JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Practical Software Supply Chain Assurance

High Confidence Software and Systems Conference 2024

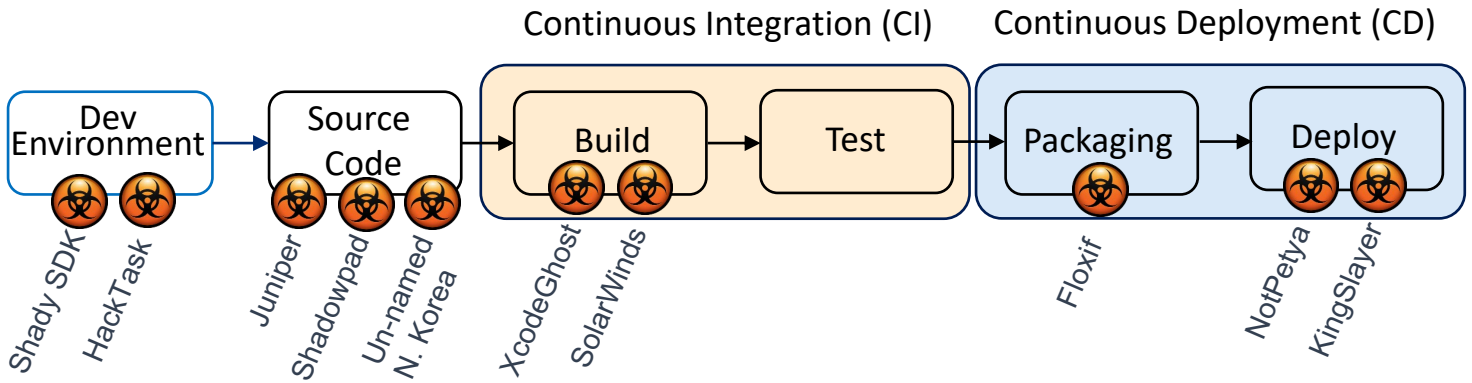
Leo Babun, Ph.D.
Leo.Babun@jhuapl.edu

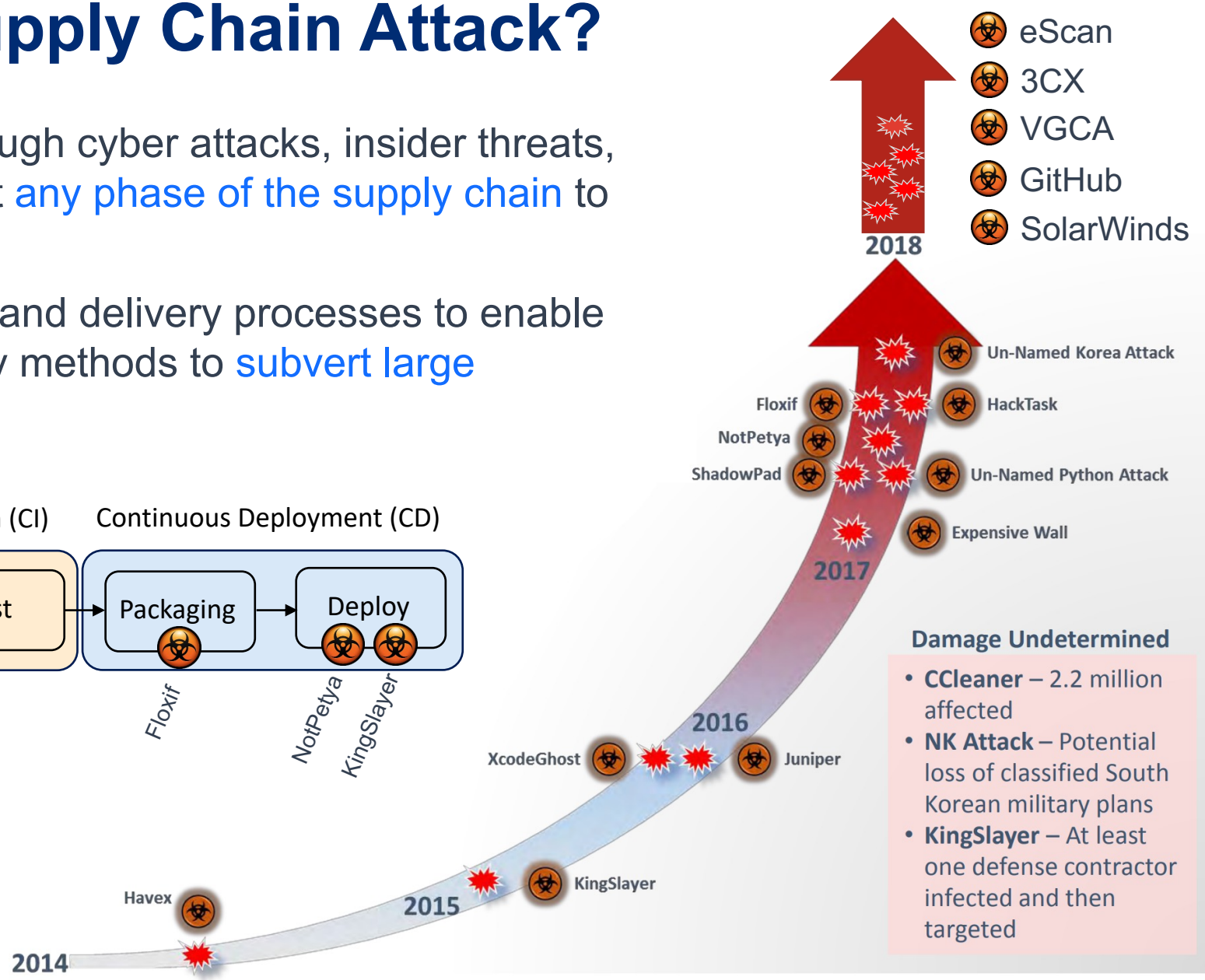
Kathleen McGill, Ph.D.
Kathleen.McGill@jhuapl.edu

