Decomposing & Measuring Trust on the Software Supply Chain

Trust-Contract Table

Trust Relationship	Attack (Effect)	Associated Threats (Cause)	Associated Trust Contracts
Dependency users & their *dependency developers*	A developer adopts a dependency that contains malicious code	● A malicious package is developed and advertised as a legitimate package [3] ● A malicious name confusion package is created *(Combosquatting, typosquatting, brandjacking, Similarity Attack, Altering Word Order, Manipulating Word Separators, etc.)* [3] ● *Masking legitimate packages* (targeting package name or URL resolution [3] ● *Dangling references* (using resource identifiers of orphaned projects (names or URLs)) [3] ● Dependency developers include a malicious package in their software [2]	*The dependency user (trustor) trusts the dependency developers (trustee) to…*
			…not intentionally include malicious code
			…be honest about their intentions and the functions of their package
	A developer adopts a dependency that contains inadequate security	● Dependency developers include a malicious package in their software [2] ● A developer's dependency becomes abandoned [1] ● A package is created with exploitable vulnerabilities	…employ proper security practices to prevent vulnerabilities
			…properly handle vulnerabilities

			...recognize malicious code or packages and not include them in their software
			...continue maintaining their package
Maintainers & their *co-maintainers*	A malicious maintainer is added to a project	● *Contribute as Maintainer* (obtaining contributor privileges towards the actual codebase) [3, 1] ● *Taking over Legit Accounts* (stealing account credentials) [3] ● *Compromising the maintainer system* (exploiting vulnerabilities, adding malicious components to the maintainer systems) [3] ● *Tamper the build job as a maintainer* (becoming a maintainer and tampering with code) [3, 1] ● Malicious code is added during a code refactor ● *Running a malicious build job* (tampering with system resources) [3]	*Fellow maintainers (trustors) trust this new maintainer (trustee) to...*
			...not add malicious code
			...not include any malicious dependencies
			...not steal vulnerable information
	An incompetent maintainer is added to a project	● A project is improperly documented ● Code tests are poorly written	...implement proper security practices to prevent vulnerabilities
			...properly document their work
Developers & their	A developer	● *Hypocrite Merge Request* (an	*The current developer*

contributors	accepts a pull request that contains malicious code	attacker, acting as a contributor, turns code malicious) [3, 1] ● A false vulnerability disclosure is reported	*(trustor) trusts the contributor (trustee) to...*
			...not include malicious code in their pull request
			...be honest about their intentions
			...not steal vulnerable information
			...use proper security practices to prevent vulnerabilities
Developers & open-source 'hubs'	A developer adopts a malicious package from an open-source hub	● A malicious package is developed and advertised as a legitimate package [3] ● A malicious name confusion package is created *(Combosquatting, typosquatting, brandjacking, Similarity Attack, Altering Word Order, Manipulating Word Separators, etc.)* [3] ● *Masking legitimate packages* (targeting package name or URL resolution [3]	*The developers (trustors) trust the open-source hubs (trustee) to...*
			...block or delete malicious packages on the website
			...patch all vulnerabilities on the website

References

1) Wermke, Dominik, et al. "Committed to trust: A qualitative study on security & trust in open source software projects." *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

2) Kshetri, Nir, and Jeffrey Voas. "Supply chain trust." *IT Professional* 21.2 (2019): 6-10.

3) Ladisa, Piergiorgio, et al. "Sok: Taxonomy of attacks on open-source software supply chains." *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023.