

Practical Software Supply Chain Assurance

High Confidence Software and Systems Conference 2024

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

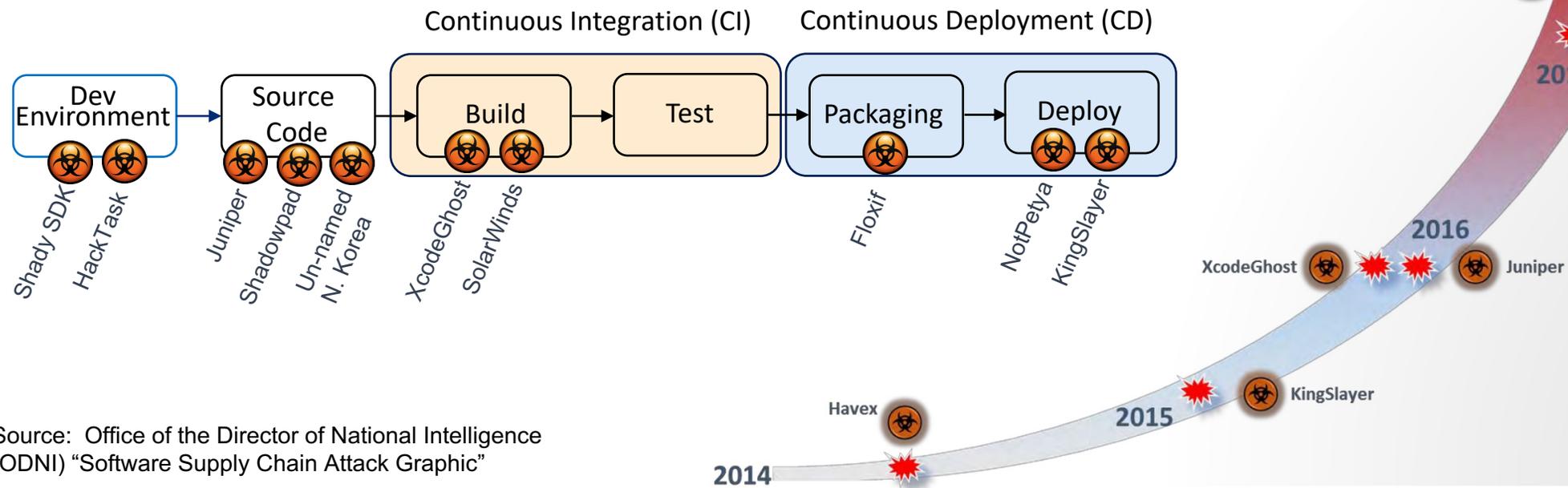
Kathleen McGill, Ph.D.
Kathleen.McGill@jhuapl.edu