May 2024

What is a Software Supply Chain Attack?

- "Compromising software code through cyber attacks, insider threats, and other close access activities at any phase of the supply chain to infect an unsuspecting customer."
- "Hackers … compromise software and delivery processes to enable successful, rewarding, and stealthy methods to subvert large numbers of computers."

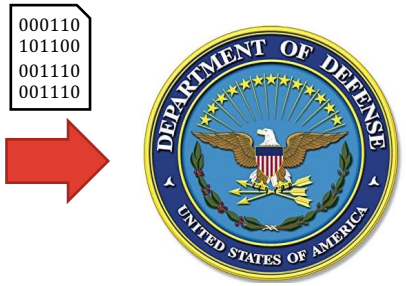


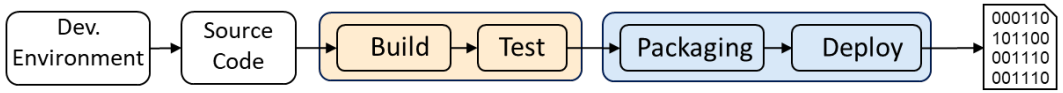


Source: Office of the Director of National Intelligence (ODNI) "Software Supply Chain Attack Graphic"

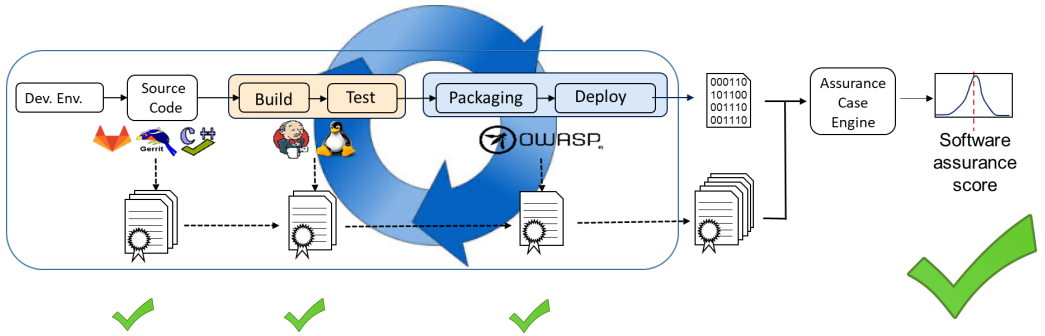
Practical Software Supply Chain Assurance

Abby and team develop software for the DoD.

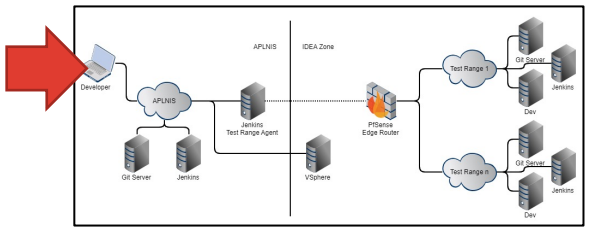


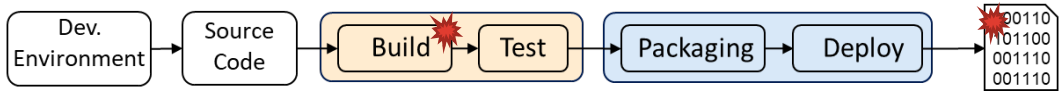


These are all the tools that run when they produce a software delivery.

