Introducing Zero Trust by Design: Principles and Practice Beyond the Zero Trust Hype

Chapter · July 2021

CITATIONS
6

READS
4,056

2 authors:

Dwight Horne
Baylor University
24 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Suku Nair
Southern Methodist University
127 PUBLICATIONS **2,124** CITATIONS
SEE PROFILE

Introducing Zero Trust by Design: Principles and Practice Beyond the Zero Trust Hype

Dwight Horne* and Suku Nair
AT&T Center for Virtualization, Southern Methodist University, Dallas, TX 75275, USA
{rhorne, nair}@smu.edu

Abstract. The zero trust (ZT) threat model is one in which no implicit trust exists. Companies across industries, governments, and more are at various stages of evaluation, planning, and implementation of re-architected networks that deviate from legacy perimeter-centric security models in favor of ZT approaches. A lack of open standards, or even consensus on ZT details and priorities in some cases, has potentially slowed adoption. But recent standardization attempts have taken strides toward filling that gap. ZT research and practice to date have focused predominantly on the network architecture and operations. However, to fully realize the potential benefits of the ZT model, we must expand our conception of ZT. In this paper, we introduce Zero Trust by Design (ZTBD), a set of guiding principles that extend beyond network architecture to also foster ZT software engineering and ZT protocol design, thereby maximizing synergies and potential benefits. Beyond the ZTBD principles, a set of ZT practices connect theory and guidelines with practical applications, while the introduction of ZT patterns provides reusable solutions to common challenges. Moreover, just as ZT research and practice are evolving, so too will ZTBD. We invite the community to further contribute to these efforts and support continued alignment of this growing body of knowledge with the rapid evolution and expansion of zero trust.

Keywords: zero trust by design, ZTBD, zero trust architecture, zero trust software engineering, zero trust protocol design, zero trust patterns, zero trust model.

1 Introduction

Trust no one. That was a recurring mantra from the well-known television show called The X-Files. The Zero Trust (ZT) threat model extends that idea beyond *trust no one [user]* to further include *trust no device* and *trust no network flow*. In an era of global pandemic accelerated pivots to remote work and major supply chain compromises like the SolarWinds attacks of 2019-2020 [1], ZT concepts are everywhere. ZT ranges from myriad vendor marketing campaigns and a recent USA National Institute of Standards and Technology (NIST) standard [2] to ZT conferences [3] and guidance from the National Security Agency of the USA [4]. Yet not all ZT practitioners fully agree on the critical characteristics of zero trust as evidenced by the results from a survey of technology professionals in 2020 [5]. That study found that 85% of respondents reported ongoing pursuit of defined ZT initiatives within their organizations. Meanwhile, only 12% of the technologists responding indicated that "location and method of access do not confer trust" was a defining characteristic of zero trust networking, despite that statement aligning with the second "basic tenet" of ZT as defined in the recently finalized NIST SP800-207 [2], a standard for Zero Trust Architectures. Consequently, one of the goals of Zero Trust by Design (ZTBD) is to harmonize the tenets of ZT from various resources by distilling them down to overarching principles that can be applied to varying contexts beyond the network architecture.

Moreover, just as it is widely recognized that full lifecycle software and systems security is more successful than when it is treated as an afterthought [6], we argue that benefits of the ZT model cannot be fully realized without extending the notion of ZT beyond the network architecture to include ZT protocol design and full lifecycle ZT software engineering. After distinguishing these separate but

complementary notions, we extend the ZTBD principles with good practices for applying the core principles. We further launch an initiative for identifying ZT patterns – a collection of named solutions to commonly recurring problems as an additional enabler for ZT initiatives.

The remainder of this paper is organized as follows. Section 2 provides background information contrasting ZT architectures, ZT software engineering, and ZT protocols. Section 3 then highlights the ZTBD principles and augments them with good ZT practices. Next, Section 4 provides sample ZT patterns that accompany the launch of an initiative to build a large, openly available body of valuable enablers. Lastly, we conclude with an open invitation to the community to further contribute to these efforts for the benefit of all.

