ATT&CK v18 has been released! Check out the blog post or changelog for more information.

Application Developer Guidance

Application Developer Guidance focuses on providing developers with the knowledge, tools, and best practices needed to write secure code, reduce vulnerabilities, and implement secure design principles. By integrating security throughout the software development lifecycle (SDLC), this mitigation aims to prevent the introduction of exploitable weaknesses in applications, systems, and APIs. This mitigation can be implemented through the following measures:

Preventing SQL Injection (Secure Coding Practice):

- Implementation: Train developers to use parameterized queries or prepared statements instead of directly embedding user input into SQL queries.
- Use Case: A web application accepts user input to search a database. By sanitizing and validating user inputs, developers can prevent attackers from injecting malicious SQL commands.

Cross-Site Scripting (XSS) Mitigation:

- Implementation: Require developers to implement output encoding for all user-generated content displayed on a web page.
- Use Case: An e-commerce site allows users to leave product reviews. Properly encoding and escaping user inputs prevents malicious scripts from being executed in other users' browsers.

Secure API Design:

- Implementation: Train developers to authenticate all API endpoints and avoid exposing sensitive information in API responses.
- Use Case: A mobile banking application uses APIs for account management. By enforcing token-based authentication for every API call, developers reduce the risk of unauthorized access.

Static Code Analysis in the Build Pipeline:

- Implementation: Incorporate tools into CI/CD pipelines to automatically scan for vulnerabilities during the build process.
- Use Case: A fintech company integrates static analysis tools to detect hardcoded credentials in their source code before deployment.

Threat Modeling in the Design Phase:

- Implementation: Use frameworks like STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) to assess threats during application design.
- Use Case: Before launching a customer portal, a SaaS company identifies potential abuse cases, such as session hijacking, and designs mitigations like secure session management.

Tools for Implementation:

- Static Code Analysis Tools: Use tools that can scan for known vulnerabilities in source code.
- Dynamic Application Security Testing (DAST): Use tools like Burp Suite or OWASP ZAP to simulate runtime attacks and identify vulnerabilities.
- Secure Frameworks: Recommend secure-by-default frameworks (e.g., Django for Python, Spring Security for Java) that enforce security best practices.

ID: M1013
Version: 1.2
Created: 25 October 2017
Last Modified: 10 December 2024

Version Permalink

ATT&CK® Navigator Layers

Techniques Addressed by Mitigation

Domain	ID		Name	Use
Enterprise	T1212		Exploitation for Credential Access	Application developers should consider taking measures to validate authentication requests by enabling one-time passwords, providing timestamps or sequence numbers for messages sent, using digital signatures, and/or using random session keys.[1][2]
Enterprise	T1564		Hide Artifacts	Application developers should consider limiting the requirements for custom or otherwise difficult to manage file/folder exclusions. Where possible, install applications to trusted system folder paths that are already protected by restricted file and directory permissions.
		.009	Resource Forking	Configure applications to use the application bundle structure which leverages the `/Resources` folder location.[3]
		.012	File/Path Exclusions	Application developers should consider limiting the requirements for custom or otherwise difficult to manage file/folder exclusions. Where possible, install applications to trusted system folder paths that are already protected by restricted file and directory permissions.
Enterprise	T1574		Hijack Execution Flow	When possible, include hash values in manifest files to help prevent side-loading of malicious libraries.[4]
		.001	DLL	When possible, include hash values in manifest files to help prevent side-loading of malicious libraries.
Enterprise	T1559		Inter-Process Communication	Enable the Hardened Runtime capability when developing applications. Do not include the `com.apple.security.get-task-allow` entitlement with the value set to any variation of true.
		.003	XPC Services	Enable the Hardened Runtime capability when developing applications. Do not include the `com.apple.security.get-task-allow` entitlement with the value set to any variation of true.
Enterprise	T1647		Plist File Modification	Ensure applications are using Apple's developer guidance which enables hardened runtime.[5]
Enterprise	T1496	.003	Resource Hijacking: SMS Pumping	Consider implementing CAPTCHA protection on forms that send messages via SMS.
Enterprise	T1593		Search Open Websites/Domains	Application developers uploading to public code repositories should be careful to avoid publishing sensitive information such as credentials and API keys.
		.003	Code Repositories	Application developers uploading to public code repositories should be careful to avoid publishing sensitive information such as credentials and API keys.
Enterprise	T1195		Supply Chain Compromise	Application developers should be cautious when selecting third-party libraries to integrate into their application. Additionally, where possible, developers should lock software dependencies to specific versions rather than pulling the latest version on build.[6]
		.001	Compromise Software Dependencies and Development Tools	Application developers should be cautious when selecting third-party libraries to integrate into their application. Additionally, where possible, developers should lock software dependencies to specific versions rather than pulling the latest version on build.[6] GitHub Actions may be pinned to a specific commit hash rather than a tag or branch.[7]