May 2024

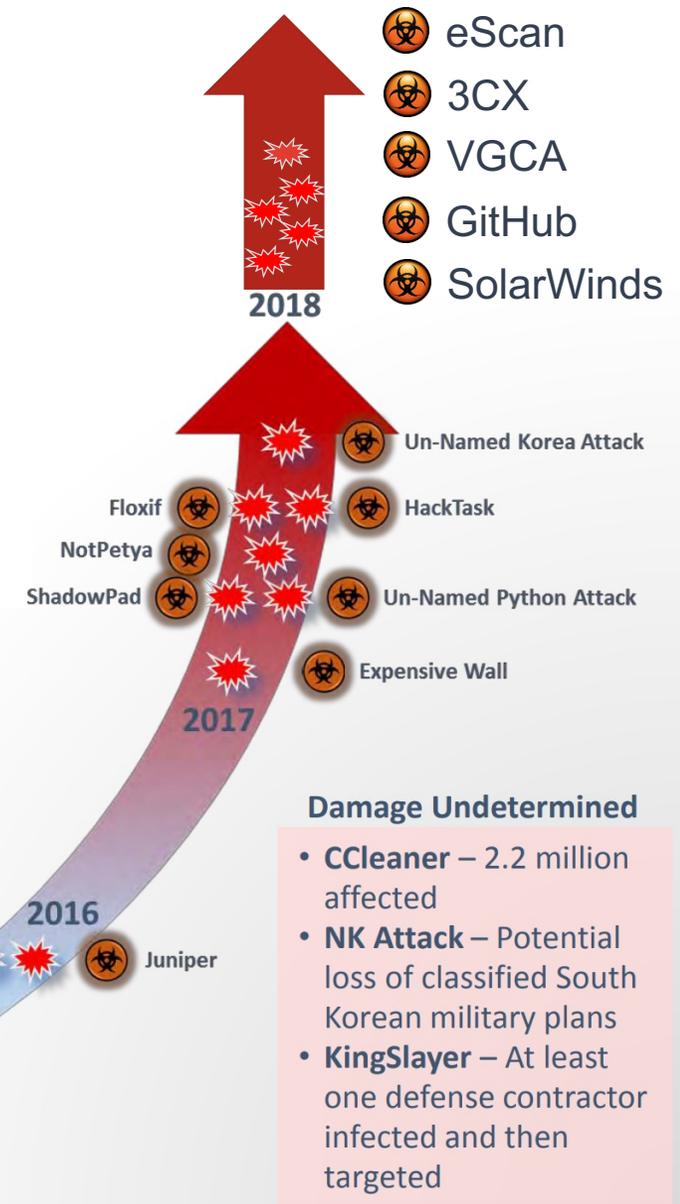
Approved for Public Release

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”

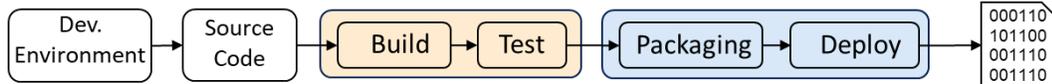
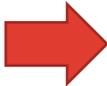


Practical Software Supply Chain Assurance

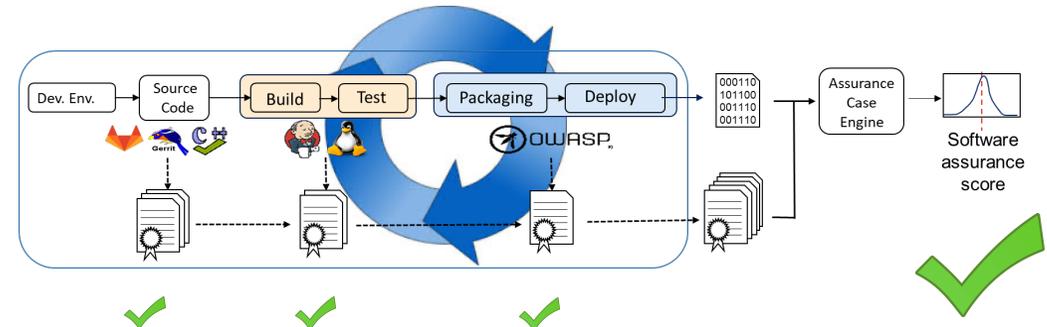
Abby and team develop software for the DoD.



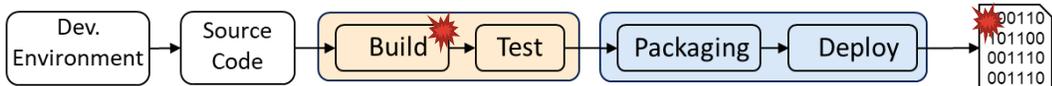
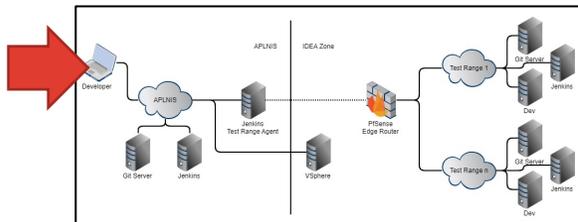
000110
101100
001110
001110



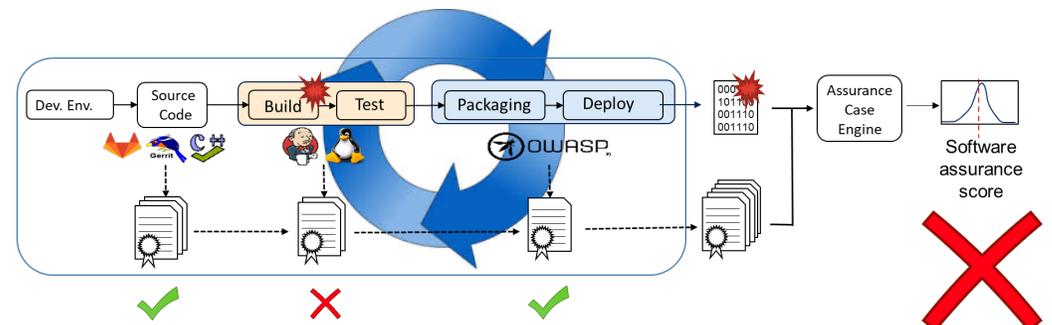
These are all the tools that run when they produce a software delivery.