2 Background - A Contextual Understanding of Zero Trust

2.1 Zero Trust Security Model

In a ZT security model, no implicit trust exists. That is, all users, devices, network traffic, program inputs/outputs, and more might be considered hostile until proven otherwise depending on the context. In this paper, we consider the abstract idea of the ZT model and its application in certain contexts as an aid for applied researchers and practitioners. For theory related to definitions of trust across disciplines or trust modeling formalizations, please see related resources such as [7] and the works cited therein.

The general notion behind zero trust likely dates back to a time much earlier than when ZT terminology became a regular part of the technology lexicon. Technical solutions specifically targeting a ZT security model by that name date back to at least 2002 with an approach to zero trust intrusion tolerant systems that assumed compromise was inevitable [8]. That approach described self-cleansing systems that periodically restore themselves from a trusted source, a technique that resembles frequent infrastructure churn with trusted image restoration in modern networks and systems. The ZT model was later popularized with application to network architectures, as discussed in Section 2.2. But applicability of the model goes much further than the traditional network architecture centric view of ZT. ZT software engineering and ZT protocol design can be characterized as disparate but complementary applications of the ZT model in related domains. To fully realize the benefits of the ZT model, this comprehensive view of ZT is a necessity.

2.2 Zero Trust Network Architecture

The zero trust model was popularized in the context of network architectures as a complete re-thinking of the traditional, perimeter-centric approach to network security through the writing of Kindervag [9][10]. The early work of [8] described zero trust as “the extreme assumption that all intrusion detection techniques will eventually fail”. Over time, that assumption is no longer viewed as “extreme” by many as companies deal with adversaries that have penetrated networks, very real insider threats, and perimeters that have dissolved with migration to cloud-based infrastructure and applications, along with significant remote work.

A ZT network architecture has been described as having Internet security everywhere, assuming the network is already compromised, and lacking implicit trust that might be common in legacy network architectures. The main logical components of the ZT architecture as defined by NIST SP800-207 include trust and policy engines composing the policy decision point (PDP) where access decisions are made. The PDP then communicates the access determinations in some way to the policy enforcement point (PEP) where the access controls are enforced. A sample trust evaluation process that might be used with a ZT network appears in Fig. 1. For more details regarding ZT network architectures, a number of useful resources exist such as [2], [11], and [12].

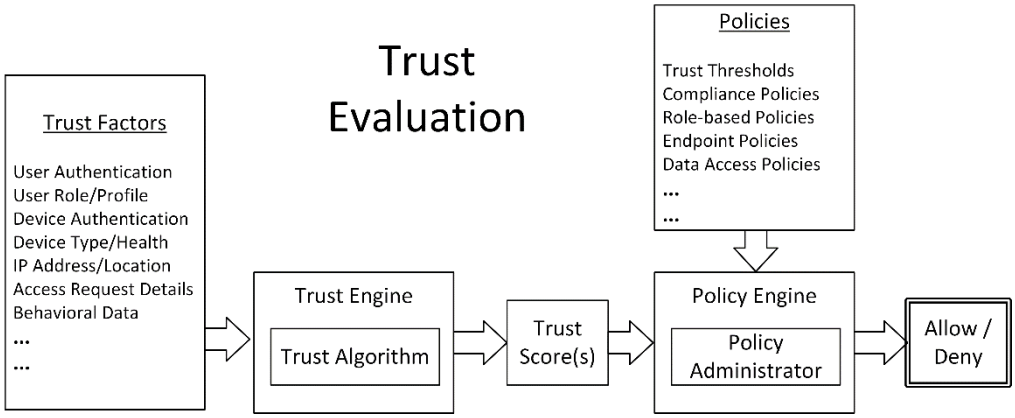

Fig. 1. An example trust evaluation process with zero trust network architectures.

2.3 Zero Trust Software Engineering

While ZT network architectures have received significant attention, ZT software engineering deserves designation as a separate but complementary domain. Viewing software engineering through the lens of the ZT security model can help facilitate a more complete realization of potential security and privacy benefits. ZT software engineering is not a major deviation from traditional software engineering, but rather a tailoring of existing requirements determination, design/development/deployment processes, and an overall change in mindset.

