tria

Building Secure Software in a Zero Trust World

Tria Federal White Paper

triafed.com

Building Secure Software in a Zero Trust World

1. Introduction

In today's digital landscape, software security is more critical than ever. With the increasing sophistication of cyber threats, traditional perimeter-based security models are no longer sufficient. The Zero Trust security model assumes that no user or device can be trusted by default and requires strict verification for every access request.

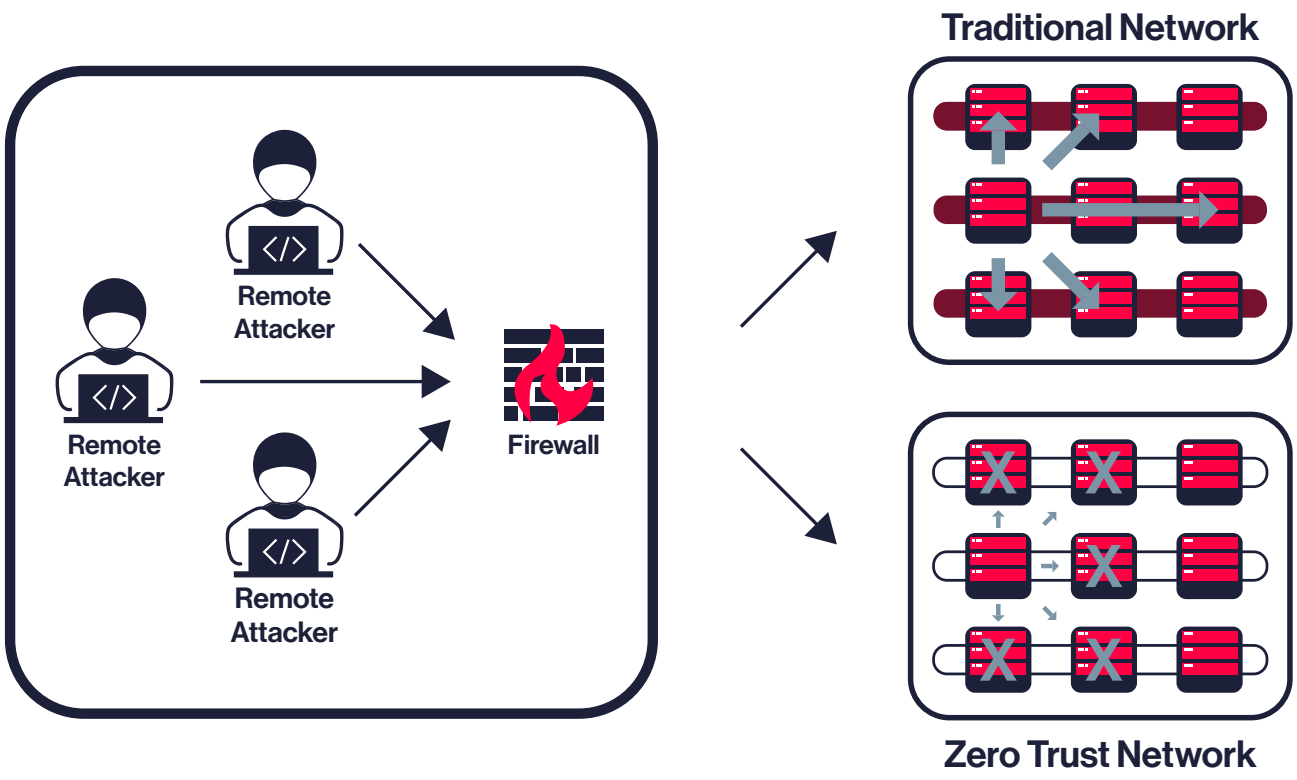
This white paper explores the importance of secure coding practices, security testing, and continuous monitoring within the Software Development Life Cycle (SDLC), and how they integrate with Zero Trust principles to create robust and resilient software.

2. What is Zero Trust?

Zero Trust is a security framework that assumes no user or device can be trusted by default and requires strict verification for every access request regardless of the user's location or device. It shifts the focus from perimeter-based defense to a data-centric approach where least privilege access is enforced.

The federal government has recognized the importance of Zero Trust and has issued mandates to agencies to adopt this model, including **Executive Order 14028** and **OMB memorandum M-22-09**. These mandates aim to improve the cybersecurity posture of federal agencies and protect sensitive data by requiring the implementation of Zero Trust principles and practices.

3. Zero Trust vs. Trust-Based Network



4. With Zero Trust Security

- **Breach is assumed:**
  Expect that attackers have already breached your system and design security controls accordingly.

- **Nothing is inherently trusted:**
  Everyone and everything, inside or outside the network, must be verified.

- **Access is granted by need:**
  Only give users and devices the minimum access required for their jobs.

- **Verification is continuous:**
  All users, devices, and requests are fully verified before granting access.

- **Networks are segmented:**
  Divide networks into smaller segments to limit the breach.