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!

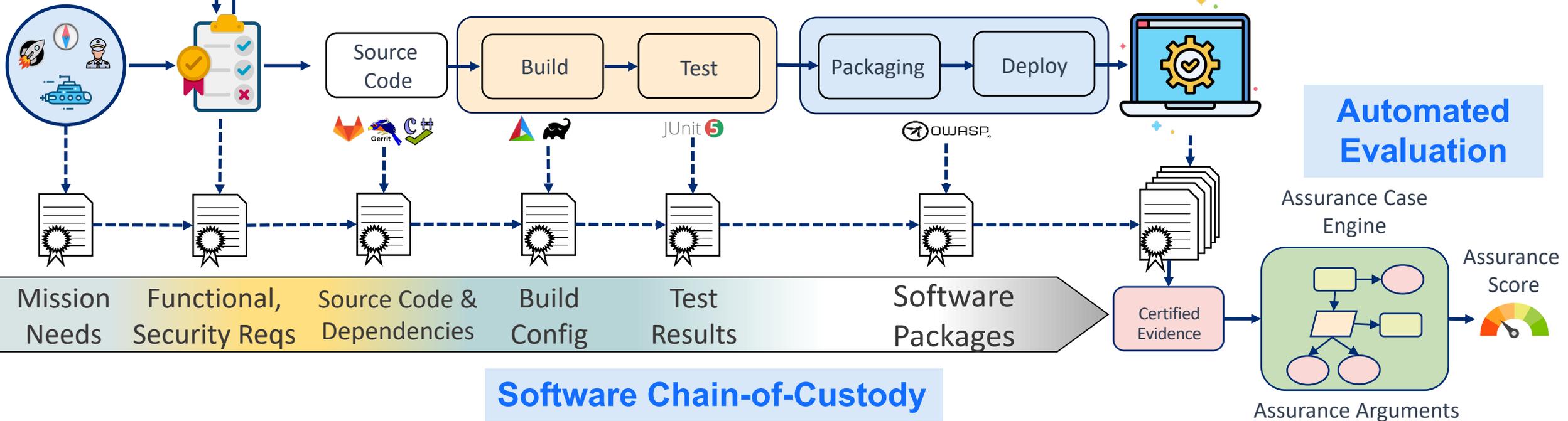


The CSAADE Methodology

Cryptographically Secure, Automated Development Environment



Evidence-based Pipeline Characterization



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

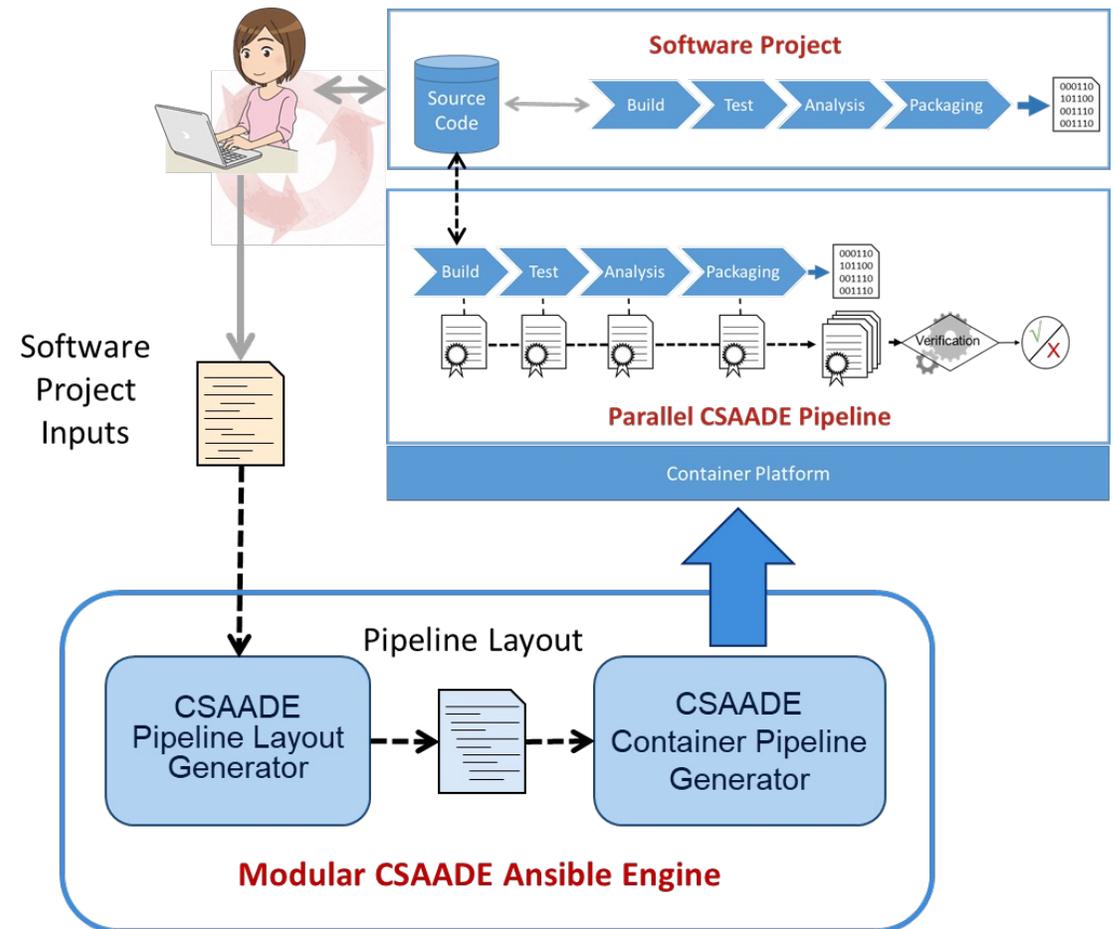
```
centos@leob_dev1:~/csaa-de-demo/in-toto-demo
Verifying 'CREATE in-toto-demo/src/Makefile.am'...
Verifying 'CREATE in-toto-demo/src/demo.c'...
Verifying 'CREATE in-toto-demo/keys/centos.key.pub'...
Verifying 'CREATE in-toto-demo/keys/admin.key.pub'...
Verifying 'CREATE in-toto-demo/in-toto-demo.layout'...
Verifying 'CREATE in-toto-demo/verification/inspect-byproducts'...
Verifying 'CREATE in-toto-demo/verification/inspect-gitlog'...
Verifying 'CREATE in-toto-demo/verification/inspect-test-results'...
Verifying 'CREATE in-toto-demo/verification-coverage'...
Verifying 'CREATE in-toto-demo/README.md'...
Verifying 'MATCH verification/baseline.json IN in-toto-demo WITH PRODUCTS FROM rebaseline'...
Verifying 'MATCH verification/ima-verify IN in-toto-demo WITH PRODUCTS FROM rebaseline'...
Verifying 'CREATE in-toto-demo/verification/rebaseline.py'...
Verifying 'CREATE in-toto-demo/verification/inspect-coverage'...
Verifying 'CREATE in-toto-demo/test/Makefile.am'...
Verifying 'CREATE in-toto-demo/test/test_demo.c'...
Verifying 'CREATE in-toto-demo/test/gritty.png'...
Verifying 'DISALLOW *'...
Verifying material rules for 'vcs-verify'...
Verifying 'STRONGMATCH * WITH PRODUCTS IN in-toto-demo FROM vcs-checkout'...