ZT software engineering explicitly recognizes the evolving nature of the threat landscape, the adaptations necessary to fully realize the potential of ZT networks, and features required to function properly within the context of ZT frameworks. For example, software applications must support the goals of the ZT network architecture in areas such as authentication, authorization, appropriate logging levels, and more in an environment with Internet security everywhere. Software lacking support for key features of the network architecture could inhibit ZT ambitions. In fact, Gilman and Barth devoted an entire chapter of their book to "Trusting Applications" in the context of ZT networks [11].

Moreover, insecure software could serve as the weakest link introducing unwanted vulnerabilities that might serve to circumvent controls. In an increasingly API driven world, consider that Gartner suggested "API abuse" would be the most common attack category by 2022 [13]. Perhaps more alarming, a report from Salt Security suggested that 91% of enterprise security professionals reported incidents in 2020 involving API security and that the rate of growth of malicious API traffic exceeded that of non-malicious API traffic [14]. Clearly, software requirements driven by the needs of a ZT network and secure software development practices, particularly for interfaces such as APIs, are critical elements of ZT software engineering.

2.4 Zero Trust Protocols

Protocol design plays a foundational role in support of both ZT architectures and ZT software engineering. Even the most secure networks and applications would fail to afford adequate protections with the use of insecure protocols. For this reason, it is important to recognize ZT protocol design as an essential enabler and contributor to ZT success. ZT protocols can be divided into at least two broad categories including ZT networking protocols and domain specific ZT protocols.

Zero Trust Networking Protocols. Gilman and Barth claim that ZT networks use existing technologies in novel ways, but do not require new protocols or libraries [11]. While that may be true in some sense, ZT networking can potentially be strengthened by new protocols, particularly if standardized with a focus on interoperability. Indeed, some researchers have already recognized this. For instance, [15] presents autonomic security for ZT networks with protocols and software to facilitate log parsing, orchestration, and threat response support. Meanwhile, others have worked to develop a generic firewall policy language to support protocols for dynamic access control policy enforcement [16]. Moreover, other lines of research have aimed to demonstrate how transport access control and first packet authentication protocols can be effectively used to implement ZT cloud networks [17]. Secure protocol design is expected to continue to play a foundational role in the evolution of ZT software and networks.

Domain Specific Zero Trust Protocols. The ZT model can also be used in the design of novel, domain specific protocols. There are a number of examples of protocols that embody one or more ZT principles, although they do not necessarily use ZT terminology directly. For example, the Bitcoin electronic cash protocols utilize a distributed ledger and hash-based proof-of-work to avoid reliance on a trusted third party and to offer a level of pseudo-anonymity [18]. The Monero cryptocurrency uses a different protocol with an obfuscated ledger attempting to achieve a greater level of anonymity, again assuming a lack of trust in third parties [19]. Privacy enhanced matchmaking technologies such as [20] and [21] exhibit various ZT principles applied to novel matchmaking protocols. Yet another example established protocols for ZT hierarchical trust management in IoT networks [22]. Domain specific protocol designs inspired by a ZT model are expected to continue to be active areas of research, likely with increasing adoption of practical technologies leveraging such protocols going forward.

1. **Trust no one (and no thing)**
   Assume compromise. No user, device, packet, or input is implicitly trusted.
2. **What perimeter?**
   Secure perimeters have morphed or disappeared in a cloud-centric world with remote work and insider threats. There are no implicit trust zones.
3. **Apply the principle of least privilege**
   Limit access to the minimum required; deny by default where feasible.
4. **Dynamic, risk-based policies**
   Use dynamic, contextual risk assessment and strict policy enforcement.
5. **Require strong authentication and authorization**
   All resource access requests and network flows must be authenticated and authorized.
6. **Log, monitor, inspect, and adapt**
   Traffic monitoring, input validation, status logging, analysis, alerts, adaptive trust levels, issue prevention and recovery actions.
7. **Employ defense in depth**
   Secure devices and communications; secure the weak links; think like an adversary and increase work factor for lateral movement.
8. **Full lifecycle, zero trust security**
   Full lifecycle security under a zero trust model; security is not an afterthought.
9. **Confidentiality and integrity by default**
   Encrypt all communications; protect data at rest; verify integrity.
10. **Strike the right balance with tradeoffs**
    Cost vs. benefit; security vs. usability; cost proportional to risk/impact.

