encrypted format, which is analogous to the encryption provided by protocols like TLS at the higher layers (Layer 4/5). However, unlike TLS, IPsec transport mode secures data directly at the network layer, ensuring that all routed traffic between two endpoints can be encrypted without application or network termination modifications. While IPsec transport mode shares the goal of securing traffic, it differs from more common VPN solutions, such as IPsec in tunnel mode, Wireguard, or SSL/TLS-based VPNs, as it does not encapsulate the entire packet or require virtual IPs, instead protecting only the payload (Kozierok).

Transport mode is enforced for all traffic destined for endpoints running the agent by way of enforcing inbound traffic to be Encapsulating Security Payload (ESP), and having the network stack reject non-compliant traffic. Dropping packets that are not correct is the ultimate realization of ZT: because traditional, non-encrypted packets cannot be verified, they cannot be trusted. This would mean that *bzt* deployments would be impossible to enter without playing by the enforced rules, ultimately creating a new perimeter based not on network locations but on verifiable chains of trust that stem from the system's asymmetric encryption and control plane.

bzt's design makes it well-suited for environments that embrace a perimeter-less, zero-trust architecture, where networks are public, and Network Address Translation (NAT) or other middlemen that rewrite packets are absent. By moving away from NAT, which can interfere with IPsec by modifying packet headers and causing connectivity issues, networks can adopt a model that emphasizes direct, encrypted communication between devices. Implementing IPsec transport mode would be easier to realize while using IPv6 than IPv4, as IPv6 natively supports unique global addressing without NAT, aligning with the goal of a secure, open, and scalable network structure that is global instead of segmented into perimeters by network administrators.

In the context of the CIA triad, IPsec in conjunction with a control plane that only permits authorized traffic provides robust security for host-to-host communication. Confidentiality is achieved through IPsec, which encrypts the traffic at the network layer, ensuring that data in transit remains unreadable to unauthorized parties. Integrity is maintained as the control plane establishes permissions for each encrypted session, preventing tampered or unauthorized data or devices from reaching its destination. Netfilter further verifies that only authorized traffic types, such as SSH over TCP/22 from an authorized source, are allowed.

Filters can be constructed for Layer 3 or Layer 4 values, which are tightly scoped in *bzt* by the control plane. Availability is impacted in two ways: first, the endpoint's IPsec policy mandates ESP-encrypted traffic. The default drop IPsec policy ensures unencrypted traffic is dropped by the destination host, enforcing stringent security but limiting traffic that does not meet these standards by discarding it. Second, availability of applications may be improved if computing resources are not required to process invalid traffic that is caught by *bzt*'s additional checks and enforcement at Layers 3-5 with its IPsec and Netfilter rules.

ZT is not a new architecture pattern; the term dates to 1994, but was codified in 2010. In 2021, the US DoD was instructed to begin implementing such systems, and other governments with similar requirements of high security, chains of trust, and enforceable protections have followed suit (Rais). With *bzt*, another tool for achieving the desired architecture is introduced: it is simple, uses open primitives that exist across systems and are understood by administrators commonly, and achieves the principles of ZT. When the scale of endpoints required to talk to each other without humans behind the keyboard grows, new ways of managing the system need to be established, and ZT networking is the defensive strategy of the foreseeable future.

Works Cited

1. Mjcaparas. “What Is Zero Trust?” *Microsoft Learn*, learn.microsoft.com/en-us/security/zero-trust/zero-trust-overview. Accessed 21 Nov. 2024.
2. Kozierok, Charles M. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.
3. Rais, Razi, et al. *Zero Trust Networks: Building Secure Systems in Untrusted Network*. O’Reilly Media, Inc, 2024.
4. Zero Trust security | What is a Zero Trust network? (n.d.). cloudflare.com. Retrieved November 12, 2024, from https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/
5. DelBene, K., Medin, M., & Murray, R. (2019, July 9). *The road to zero trust (security)*. The Road to Zero Trust (Security). https://media.defense.gov/2019/Jul/09/2002155219/-1/-1/0/DIB_THE_ROAD_TO_ZERO_TRUST_(SECURITY)_07.08.2019.PDF

Appendix

The areas covered in this submission are “What new or uncommon protocols are used?”, “Discuss Confidentiality / Availability / Integrity”, and “A working demonstration of the thing”, included as code.