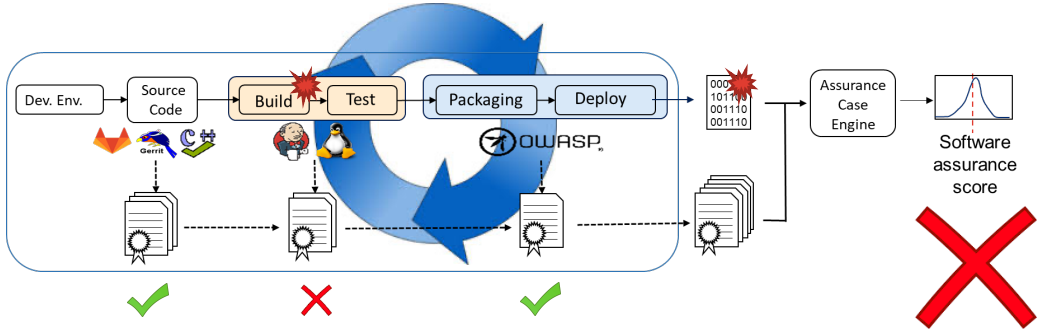


Zoe compromises Abby's host machine and inserts a malicious library into the codebase.



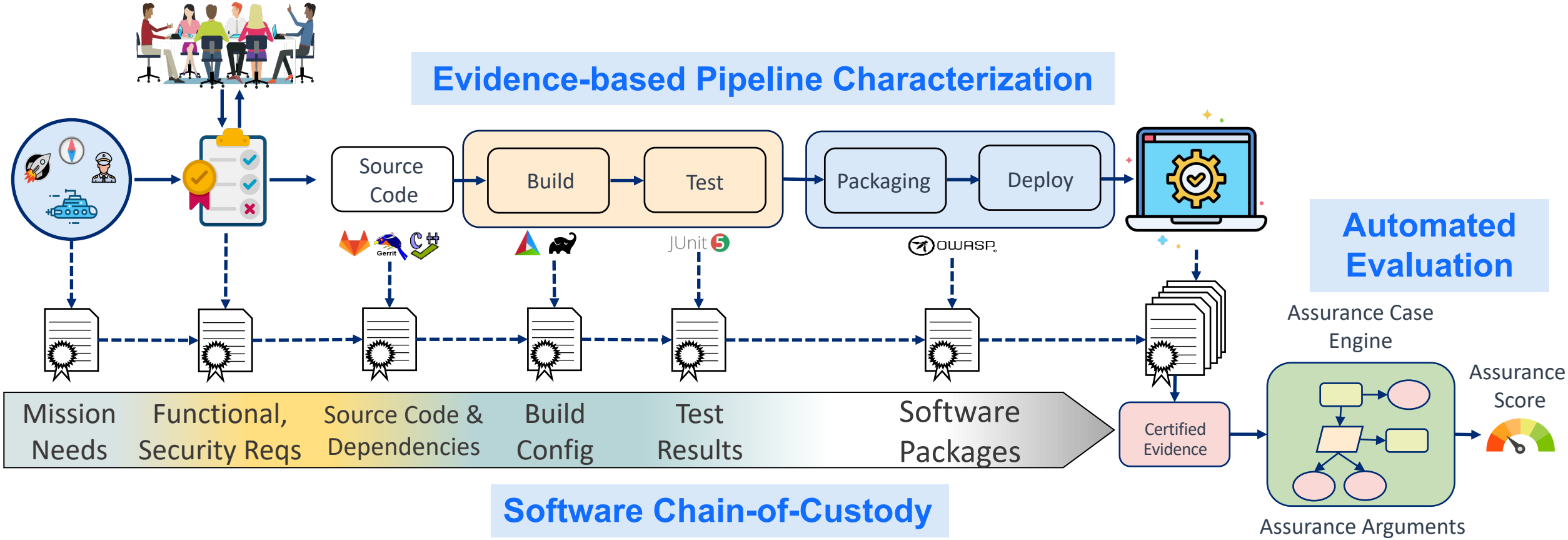


When Abby goes to build new software, the tools sense the compromise!