Fig. 2. Foundational principles of Zero Trust by Design (ZTBD) v1.0

3 Introducing Zero Trust by Design

3.1 Core Principles of Zero Trust by Design

The principles of Zero Trust by Design (ZTBD) aggregate and harmonize core elements of the approach and communicate the essence of the ZT model, whether applied to network architecture, software engineering, protocol design, or another domain. The ZTBD principles codify fundamental considerations that are critical to the ZT mindset. The principles for ZTBD v1.0 appear in Fig. 2.

3.2 Beyond the Core Principles – Zero Trust Practices

Use of the ZTBD principles can facilitate application of the ZT model in specific contexts. Beyond the core principles, this work further attempts to capture ZT practices that have proven to be beneficial for translating the principles into practice. Notice that these are presented as good practices rather than best practices. This is because they have been shown to be good practices in many contexts, but claiming a universal best practice can be risky as it only takes one counter-example to prove otherwise.

Good Practices for ZT Networks. Application of the ZTBD principles can translate to a number of good practices for transforming high level principles of ZT network architectures to practical security controls. Some of such good practices include the following.

Strongly and securely identify users with multi-factor authentication. Strong identity validation is essential for trust evaluation and access control. Leverage multiple factors for authentication such as something the user knows (e.g., password), is (e.g., biometrics), and possesses (e.g., token).

Securely provision and strongly identify devices. Managed devices should be securely configured with trusted images, securely provisioned, and strongly identified, which may for example include use of a Trusted Platform Module (TPM) with certificate based authentication schemes like X.509.

Enforce device security. Devices should be in the most secure state possible at all times. Enforce timely patching of OS, firmware, and applications. Consider patch level and security posture as a factor in trust evaluation. Leverage appropriate security tools for the context such as antivirus, host-based firewalls, intrusion prevention software, and activity monitoring. Consider tailored access privileges and custom policies based on device types or functional characteristics. For example, one might treat a provisioned company laptop differently than an IoT network device or personal phone in a BYOD scenario.

Adopt an image refresh strategy. Somewhat akin to the original ZT work of [8], develop a strategy to restore from trusted base images where feasible. Consider the age of a device image to be a risk factor under the assumption that more usage time since original image installation implies greater probability of compromise.

Use dynamic policy enforcement and trust scoring. Evaluate trust based on as many pertinent data points as possible. Trust evaluation can potentially provide the greatest benefit if considering a variety of factors that may contribute to the level of trust such as user ID, device ID, sensitivity of target resource, temporal data, geographic data, and real-time threat intelligence data. While a core tenet of ZT security is that location alone should not confer trust, note that location could potentially be one of a number of factors considered when assessing risk depending on the context.

Employ re-authentication policies for users and devices, but balance security with usability in the process. There are risks associated with one-time authentication and permanent trust thereafter. It is common to adopt a periodic re-authentication strategy based on temporal constraints or continual risk assessments. However, when developing a re-authentication strategy, balance risk management with usability impact.

Leverage micro-segmentation and micro-perimeterization. Segregate networked resources strategically to facilitate least privilege access control policies and reduce risks. This can be accomplished in different ways depending on the context. For example, security gateway appliances such as firewalls, hypervisors, virtual network overlays, software-defined micro-perimeters, routing zones, Layer 2 subnetworks, and more. Micro-segmentation and software defined perimeters have become commonplace in complex cloud environments [23][24] and efforts to apply micro-segmentation to new areas such as 5G cellular networks [25] will also continue to be explored. Regularly monitor for the latest research trends and vendor offerings in this category for future enhancements.

Continually monitor network traffic, configure alerts, and adapt policies as needed. Ensure that continuous monitoring is part of the network security strategy. Additionally, log network traffic for post analysis, investigation support, training data sets for machine learning driven security solutions, and more. Make use of risk-based alerting features for risk detection and rapid response.