(in-toto-verify) RuleVerificationError: 'STRONGMATCH *' mismatch on selected artifact src/demo.c
Full trace for 'expected_materials' of item 'vcs-verify':
Available materials (used for queue):
['Makefile.am', 'README.md', 'configure.ac', 'in-toto-demo.layout', 'keys/admin.key.pub', 'keys/centos.k
ey.pub', 'run-in-toto.sh', 'src/Makefile.am', 'src/demo.c', 'test/Makefile.am', 'test/gritty.png', 'test
/test_demo.c', 'verification/baseline.json', 'verification/ima-verify', 'verification/inspect-byproducts
', 'verification/inspect-coverage', 'verification/inspect-gitlog', 'verification/inspect-test-results',
'verification/rebaseline.py']
Available products:
['Makefile.am', 'README.md', 'configure.ac', 'in-toto-demo.layout', 'keys/admin.key.pub', 'keys/centos.k
ey.pub', 'run-in-toto.sh', 'src/Makefile.am', 'src/demo.c', 'test/Makefile.am', 'test/gritty.png', 'test
/test_demo.c', 'verification/baseline.json', 'verification/ima-verify', 'verification/inspect-byproducts
', 'verification/inspect-coverage', 'verification/inspect-gitlog', 'verification/inspect-test-results',
'verification/rebaseline.py']
(in-toto-venv) [centos@leob_dev1 in-toto-demo]$
```



```
Verifying 'DISALLOW *'...
Verifying material rules for 'vcs-verify'...
Verifying 'STRONGMATCH * WITH PRODUCTS IN in-toto-demo FROM vcs-checkout'...
(in-toto-verify) RuleVerificationError: 'STRONGMATCH *' mismatch on selected artifact src/demo.c
Full trace for 'expected_materials' of item 'vcs-verify':
Available materials (used for queue):
```