5. What Zero Trust is Not

Perimeter-based security models, which focus on securing the network perimeter and assume that everything inside the network is trusted, are not Zero Trust. This includes traditional security measures like firewalls, intrusion detection systems, and virtual private networks (VPNs).

These solutions alone are not enough in a Zero Trust architecture, as they do not verify every access request and do not assume that everything inside the network is untrusted. Additionally, implicit trust in any user or device, regardless of their location or network, goes against the Zero Trust model.

FEATURE	ZERO TRUST SECURITY	TRADITIONAL PERIMETER-BASED SECURITY
Core Principle	"Never trust, always verify"	"Trust but verify" (within the perimeter)
Trust Assumption	No implicit trust, assumes breach	Implicit trust within the network perimeter
Security Focus	Users, devices, and applications, regardless of location	Network perimeter (firewalls, VPNs)
Access Control	Granular, based on user identity, device posture, context, and least privilege	Broader, often based on network location or user role
Network Segmentation	Micro-segmentation, isolating resources and limiting lateral movement	Limited segmentation, larger zones of trust
Authentication & Authorization	Continuous verification at every access request	Authentication at network entry, less frequent authorization within
Threat Detection	Continuous monitoring and analytics, inside and outside the network	Focus on external threats, less visibility into internal activity
Data Protection	Data-centric, focuses on protecting data regardless of location	Perimeter-centric, relies on network security to protect data
Attack Surface	Reduced attack surface due to granular access control and micro-segmentation	Larger attack surface, potential for lateral movement within the network

6. The Goal of Zero Trust-Based Software Development

The goal of a Zero Trust initiative is to maximize data security by maximizing the confidentiality, integrity, and availability of data. This is essential to maintaining trust and ensuring business continuity. Zero Trust leverages tools including:

- **Encryption:**
  Implementing encryption at rest and in transit safeguards data from unauthorized access and tampering.

- **Data Integrity Checks:**
  Using hashing algorithms and digital signatures helps verify data integrity and prevent unauthorized modification.

- **Immutable Infrastructure:**
  Adopting immutable infrastructure, where servers and components are never modified directly after deployment, enhances security and reliability.

7. Zero Trust and Software Development

Embracing Zero Trust within software development necessitates a fundamental shift in perspective, where secure coding practices are not merely a checkbox but an integral component of the development lifecycle. This entails shifting security to the left in the SDLC, ensuring that security is addressed from the earliest stages of design and development. Secure coding serves as the bedrock upon which the Zero Trust model is built, as it proactively addresses vulnerabilities and misconfigurations that could be exploited by malicious actors. By integrating secure coding practices from the outset, organizations can significantly reduce the attack surface and mitigate potential risks, thereby aligning their software development efforts with the core principles of Zero Trust. This synergy between secure coding and Zero Trust not only enhances the overall security posture but also fosters a culture of security awareness throughout the development process.

Zero Trust and secure coding practices complement each other by creating a multi-layered defense strategy. Secure coding practices proactively prevent vulnerabilities during development, while Zero Trust principles continuously verify trust and access, even if the code has unforeseen flaws. This combination ensures that even if one layer of defense fails, the others can still protect the system and data.

Educating developers on secure coding principles early in a project is crucial because it prevents the introduction of vulnerabilities early in the development process, saving time and resources by avoiding costly fixes later. This proactive approach ensures that security is integrated into the software's foundation, making it inherently more resilient to attacks.

Security must be integrated into every phase of the SDLC to ensure that software is built with a strong security foundation. During the **requirements gathering and analysis phase**, security requirements should be identified and prioritized.

In the **design phase**, security measures should be incorporated into the software's architecture and design.

During **implementation**, secure coding practices should be followed to prevent vulnerabilities.

In the **testing phase**, both manual and automated security testing should be conducted to identify and address any security flaws.

After **deployment**, continuous **monitoring and change management** should be in place to detect and respond to any security threats. By addressing security at each stage of the SDLC, organizations can proactively mitigate risks and ensure the software's overall security posture.

8. SDLC Phases and Zero Trust Alignments

SDLC PHASE	ZERO TRUST ALIGNMENT
Requirements Gathering & Analysis	Define "protect surface" – the data, assets, applications, and services (DAAS) most critical to the organization. Map data flows and dependencies.
Design	Design for least privilege access to DAAS. Implement micro-segmentation to isolate DAAS and limit lateral movement.
Implementation (Coding)	Enforce least privilege through code. Integrate with Identity Access Management (IAM) systems for authentication and authorization. Implement contextual access controls based on user, device, and location.
Testing	Verify that only authorized users and devices can access DAAS. Test micro-segmentation effectiveness. Simulate attacks to test lateral movement prevention.
Deployment	Deploy with least privilege access by default. Enforce micro-segmentation in production. Continuously monitor and log access attempts.
Monitoring and Change Management	Continuously monitor and validate access controls. Adapt micro-segmentation rules based on changing threats and business needs. Maintain up-to-date inventory of DAAS.