For more complex application scenarios, adopt a cross-functional ZT working group. The involvement of a cross-functional group, whether for actual decision making or advisory consultation purposes, can help to ensure that all stakeholders' interests are considered with ZT initiatives in complex environments.

Good Practices for ZT Software Engineering and ZT Protocol Design. Application of the ZTBD principles can also translate to good practices for software engineers and protocol designers. Some of such good practices include the following.

Encrypt all communications. There is no longer a "safe" internal network inside a secured perimeter (perhaps there never was!). Perimeters are dissolving. Even the best perimeter defenses can be circumvented, and the insider threat is real. Securely encrypt communications to the fullest extent practicable.

Authenticate/authorize network connections. Here again, there is no safe internal network guarded by a perfectly secured perimeter. Assume connection requests and packets are hostile until proven otherwise. One example of this might be via mutually authenticated Transport Layer Security (TLS).

Secure the software supply chain. The SolarWinds attack of 2020 significantly impacted many government agencies and Fortune 500 companies, and was a stark reminder of the criticality of securing the software supply chain [1]. The entire supply chain must be secured including code/artifact repositories, dependencies, build systems, DevOps continuous integration pipelines, and more.

Maximize automated use of security analysis tools. Beyond human security reviews, maximize use of value-added tools like static code analysis / static application security testing (SAST), dynamic application security testing (DAST), vulnerability scanners, and fuzzing tools.

Secure design and coding practices. Use appropriate practices like threat modeling, mandatory secure coding guidelines, security focused design and code reviews, and code signing.

Take API security very seriously. Despite broad encouragement to adopt secure coding practices, vulnerability counts continue to rise as seen in Fig. 3, which reflects annual totals from the NIST National Vulnerability Database (NVD). Since APIs have become a cornerstone of modern application development, ensuring API security is essential. More information on API security can be found in resources such as [26], [13], and [27].

Adopt privacy by design. Be mindful of applicable privacy regulations and adopt privacy by design as part of the full lifecycle security and privacy practices. Avoid the use of personally identifiable information in protocol messages or permanent storage to the fullest extent possible.

Design protocols to support access proxies for externalized applications and services. Access proxies can add a level of indirection and provide features such as enforcement of access controls, traffic anomaly detection, denial-of-service protection, and more. A good example of this can be seen with Google's BeyondCorp implementation [28].

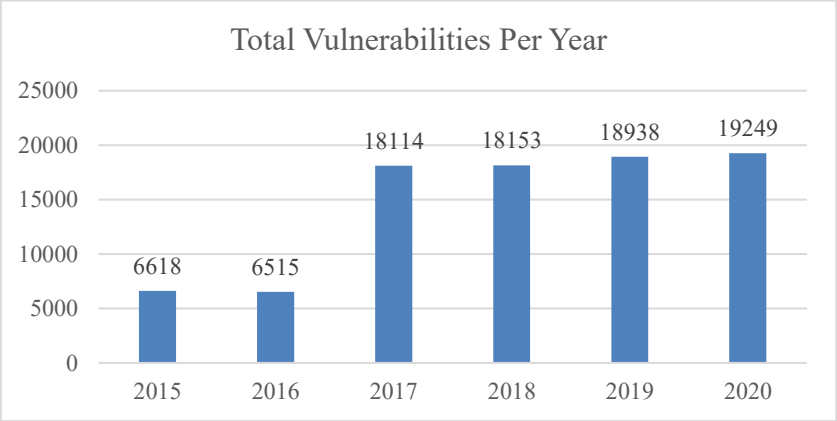

Fig. 3. Total number of vulnerabilities per year recorded in the NIST National Vulnerability Database (https://nvd.nist.gov).

4 Enhancing Zero Trust by Design with Zero Trust Patterns

4.1 Zero Trust Patterns – A Key Enabler for Practitioners

The 23 software design patterns from [29] have had a remarkable impact on software engineering through the re-use of effective solutions to commonly recurring problems. That success has fostered the identification of additional software design patterns, as well as patterns in other areas like security [30]. Beyond establishment of the ZTBD principles and a growing set of good ZT practices, current efforts include further aid to practitioners in the form of ZT patterns. A few examples of such ZT patterns are given here, while many more will continue to be added to the growing ZT body of knowledge (ZTBOK) accompanying ZTBD.