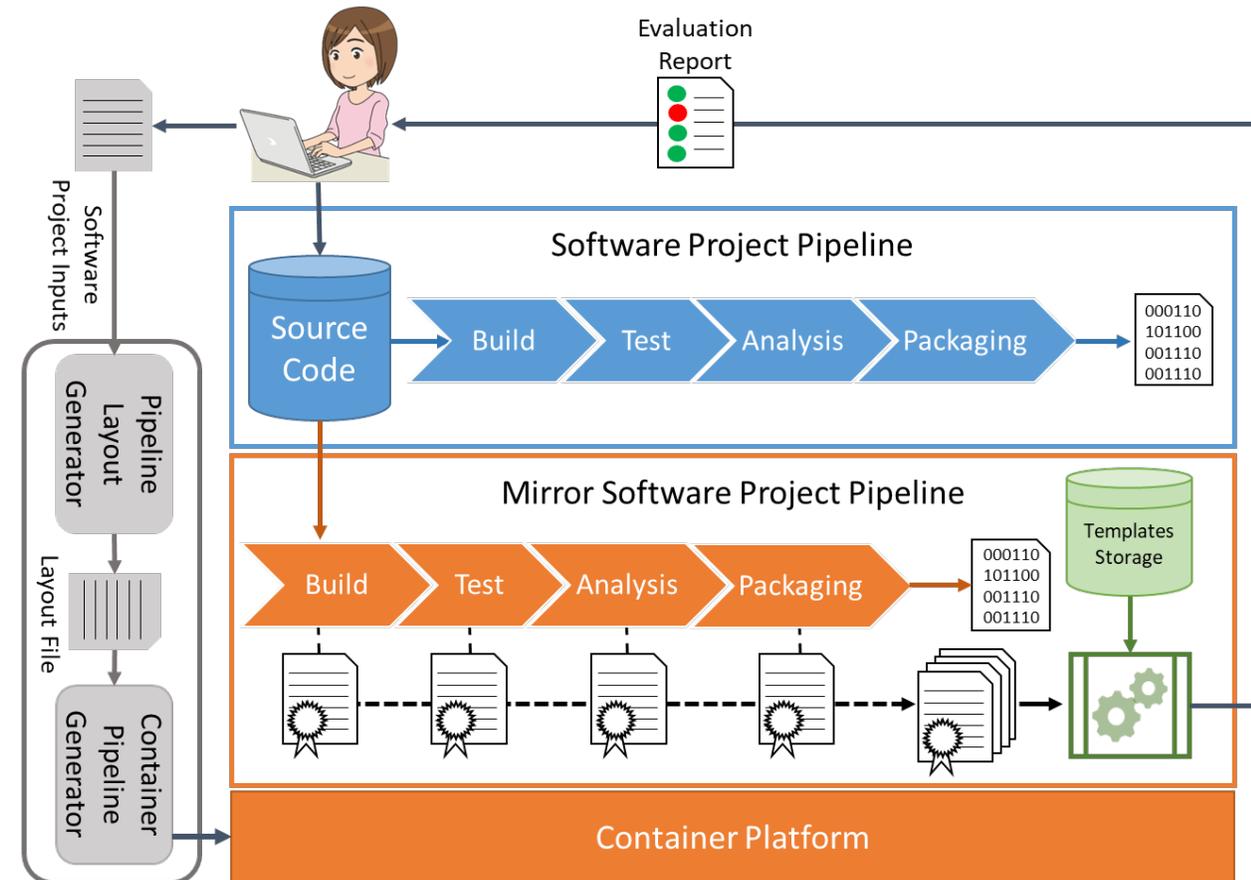
Practical Enhancements to Address Barriers to Adoption