The CSAADE Methodology

Cryptographically Secure, Automated Development Environment

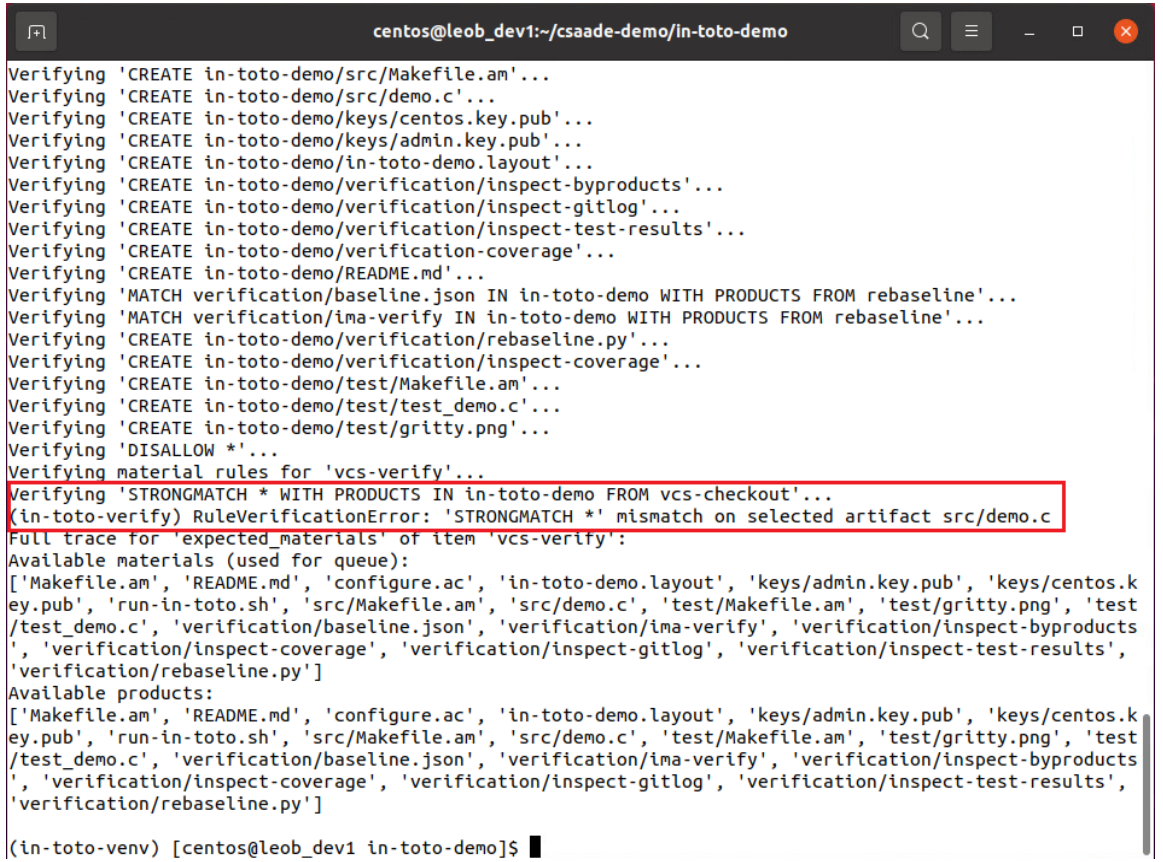


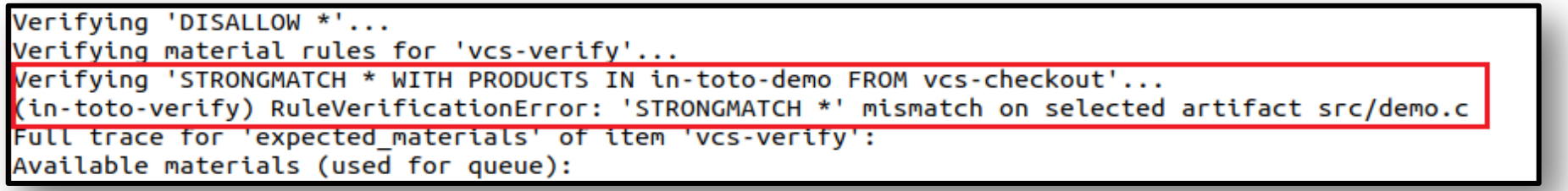
A comprehensive toolchain to generate and evaluate evidence from the software supply chain automatically and establish confidence in software products.

Proof-of-Concept Results

- CSAADE framework detects compromised software!
  - SolarWinds-like attacks detected