Zero Trust Software Design Patterns. This category of ZT patterns facilitate software engineering success with identified solutions to common problems in software system design. The ZTBOK includes a growing set of patterns in areas such as API security, authentication and authorization, effective multi-level logging, and more. For instance, consider the following example.

Pattern Title: Application Layer for Confidentiality

Problem: I need to ensure confidentiality of information exchanged.

Solution: Use application layer encryption for end-to-end confidentiality guarantees.

Details: Applications should not rely on confidentiality mechanisms at lower layers. While approaches such as link-level point-to-point encryption and VPNs can afford some advantages, they can leave traffic vulnerable outside of the encryption context. Use encryption between endpoints at the application layer for confidentiality. Also, be sure to use strong cipher suites.

Zero Trust Networking Patterns. Similarly, the ZTBOK includes a growing list of patterns targeting success in ZT networks. The following example represents a pattern in this category.

Pattern Title: Prefer a Private PKI

Problem: I need to choose the right PKI for my certificate based ZT authentication schemes.

Solution: Prefer a private PKI system over public PKI entities.

Details: While public PKIs are a good fit for many situations, the implementation of ZT networks tends to result in a large growth in the number of certificates and private PKIs generally have more benefits and fewer risks. Private PKIs can offer clear cost advantages and more control over the certificate creation and rotation processes as well as more flexible automation opportunities. Moreover, with the very large number of public PKI systems, it brings into question the extent of the trust that can confidently be afforded to third parties. For more on the public vs. private PKI discussion for ZT networks, refer to [11].

Zero Trust Protocol Patterns. The third category of patterns in the ZTBOK assist ZT protocol designers, although some could also help with ZT domains beyond protocol design. The following examples represent two sample ZT protocol design patterns.

Pattern Title: Hybrid Post-Quantum Confidentiality

Problem: My protocol uses key exchange algorithms like RSA and ECDH that are vulnerable to compromise by quantum computing enabled adversaries with Shor's algorithm.

Solution: Use post-quantum cryptography. But full confidence cannot be placed in candidate quantum-resistant cryptographic algorithms prior to standardization, or possibly for some period of time thereafter. Combine classical and quantum-resistant algorithms in a hybrid scheme that leverages both approaches to retain the soundness of the strongest approach in the event that the other is weakened or compromised.

Details: NIST in the USA has invited global cooperation for development, testing, and standardization of quantum-resistant public-key cryptographic algorithms. At the time of writing, draft standards were planned for the 2022-2024 timeframe [31]. In the meantime, hybrid post-quantum protocols are being developed for research, testing, and preparation for security against quantum enabled adversaries. The post-quantum hybrid Transport Layer Security (TLS) [32][33], a hybrid post-quantum protocol for fair and privacy-enhanced matchmaking [34], and X.509 compliant hybrid certificates [35] are three examples reflecting the main idea embodied by this pattern.

Pattern Title: Single Packet Authorization (Alt: First / Single Packet Authentication)

Problem: How can I limit my service to authorized packet flows only as part of my zero trust initiatives and thereby reduce my attack surface?

Solution: Use Single Packet Authorization (SPA) where the payload of the first packet includes authentication/authorization information. Packets are passively monitored and dropped by default if not appropriately pre-authenticated as an authorized flow.

Details: SPA packets are encrypted and typically include fields such as username, date/time stamp, protocol version, message type, access request related information, random data to prevent replay attacks, and a message digest. One example of practical SPA tools is fwknop [36][37]. Although SPA was distinguished from port knocking and analyzed quite some time ago [38], attempts to further improve SPA [39][40][41] and to integrate first packet authorization with ZT protocols and cloud networks such as [17] continue. Note in the title and aliases that there is a clear difference between authentication and authorization, but that topic is beyond the scope of this paper and is addressed at [36].

