OVERVIEW: DEFENSE IN DEPTH

Our organization has implemented a comprehensive security policy, and the Defense in Depth (DiD) framework is at the heart of our security policy. Defense in Depth (DiD) emphasizes a multi-layered approach to security. Multiple security controls are implemented at different levels to create overlapping layers of defense. This approach helps mitigate risks and protect against threats and vulnerabilities.

THREATS MATRIX

* Likely threats: Phishing attacks, ransomware infections, social engineering attempts.
* Priority threats: Insider threats, third-party breaches, password attacks.
* Low-priority threats: Natural disasters, physical security breaches.
* Unlikely threats: Zero-day attacks, advanced persistent threats.

10 PRINCIPLES

1. Input Validation: Ensure all user inputs are properly validated to prevent vulnerabilities like injection attacks.

2. Authentication and Authorization: Implement secure authentication and authorization mechanisms to protect against unauthorized access.

3. Secure Password Storage: Follow best practices for securely storing passwords, such as using hashing solid algorithms and salting.

4. Error Handling and Logging: Implement proper error handling and logging mechanisms to detect and respond to potential security incidents.

5. Secure Session Management: Implement secure session management techniques to prevent session hijacking or fixation attacks.

6. Secure Communication: Use secure communication protocols (e.g., HTTPS) to protect sensitive data during transmission.

7. Secure File Handling: Implement proper file handling techniques to prevent vulnerabilities like directory traversal or file inclusion attacks.

8. Secure Database Access: Use parameterized queries or prepared statements to prevent SQL injection attacks.

9. Code Review and Testing: Conduct regular code reviews and comprehensive testing to identify and fix security vulnerabilities.

10. Secure Configuration Management: Ensure secure configuration of software components, frameworks, and libraries to minimize potential security risks.

CODING STANDARDS

* Principle of Least Privilege: Limit access privileges to the minimum necessary, vulnerabilities ranked as high priority.
* Secure Authentication and Authorization: Implement secure authentication and authorization mechanisms; vulnerabilities ranked as high priority.
* Memory Protection: Protect memory against vulnerabilities like buffer overflows, vulnerabilities ranked as high priority.
* Data Type: Check and validate data types to prevent type-related vulnerabilities, vulnerabilities ranked as medium priority.
* Data Value: Check and validate data values to prevent input validation vulnerabilities, vulnerabilities ranked as medium priority.
* String Correctness: Check and validate strings to prevent string-related vulnerabilities, vulnerabilities ranked as medium priority.
* Assertions: Use assertions to check for expected conditions vulnerabilities ranked as low priority.
* Exceptions: Use exceptions to handle unexpected conditions and vulnerabilities ranked as low priority.
* DRY (Don't Repeat Yourself): Keep code concise and modular, vulnerabilities ranked as low priority.
* SQL Injection: Specifically address SQL injection vulnerabilities, vulnerabilities ranked as a high priority.

ENCRYPTION POLICIES

At REST:

* Ensures data protection when stored on physical or digital storage devices.
* Prevents unauthorized access to sensitive information in case of loss, theft, or improper access.
* Applied to sensitive data stored in databases, file systems, cloud storage, or any other storage medium.
* At-rest encryption adds an extra layer of security to mitigate the risk of unauthorized access.

In FLIGHT:

* Ensures data protection during transmission between systems or over networks.
* Applied when sensitive data is transmitted over networks (web applications, emails, file transfers).
* In-flight encryption safeguards data from unauthorized access during transit, reducing the risk of breaches.

In USE:

* Protects sensitive data while being processed or accessed by authorized users or applications.
* Prevents viewing plaintext data even if an attacker gains system access.
* Applied when processing or accessing sensitive data by applications, databases, or users.
* In-use encryption provides additional protection, reducing the risk of unauthorized access or leakage.

TRIPLE-A POLICIES

**Authentication:**

Users must authenticate themselves using unique credentials or multi-factor authentication. This helps verify user identity before granting access to the system. This will be applied to all systems and applications requiring user authentication to prevent unauthorized access and protect sensitive information.

**Authorization**:

Users are granted appropriate access privileges based on roles and responsibilities. Access controls enforce the least privilege and need-to-know principles. This controls user actions and resource access within the system and will be applied to systems and applications requiring user authorization, preventing unauthorized actions and protecting sensitive information.

**Audit Logging:**

System activities are logged and audited regularly, including user actions, access attempts, and security events. Audit logs are protected from tampering and retained for a specified period, enabling monitoring, incident investigation, and compliance purposes. This will be applied to systems and applications requiring auditing capabilities\ aiding in the detection of incidents, identifying anomalies, and tracking user actions.