9. Design Considerations When Incorporating Zero Trust-Based Secure Development

Risk assessment methodologies, threat modeling, and micro-segmentation are essential components of a secure development process that aligns with Zero Trust principles. Incorporating these practices during the initial phases of development is crucial for establishing a strong security foundation and proactively mitigating potential risks.

Risk assessment methodologies enable organizations to identify and prioritize potential threats, allowing them to allocate resources effectively and focus on the most critical security concerns. By conducting thorough risk assessments early in the development process, organizations can make informed decisions about security controls and design choices, reducing the likelihood of vulnerabilities and misconfigurations.

Threat modeling complements risk assessment by providing a structured approach to identifying and analyzing potential attack vectors. By applying threat modeling frameworks like **STRIDE** or **DREAD** during the design phase, developers can proactively address security weaknesses and ensure that the software is resilient to various attack scenarios.

Micro-segmentation enhances security by dividing the network into smaller, isolated segments, limiting the lateral movement of attackers and containing the impact of security breaches. By incorporating micro-segmentation into the software's architecture from the outset, organizations can create a more defensible environment that aligns with the Zero Trust principle of least privilege access.

Integrating these practices early in the SDLC ensures that security is not an afterthought but a core consideration throughout the development process, resulting in more secure and resilient software that aligns with the principles of Zero Trust.

10. Secure Coding Practices Applicable to Zero Trust

The primary objective of secure coding practices in the context of Zero Trust is to **proactively address and prevent vulnerabilities and misconfigurations** that could be exploited by malicious actors. Examples of secure coding techniques which support the Zero Trust model include:

- **Input Validation:**
  Ensuring that all data input into a system is checked for appropriate formatting and type to prevent malicious data from causing unexpected behavior or vulnerabilities.

- **Output Encoding:**
  Converting data from one format to another to prevent it from being misinterpreted or used to inject malicious code.

- **Parameterized Queries:**
  Structuring database queries so that user input is separated from the actual query command, preventing injection attacks.

- **Code Reviews:**
  Systematic examination of source code by other developers to find and fix errors and security flaws that were missed in the initial development process.

- **Least Privilege:**
  Limiting access rights for users, applications, and processes to only what is necessary to perform their tasks.

- **Explicit Verification:**
  The process of proactively and continuously verifying the identity, authorization, and trustworthiness of users and devices before granting them access to resources.

11. Security Testing and Continuous Monitoring

Testing and continuous monitoring are necessary for Zero Trust development because they assist in identifying vulnerabilities and security threats in real time, even after the software has been deployed. This aligns with the Zero Trust principle of continuous verification, ensuring that trust is never assumed, and that security is constantly maintained. Continuous monitoring also creates a feedback loop which can foster continuous design improvement. Some tools and methodologies include:

- **Security Testing:**
  Employing a combination of static, dynamic, interactive, and penetration testing helps identify vulnerabilities throughout the SDLC.

- **Automated Security Testing Tools:**
  Leveraging automated tools for security testing increases efficiency and enables early detection of vulnerabilities.

- **Manual Security Testing:**
  While automation is valuable, manual security testing remains crucial for identifying complex vulnerabilities and business logic flaws.

- **Continuous Monitoring:**
  Implementing continuous monitoring, logging, and auditing helps detect and respond to security threats in real time.

- **Anomaly Detection:**
  Utilizing rule-based and machine learning-based techniques for anomaly detection allows for proactive identification and mitigation of potential security incidents.

12. Best Practices for Zero Trust Deployment

Here are some best practices for Zero Trust deployments:

- **Start with an understanding of your goals and priorities:**
  What are you trying to protect? What are your biggest security risks?

- **Take a phased approach:**
  Don't try to implement everything at once. Start with a few key areas and gradually expand your Zero Trust implementation over time.

- **Focus on identity and access management:**
  This is the foundation of Zero Trust. Make sure that there are strong authentication measures in place and that granular access controls are being enforced.

- **Use micro-segmentation to isolate sensitive data and applications:**
  This can help to limit the impact of a breach.

- **Monitor your network activity closely:**
  This will help to identify and respond to threats and vulnerabilities quickly.

13. Conclusion

Building secure software in a Zero Trust world requires a multi-faceted approach that encompasses secure coding practices, rigorous security testing, continuous monitoring, and a commitment to data protection. By integrating Zero Trust principles into the SDLC and adopting a proactive security posture, organizations can develop software that is resilient to modern threats and safeguards sensitive data.

14. Resources

NIST Overview of Zero Trust - https://www.nist.gov/publications/zero-trust-architecture

Secure Development Lifecycle NIST 800-218 V 1.1 - https://csrc.nist.gov/pubs/sp/800/218/final

Zero Trust Advancement Center - https://cloudsecurityalliance.org/zt

15. Acknowledgements

Paul Misner, CISSP, is a Cybersecurity Subject Matter Expert for Tria Labs. Paul specializes in the areas of Threat Hunting and SOC management.

tria

Tria Federal White Paper

triafed.com