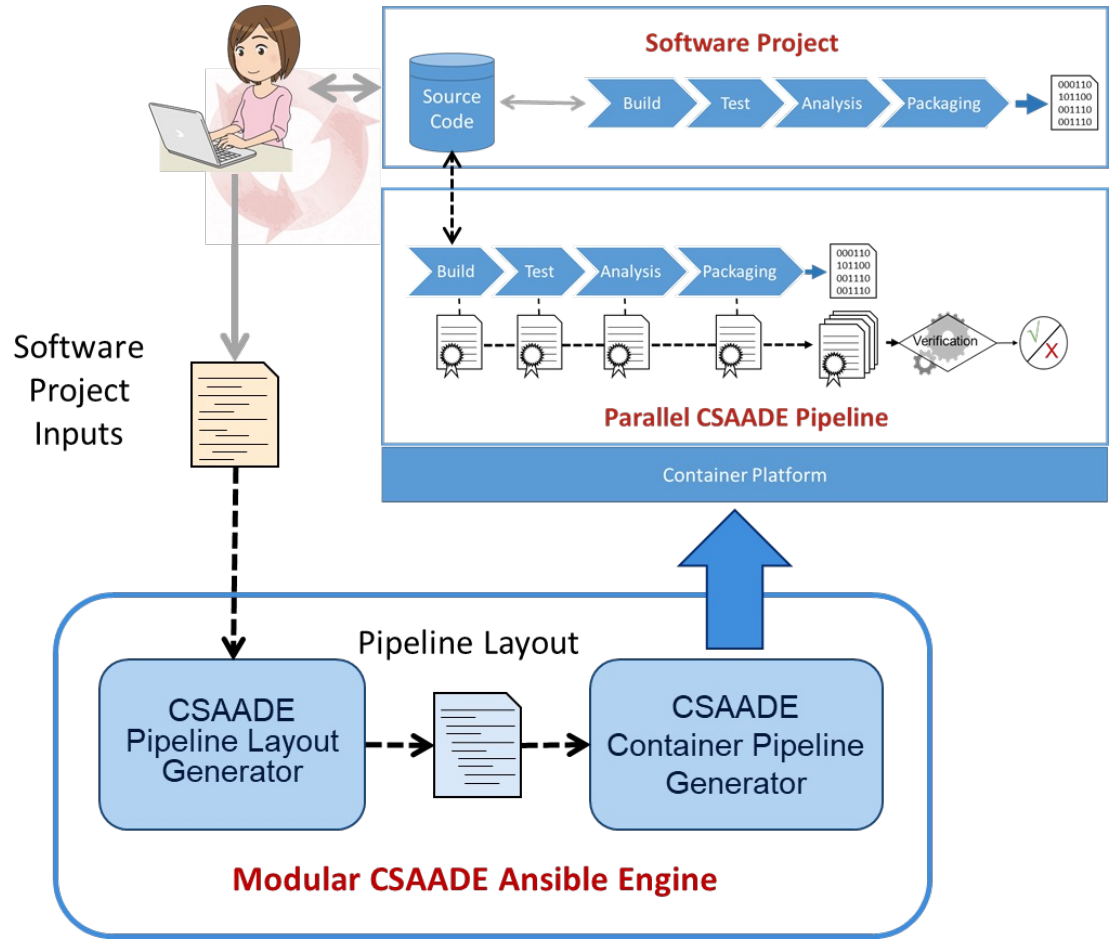
- Practicality issues and developer friction
  - Manual, error prone deployment and configuration
  - Too difficult for software developers to use
  - Hard to adapt to existing projects and legacy pipelines





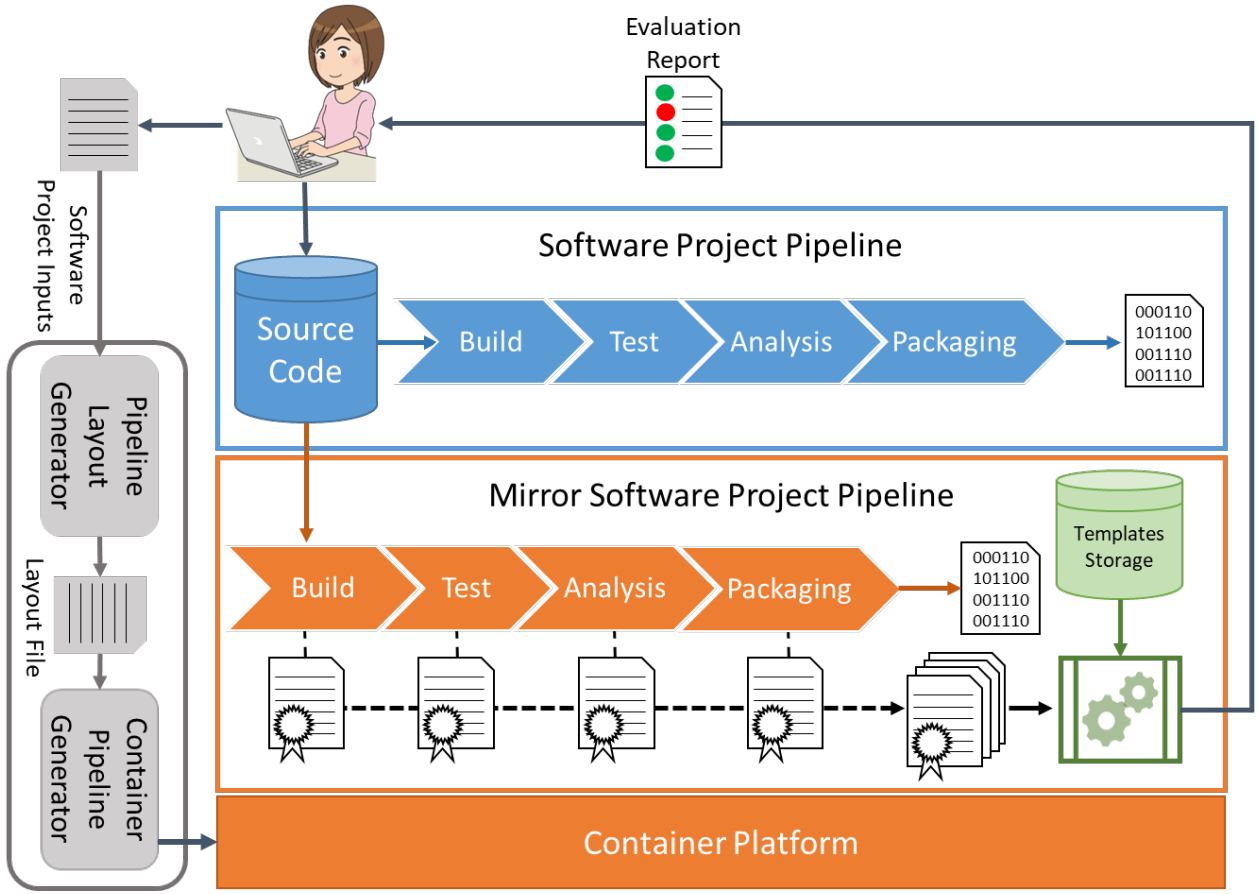
Practical Enhancements to Address Barriers to Adoption