Domain	ID		Name	Use
Enterprise	T1550		Use Alternate Authentication Material	Consider implementing token binding strategies, such as Azure AD token protection or OAuth Proof of Possession, that cryptographically bind a token to a secret. This may prevent the token from being used without knowledge of the secret or possession of the device the token is tied to.[8][9]
		.001	Application Access Token	Consider implementing token binding strategies, such as Azure AD token protection or OAuth Proof of Possession, that cryptographically bind a token to a secret. This may prevent the token from being used without knowledge of the secret or possession of the device the token is tied to.[8][9]
Enterprise	T1078		Valid Accounts	Ensure that applications do not store sensitive data or credentials insecurely. (e.g. plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage).
Mobile	T1626		Abuse Elevation Control Mechanism	Applications very rarely require administrator permission. Developers should be cautioned against using this higher degree of access to avoid being flagged as a potentially malicious application.
Mobile	T1517		Access Notifications	Application developers could be encouraged to avoid placing sensitive data in notification text.
Mobile	T1513		Screen Capture	Application developers can apply the `FLAG_SECURE` property to sensitive screens within their apps to make it more difficult for the screen contents to be captured.[10]
Mobile	T1635		Steal Application Access Token	Developers should use Android App Links[11] and iOS Universal Links[12] to provide a secure binding between URIs and applications, preventing malicious applications from intercepting redirections. Additionally, for OAuth use cases, PKCE[13] should be used to prevent use of stolen authorization codes.
		.001	URI Hijacking	Developers should use Android App Links[11] and iOS Universal Links[12] to provide a secure binding between URIs and applications, preventing malicious applications from intercepting redirections. Additionally, for OAuth use cases, PKCE[13] should be used to prevent use of stolen authorization codes.
Mobile	T1474		Supply Chain Compromise	Application developers should be cautious when selecting third-party libraries to integrate into their application.
		.001	Compromise Software Dependencies and Development Tools	Application developers should be cautious when selecting third-party libraries to integrate into their application.

References

1. Justin Schamotta. (2022, October 28). What is a replay attack?. Retrieved September 27, 2023.
2. Bugcrowd. (n.d.). Replay Attack. Retrieved September 27, 2023.
3. Apple Inc. (2021, February 18). App security overview. Retrieved October 12, 2021.
4. Amanda Steward. (2014). FireEye DLL Side-Loading: A Thorn in the Side of the Anti-Virus Industry. Retrieved March 13, 2020.
5. Apple Inc.. (2021, January 1). Hardened Runtime: Manage security protections and resource access for your macOS apps.. Retrieved March 24, 2021.
6. Daniel Krivelevich and Omer Gil. (n.d.). Top 10 CI/CD Security Risks. Retrieved November 17, 2024.
7. Omer Gilm Aviad Hahami, Asi Greenholts, and Yaron Avital. (2025, March 20). GitHub Actions Supply Chain Attack: A Targeted Attack on Coinbase Expanded to the Widespread tj-actions/changed-files Incident: Threat Assessment . Retrieved May 22, 2025.
8. Microsoft. (2023, October 23). Conditional Access: Token protection (preview). Retrieved January 2, 2024.
9. Venkat Viswanathan. (2023, June 13). A leap forward in token security: Okta adds support for DPoP. Retrieved January 2, 2024.
10. Nightwatch Cybersecurity. (2016, April 13). Research: Securing Android Applications from Screen Capture (FLAG_SECURE). Retrieved November 5, 2019.
11. Google. (n.d.). Verify Android App Links. Retrieved September 11, 2020.
12. Apple. (n.d.). Universal Links for Developers. Retrieved September 11, 2020.
13. N. Sakimura, J. Bradley, and N. Agarwal. (2015, September). IETF RFC 7636: Proof Key for Code Exchange by OAuth Public Clients. Retrieved December 21, 2016.