5 Conclusion and Future Work

5.1 Fostering the Next Generation Zero Trust Ecosystem

ZT is not a security appliance to buy or a one-time project. ZT is a realistic security model with a new mindset appropriate for the modern threat landscape with borderless networks, insider threats, remote workers, cloud applications, and advanced persistent threats. In this paper, we aggregated and

harmonized guidance from disparate ZT resources, distilling them down in the form of the Zero Trust by Design (ZTBD) core principles. We further provided a set of good practices to foster successful ZT initiatives and extended ZT concepts beyond the network architecture to encompass ZT protocol design and ZT software engineering, which are separate but complementary domains that further facilitate ZT success. The sample ZT patterns represent additional enablers accompanying ZTBD v1.0. Given modern security threats and perimeter-less networks, adoption of the ZT model continues to accelerate.

5.2 An Open Invitation to the Community

Lastly, we extend an open invitation to the community to visit ZeroTrustByDesign.com and contribute to the growing ZTBD body of knowledge. The site also includes links to many useful ZT references to facilitate further ZT learning. We invite feedback on the core principles, additional good practices, and supplementary ZT patterns as we evolve toward ZTBD v2.0 and work together to foster the next generation of the zero trust ecosystem.

References

1. Peisert, S., Schneier, B., Okhravi, H., Massacci, F., Benzel, T., Landwehr, C., Mannan, M., Mirkovic, J., Prakash, A. and Michael, J.B.: Perspectives on the SolarWinds incident. IEEE Security & Privacy, 19(02), pp. 7-13 (2021).
2. Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero trust architecture. NIST Special Publication 800-207, National Institute of Standards and Technology, U.S. Department of Commerce (2020).
3. DevSecOps and zero trust architecture (ZTA) for multi-cloud environments, January 27, 2021, https://www.nist.gov/news-events/events/2021/01/devsecops-and-zero-trust-architecture-zta-multi-cloud-environments, last accessed 2021/04/24.
4. Embracing a zero trust security model, United States (US) National Security Agency (NSA), February 25, 2021, https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF, last accessed 2021/04/24.
5. McGillicuddy, S.: Enterprise Zero Trust Networking Strategies: Secure Remote Access and Network Segmentation, Enterprise Management Associates, Inc., pp. 1-32 (2020).
6. McGraw, G.: Software Security: Building security in. 1st edn. Addison-Wesley Professional (2006).
7. Cho, J. H., Chan, K., and Adali, S.: A survey on trust modeling. ACM Computing Surveys (CSUR), *48*(2), pp. 1-40 (2015).
8. Sood, A.K., Huang, Y., Simon, R., White, E., and Cleary, K.: Zero trust intrusion containment for telemedicine, George Mason University, Fairfax, VA, USA (2002).
9. Kindervag, J.: Build security into your network's DNA: The zero trust network architecture. Forrester Research Inc., pp. 1-26 (2010).
10. Kindervag, J. and Balaouras, S.: No more chewy centers: Introducing the zero trust model of information security. Forrester Research, pp. 1-13 (2010).
11. Gilman, E. and Barth, D.: Zero Trust Networks, O'Reilly Media, Incorporated (2017).
12. Kerman, A., Borchert, O., Rose, S., and Tan, A.: Implementing a zero trust architecture, The MITRE Corporation, Tech. Rep. (2020).
13. O'Neill, M., Zumerle, D., and D'Hoinne, J.: How to build an effective API security strategy. Gartner, Inc. (2017).
14. State of API Security Q1 2021. Salt Security, 2021, https://salt.security/api-security-trends, last accessed 2021/04/26.
15. Eidle, D., Ni, S. Y., DeCusatis, C., and Sager, A.: Autonomic security for zero trust networks. In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pp. 288-293, IEEE (2017).
16. Vanickis, R., Jacob, P., Dehghanzadeh, S., and Lee, B.: Access control policy enforcement for zero-trust-networking. In 2018 29th Irish Signals and Systems Conference (ISSC), pp. 1-6, IEEE (2018).