- Automated provisioning and configuration
  - Use of Ansible and containerization for easier CSAADE configuration and deployment
- Mirror existing software development projects
  - Steps for pipeline characterization
- Template-based solution
  - Support for a variety of software development pipelines with minimal burden on developers

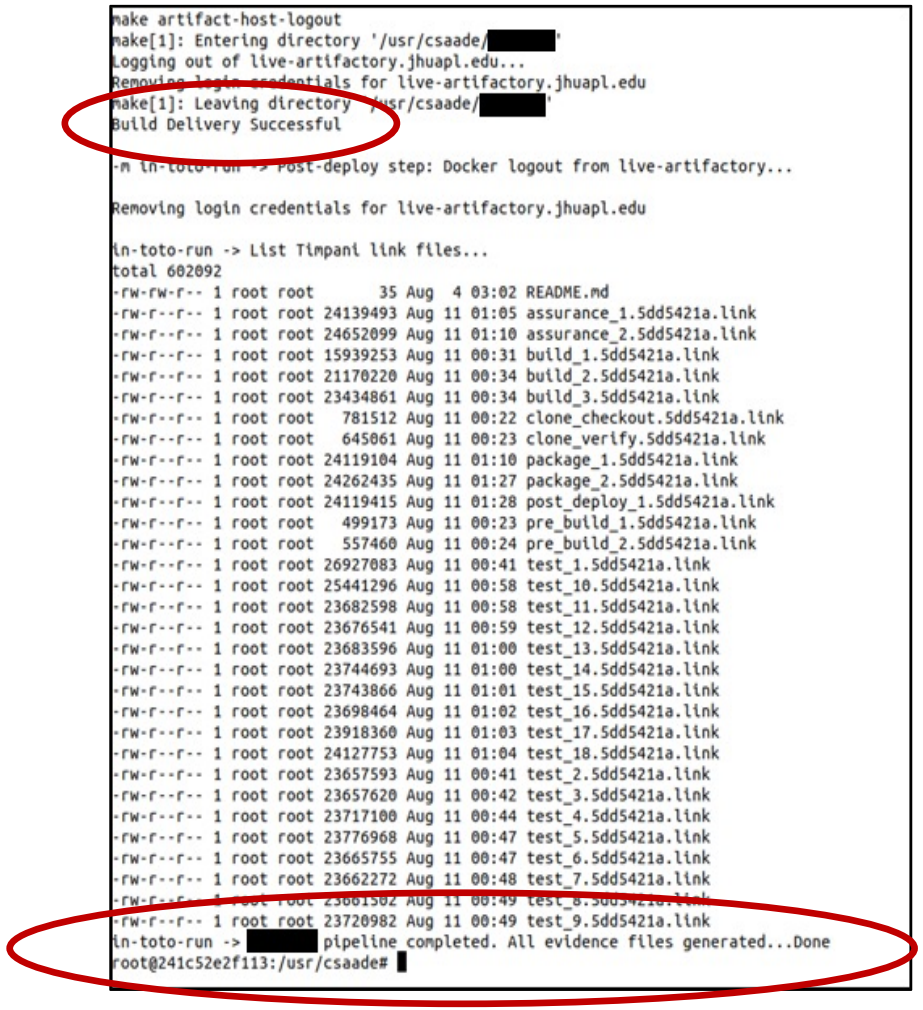


Technical Tasks and Challenges

- Ansible Pipeline *Layout* Generator
  - Predict inputs and outputs of each step
  - Provide basis for software chain-of-custody
  - Engineering Challenge: File explosion presents design decisions to balance assurance and scalability
- Ansible *Pipeline* Generator
  - Automate build of CSAADE mirror of legacy pipeline
  - Automate project access with Ansible Vault
  - Engineering Challenge: Build everything to be project-agnostic