UNIT TESTING

* Here, we see an example of the acceptable means of validating an email vs an invalid means.
* Next, we display how secure code can be used to provide numeric input within a specified range.

AUTOMATION FLOW

Here, we have a diagram of the Standard Automation Flow. Next, we will discuss the security tools that reside in each phase of the automation flow.

TOOLS

1. Access and Plan: Define project scope, identify stakeholders, and establish goals. Use project management tools like JIRA or Trello.

2. Design: Create a high-level design for the software, including architecture and user interface. Use design tools like Sketch or Figma.

3. Build: Write and test the code for the software. Use development tools like Git, Jenkins, or Visual Studio Code.

4. Verify and Test: Test the software to ensure quality and security. Use testing tools like Selenium or Appium.

5. Transition and Health Check: Deploy the software to production and perform a health check. Use deployment tools like Ansible or Kubernetes.

6. Monitor and Detect: Monitor the software in production to detect issues or vulnerabilities. Use monitoring tools like Nagios or Prometheus.

7. Respond: Respond to issues or vulnerabilities detected during monitoring. Use incident response tools like PagerDuty or Opsgenie.

8. Maintain and Stabilize: Maintain and stabilize the software over time. Use maintenance tools like Chef or Puppet.

External tools automate and streamline processes throughout the DevSecOps pipeline, helping with security vulnerability identification, code testing, software deployment, system monitoring, incident response, and maintenance. Integrating these tools enables faster and more secure software development while reducing the risk of security breaches or downtime.

RISKS AND BENEFITS

A standardized security policy is essential for mitigating security threats and reducing vulnerability. Organizations are at a higher risk of security breaches without a clear and consistent policy.

Implementing a security policy brings benefits such as improved security posture and reduced risk of breaches. Guidelines, procedures, and best practices help protect systems, networks, and sensitive data from unauthorized access or malicious activities.

RECOMMENDATIONS

The existing security policy is comprehensive but has potential gaps that must be addressed. These gaps include securing third-party libraries or APIs, secure password storage and handling, and incomplete coverage of secure coding.

To address these gaps, the following standards should be adopted:

1. OWASP Top 10: Provides guidance on critical web application security risks.

2. NIST Cybersecurity Framework: Offers guidelines and best practices for managing cybersecurity risk.

3. CIS Controls: Provides a prioritized set of actions to improve cybersecurity posture.

Organizations can enhance their security measures and mitigate potential vulnerabilities by adopting these standards.

CONCLUSIONS

Maintaining a strong security posture and mitigating the risk of security breaches requires adopting standardized security policies and best practices. The benefits and risks of not taking action outweigh the initial costs and effort involved.

Standardized security policies help achieve compliance with regulations and industry standards, which are crucial for industries handling sensitive data. Compliance avoids penalties and demonstrates a commitment to customer privacy and trust.

Standardized policies improve incident response capabilities, enabling swift and effective responses to security incidents. Transparent reporting, containment, investigation, and recovery procedures minimize impact and downtime.

Adopting standardized security policies is essential for organizations to maintain a strong security posture, reduce the risk of breaches, and safeguard reputation, customer trust, and long-term success.

Pennsylvania Government. (n.d.). *Cyber Threat Advisory Level*. Office of Administration. [Cyber Threat Advisory Level](https://www.oa.pa.gov/Programs/Information%20Technology/cybersecurity/Pages/Cyber-Threat-Advisory-Level.aspx)

Michali. (2022, September 15). *What is a DevSecOps pipeline?*. Check Point Software. [What is a DevSecOps Pipeline? - Check Point Software](https://www.checkpoint.com/cyber-hub/cloud-security/devsecops/what-is-a-devsecops-pipeline/)

Ahmed, M. (2023, November 4). *Know why defense in depth is useful & effective*. TekWeck. [Know Why Defense in Depth is Useful & Effective](https://tekweck.com/defense-in-depth/)

Schrader, D. (n.d.-a). *Top 12 types of data security solutions for protecting your sensitive information*. 12 Types of Data Security Solutions to Keep Your Sensitive Information Safe. [Top 12 Types of Data Security](https://blog.netwrix.com/2019/09/12/top-12-data-security-solutions-to-protect-your-sensitive-information/)

Timothy A. Chickand Joe Yankel. (2022, June 13). *Modeling DevSecOps to Protect The Pipeline*. SEI Blog. [Modeling DevSecOps to Protect the Pipeline](https://insights.sei.cmu.edu/blog/modeling-devsecops-to-protect-the-pipeline/)