CERIAS

The Center for Education and Research in Information Assurance and Security

SoK: A Defense-Oriented Evaluation of Software Supply Chain Security

Eman Abu Ishgair†, Marcela S. Melara‡, Santiago Torres-Arias†

† Purdue School of Electrical and computer Engineering , ‡ Intel Labs

SUMMARY

The software supply chain comprises a highly complex set of operations, processes, tools, institutions and human factors involved in creating a piece of software. Several high-profile attacks that exploit a weakness in this complex ecosystem have spurred research in identifying classes of supply chain attacks. Yet, practitioners often lack the necessary information to understand their security posture and implement suitable defenses against these attacks. We argue that the next stage of software supply chain security research and development will benefit greatly from a defense-oriented approach that focuses on holistic bottom-up solutions. We introduces the AStRA model, a framework for representing fundamental software supply chain elements and their causal relationships. Using this model, we identify software supply chain security objectives that are needed to mitigate common attacks and systematize knowledge on recent and well-established security techniques for their ability to meet these objectives.

Core SSC Elements

Actors: Any organization, institution or individual involved with any aspect of the of software supply chain. Actors are often identified via digital credentials such as cryptographic keys.

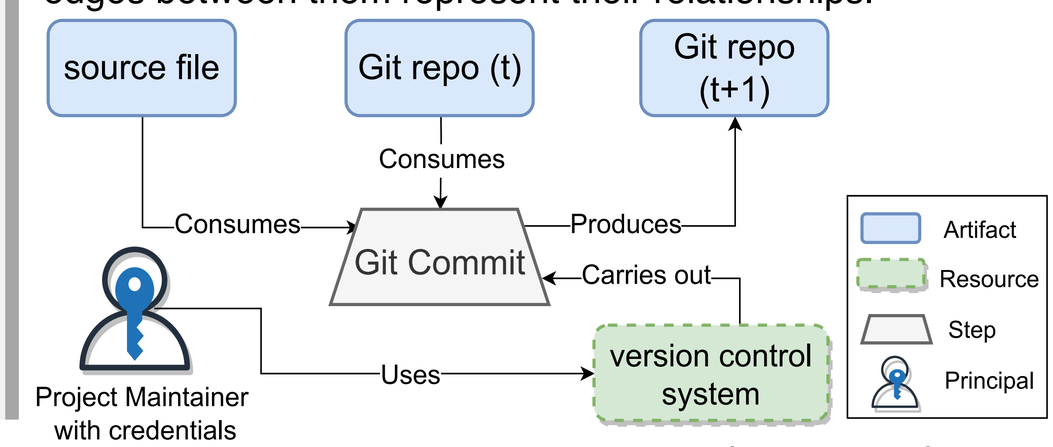
Artifacts: A unit of digital information that is required for the creation, configuration and evaluation of a piece of software. An artifact is typically represented as a file or a collection: package, a Docker image, a source code file.

Resources: Any software component, service or hardware component used in the creation, configuration, or evaluation of an artifact. Examples of resources include Git, CI/CD services such as GitHub Actions, or the OS used to build an artifact.

Steps: A step refers to a specific task or operation that creates, evaluates or distributes a software artifact. In practice, principals use resources to carry out steps.

AStRA Model

To reason about SSC defenses, we introduce AStRA model that represents a specific SSC structure as a DAG in which **A**rtifacts, **St**ep, **R**esources and **A**ctors are vertices and the edges between them represent their relationships.



Methodology

Our systematization leverages the AStRA model and groups defenses by each component of the software supply. For each component, as well as the DAG topology as a whole, we identify various security objectives that are needed to mitigate common attacks and discuss different academic and industry approaches seek to meet these objectives *(Table).*