Pilot: Integrate CSAADE with Existing Pipeline



- APL Internal Maven-based Java development project
  - `Uses npm` package manager
  - Docker-based build and testing
  - Handles sensitive credential information
  - Several project and build dependencies
- Ansible engine automates end-to-end process
  - Collects evidence and validates software chain-of-custody
- Software chain-of-custody and supply chain evaluation for a project NOT designed for CSAADE
- Ansible automation and containerized architecture drastically simplify deployment

Pilot Success Metrics

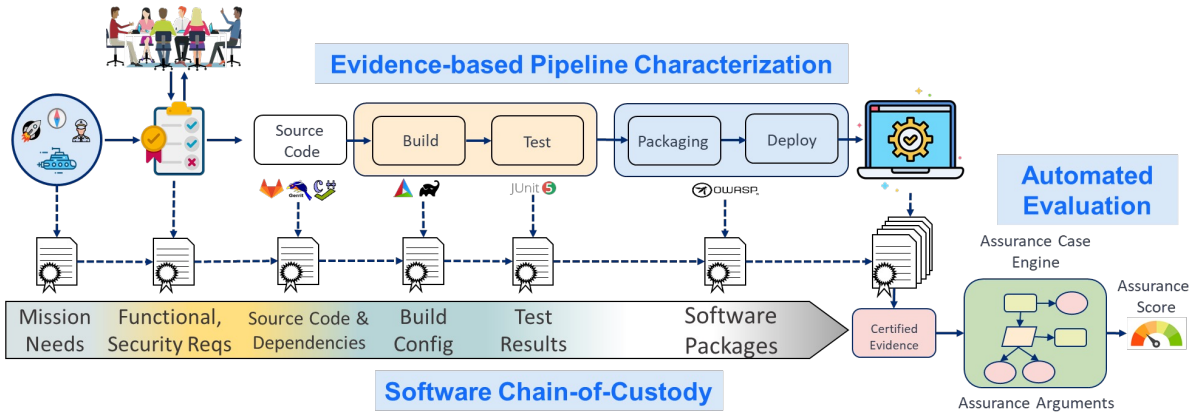
Key Metric	Result(s)
Project Developer Load	• Total commitment: 5 hours • Limited set of inputs required to replicate the entire legacy pipeline
Legacy Pipeline Characterization	• CSAADE configuration file generated in seconds (~22K of lines)
Automated Provisioning	• Total time cut from hours to minutes • Mirror pipeline deployed in minutes

We can deploy a CSAADE pipeline and get practical, adaptive software assurance without derailing primary mission objectives.

Conclusions

- CDAADE uses sensing capabilities to fully characterize the software, how it was produced, and the underlying platform that hosts the development pipeline
- The cryptographic software-chain-of-custody provides the necessary rigor to protect the integrity of the collected evidence and the software supply chain
- CSAADE easily integrates with legacy pipelines and takes the burden off the developers

APL wants to work with the community to advance research and adoption of software supply chain assurance.



JOHNS HOPKINS

APPLIED PHYSICS LABORATORY

What Makes CSAADE Different?