17. DeCusatis, C., Liengtiraphan, P., Sager, A., and Pinelli, M.: Implementing zero trust cloud networks with transport access control and first packet authentication. In 2016 IEEE International Conference on Smart Cloud (SmartCloud), pp. 5-10, IEEE (2016).
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf (2008).
19. Noether, S.: Ring signature confidential transactions for monero, IACR Cryptol. ePrint Arch., 1098 (2015).
20. Shin, J. S. and Gligor, V. D.: A new privacy-enhanced matchmaking protocol, IEICE Trans. Commun, 96(8), pp. 2049-2059 (2013).
21. Horne, D. and Nair, S.: A new privacy-enhanced technology for fair matchmaking with identity linked wishes, IEEE Systems Journal, 14(1), pp. 298-309 (2019).
22. Samaniego, M. and Deters, R.: Zero-trust hierarchical management in IoT. In 2018 IEEE international congress on Internet of Things (ICIOT), pp. 88-95, IEEE (2018).
23. Klein, D.: Micro-segmentation: securing complex cloud environments, Network Security, 2019(3), pp. 6-10 (2019).
24. Moubayed, A., Refaey, A., and Shami, A.: Software-defined perimeter (sdp): State of the art secure solution for modern networks, IEEE Network, 33(5), pp. 226-233 (2019).
25. Mämmelä, O., Hiltunen, J., Suomalainen, J., Ahola, K., Mannersalo, P., and Vehkaperä, J.: Towards micro-segmentation in 5G network security. In European Conference on Networks and Communications (EuCNC 2016) Workshop on Network Management, Quality of Service and Security for 5G Networks (2016).
26. Siriwardena, P.: Advanced API security: OAuth 2.0 and beyond, second edition, Apress (2020).
27. OWASP API Security - Top 10 | OWASP, https://owasp.org/www-project-api-security/, last accessed 2021/04/29.
28. Ward, R., and Beyer, B.: Beyondcorp: a new approach to enterprise security, ;login, USENIX (2014).
29. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design patterns: abstraction and reuse of object-oriented design. In European Conference on Object-Oriented Programming, pp. 406-431, Springer, Berlin, Heidelberg (1993).
30. Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., and Sommerlad, P.: Security Patterns: Integrating security and systems engineering, John Wiley & Sons (2013).
31. Post-Quantum Cryptography | CSRC, National Institute of Standards and Technology (NIST), United States Department of Commerce, https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline, last accessed 2021/04/29.
32. Crockett, E., Paquin, C., and Stebila, D.: Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. *IACR Cryptol. ePrint Arch.,* 858 (2019).
33. Paquin, C., Stebila, D., and Tamvada, G.: Benchmarking post-quantum cryptography in TLS. In *International Conference on Post-Quantum Cryptography* (pp. 72-91). Springer, Cham (2020).
34. Horne, D. and Nair, S.: Toward a quantum resistant secretmatch privacy enhanced technology–a case study. In Proceedings of the International Conference on Security and Management (SAM), pp. 147-153 (2019).
35. Bindel, N., Braun, J., Gladiator, L., Stöckert, T., and Wirth, J.: X.509-compliant hybrid certificates for the post-quantum transition, Journal of Open Source Software, 4(40), 1606 (2019).
36. Rash, M.: Single packet authorization with fwknop, https://www.cipherdyne.org/fwknop/docs/SPA.html, last accessed 2021/04/29.
37. Rash, M.: Single packet authorization – a comprehensive guide to strong service concealment with fwknop, https://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html#design, last accessed 2021/04/29.
38. Jeanquier, S.: An analysis of port knocking and single packet authorization, MSc Thesis, University of London (2006).
39. Srivastava, V., Keshri, A. K., Roy, A. D., Chaurasiya, V. K., and Gupta, R.: Advanced port knocking authentication scheme with QRC using AES, In 2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), pp. 159-163, IEEE (2011).
40. Mehran, P., Reza, E. A., and Laleh, B.: SPKT: secure port knock-tunneling, an enhanced port security authentication mechanism. In 2012 IEEE Symposium on Computers & Informatics (ISCI), pp. 145-149, IEEE (2012).
41. Lucion, E. L. R. and Nunes, R. C.: Software defined perimeter: improvements in the security of single packet authorization and user authentication. In 2018 XLIV Latin American Computer Conference (CLEI), pp. 708-717, IEEE (2018).