Evaluation

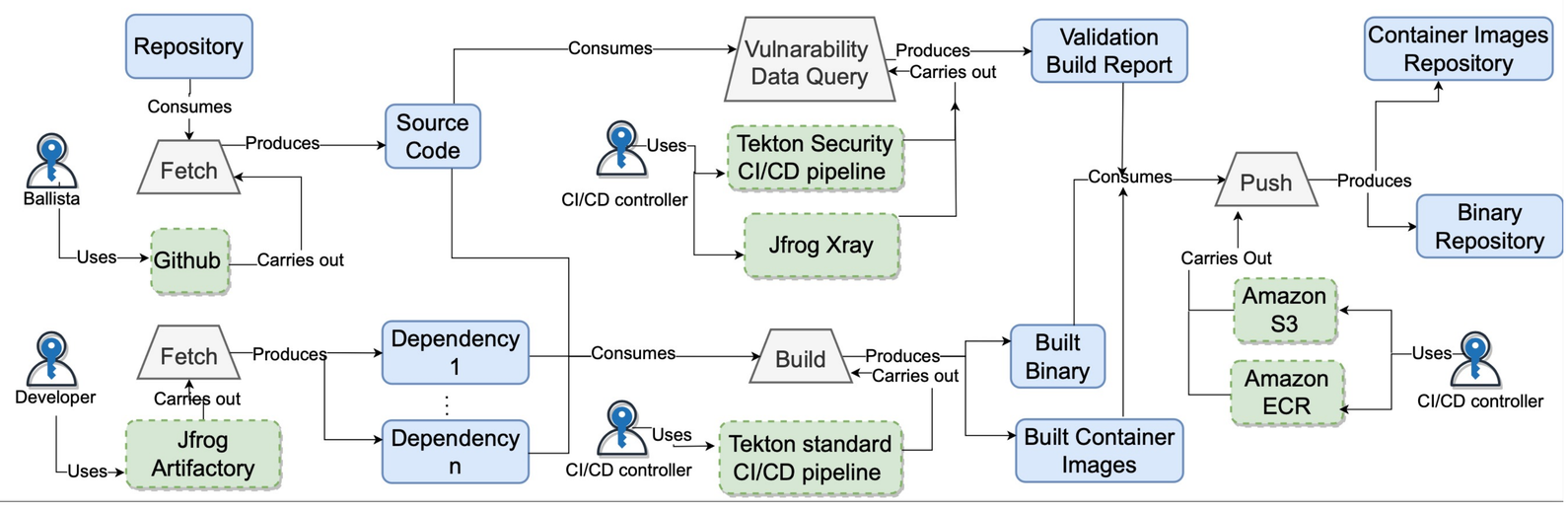
To use AStRA model for software supply chain security assessment, we need to ensure two properties: **Generality** generality as the ability of model to represent any supply chain, and **Completeness** the ability to capture every threat for every supply chain modeled. To evaluate generality, we used our model to represent popular software supply chains that was a target for high profile attacks, figure below represent **SolarWinds SSC.** To evaluate completeness, we perform a comparative analysis of the AStRA model objectives, and 72 attacks listed in the IQT Labs dataset. We explicitly map each desired security objective to mitigated attack classes.

EVALUATION

e 1: Evaluated software supply chain defense approaches mapped to security objectives

Technique	Example	Security Objectives				
Principals		**P1**	**P2**	**P3**	**P4**	**P5**
usage-limited credentials†	OTP [114]	●	-	-	-	-
ephemeral credentials†	Fulcio [88]	●	-	◐	-	-
credential transparency†	CONIKS [91]	-	●	-	-	◐
role separation	GH roles [61]	-	-	●	-	-
multi-factor authentication†	2FA [111]	◐	-	●	-	-
separation of principal privilege	2-P Code Review [39]	-	-	-	●	-
threshold authorization†		-	-	●	●	●
Artifacts		**A1**	**A2**	**A3**		
hashing†	Subresource Integrity [35]	◐	-	-		
artifact signing†	Microsoft Authenticode [127]	◐	-	-		
metadata signing†	The Update Framework [117]	●	◐	-		
artifact transparency†	Sigstore [123]	●	◐	-		
ledger-based systems†	Contour [23]	●	●	-		
static analysis†	Bandit [30]	-	-	◐		
dynamic analysis†	HARP [138]	-	-	●		
Resources		**R1**	**R2**	**R3**		
discretionary access control†	Linux file permissions [140]	◐	-	-		
mandatory access control†	AppArmor [1]	●	-	-		
filtering†	eBPF [49]	◐	●	-		
virtualization†	cgroups [5]	-	◐	●		
Steps		**S1**	**S2**	**S3**		
Formal verification†	Verifiable Compilers [85]	●	-	-		
Deterministic steps†	DetTrace [97]	●	-	-		
Step transparency	SBOMs [126]	-	●	-		
Auth. Step Transparency†	in-toto [7]	-	●	●		
Step Consensus†	CHAINIAC [6]	-	◐	●		
Topology		**T1**	**T2**	**T3**	**T4**	**T5**
supply chain layout integrity†	in-toto layouts [8]	●	-	-	●	
resource replication†	Mirrors [2]	-	●	-	-	-
dependency reduction	Bootstrappable builds [41]	-	-	●	-	-
supply chain reproducibility†	Reproducible builds [81]	-	-	-	●	●

● = objective met; ◐ = implementation-dependant objective-met; - = objective not met; † =has academic publicatio



PURDUE UNIVERSITY

CERIAS