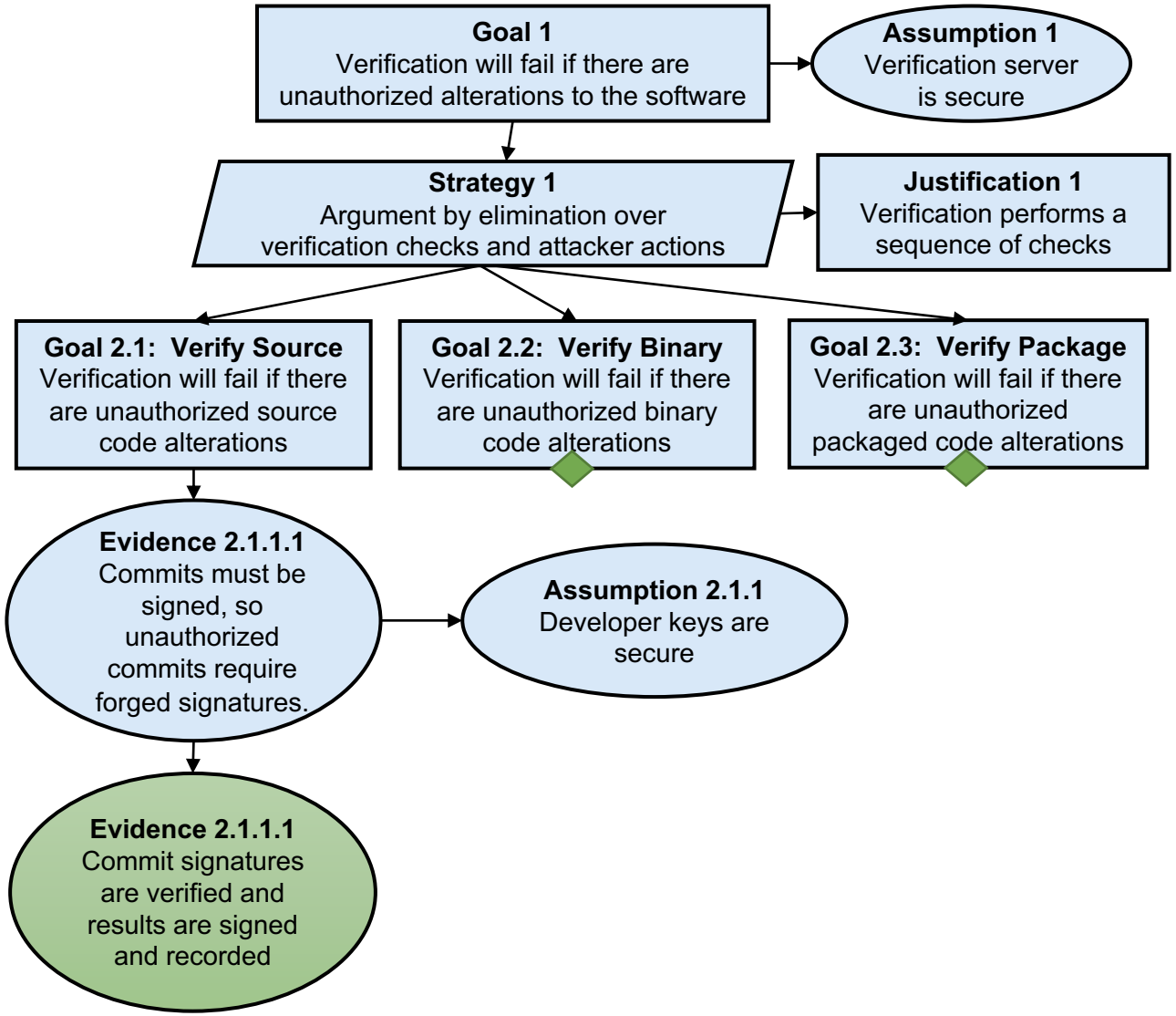
- Sensors span the software development pipeline. Every sensor contributes to the final assurance score
- Platform integrity sensors included

Sensor	Evidence	Phase	Threat Addressed
GitLab	• Commit signatures	Dev	• Malicious source submission by unauthorized actor
CLOC	• Source Lines of Code (SLOC) • Number of files	Dev	• Malicious source submission with stolen credentials
Cppcheck	• List of source code warnings and errors	Dev	• Vulnerable source submission by well-intentioned developer
CodeDNA	• Binary fingerprint • Malware similarity score	Dev, Build	• Malicious source submission with stolen credentials
gcov	• Test source code coverage	Dev	• Vulnerable source code submission by well-intentioned developer
Integrity Measurement Architecture (IMA)	• Hashes of critical files • Hashes of booted software	Dev, Build, Test, Package	• Dev, Build, Test, or Package environment compromise
Linux Kernel Integrity Measurer (LKIM)	• Linux Kernel structure and data values	Dev, Build, Test, Package	• Dev, Build, Test, or Package environment compromise
Tracer	• Trace of syscalls triggered by the build process	Dev, Build, Test, Package	• Dev, Build, Test, or Package environment compromise
OWASP Dependency Check	• List of known dependency vulnerabilities	Dev, Package	• Known vulnerable dependencies

*Sensors partially or fully integrated in prototype are highlighted in blue.

What Makes CSAADE Different?

- Assurance case: a logic tree with a top-level claim decomposed into supporting claims
- Software Supply Chain Assurance Case
  - Decompose by software pipeline stages
  - Threat model informs risks
  - Claims (or assumptions) about source code integrity, code characteristics, and development environment
  - Lowest level claims supported by evidence
- Assurance arguments are expected to change over time based on specific sensors used and known vulnerabilities.
  - Automated, template-based assurance case generation adds flexibility and prevents from having fixed arguments
- ACCELERATE computation engine processes software supply chain assurance case to provide a software assurance score.



Future Work

- Enterprise integration with key management
- Integrate additional sensors to collect evidence supporting different threat models and software programming languages
- Explore AI analysis to provide security recommendations
- Security architecture improvements