- 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

```
make artifact-host-logout
make[1]: Entering directory '/usr/csaade/███'
Logging out of live-artifactory.jhuapl.edu...
Removing login credentials for live-artifactory.jhuapl.edu
make[1]: Leaving directory '/usr/csaade/███'
Build Delivery Successful

In-toto-run -> Post-deploy step: Docker logout from live-artifactory...

Removing login credentials for live-artifactory.jhuapl.edu

In-toto-run -> List Timpani link files...
total 602092
-rw-r--r-- 1 root root      35 Aug  4 03:02 README.md
-rw-r--r-- 1 root root 24139493 Aug 11 01:05 assurance_1.5dd5421a.link
-rw-r--r-- 1 root root 24652099 Aug 11 01:10 assurance_2.5dd5421a.link
-rw-r--r-- 1 root root 15939253 Aug 11 00:31 build_1.5dd5421a.link
-rw-r--r-- 1 root root 21170220 Aug 11 00:34 build_2.5dd5421a.link
-rw-r--r-- 1 root root 23434861 Aug 11 00:34 build_3.5dd5421a.link
-rw-r--r-- 1 root root   781512 Aug 11 00:22 clone_checkout.5dd5421a.link
-rw-r--r-- 1 root root   645061 Aug 11 00:23 clone_verify.5dd5421a.link
-rw-r--r-- 1 root root 24119104 Aug 11 01:10 package_1.5dd5421a.link
-rw-r--r-- 1 root root 24262435 Aug 11 01:27 package_2.5dd5421a.link
-rw-r--r-- 1 root root 24119415 Aug 11 01:28 post_deploy_1.5dd5421a.link
-rw-r--r-- 1 root root   499173 Aug 11 00:23 pre_build_1.5dd5421a.link
-rw-r--r-- 1 root root   557460 Aug 11 00:24 pre_build_2.5dd5421a.link
-rw-r--r-- 1 root root 26927083 Aug 11 00:41 test_1.5dd5421a.link
-rw-r--r-- 1 root root 25441296 Aug 11 00:58 test_10.5dd5421a.link
-rw-r--r-- 1 root root 23682598 Aug 11 00:58 test_11.5dd5421a.link
-rw-r--r-- 1 root root 23676541 Aug 11 00:59 test_12.5dd5421a.link
-rw-r--r-- 1 root root 23683596 Aug 11 01:00 test_13.5dd5421a.link
-rw-r--r-- 1 root root 23744693 Aug 11 01:00 test_14.5dd5421a.link
-rw-r--r-- 1 root root 23743866 Aug 11 01:01 test_15.5dd5421a.link
-rw-r--r-- 1 root root 23698464 Aug 11 01:02 test_16.5dd5421a.link
-rw-r--r-- 1 root root 23918360 Aug 11 01:03 test_17.5dd5421a.link
-rw-r--r-- 1 root root 24127753 Aug 11 01:04 test_18.5dd5421a.link
-rw-r--r-- 1 root root 23657593 Aug 11 00:41 test_2.5dd5421a.link
-rw-r--r-- 1 root root 23657620 Aug 11 00:42 test_3.5dd5421a.link
-rw-r--r-- 1 root root 23717100 Aug 11 00:44 test_4.5dd5421a.link
-rw-r--r-- 1 root root 23776968 Aug 11 00:47 test_5.5dd5421a.link
-rw-r--r-- 1 root root 23665755 Aug 11 00:47 test_6.5dd5421a.link
-rw-r--r-- 1 root root 23662272 Aug 11 00:48 test_7.5dd5421a.link
-rw-r--r-- 1 root root 23661502 Aug 11 00:49 test_8.5dd5421a.link
-rw-r--r-- 1 root root 23720982 Aug 11 00:49 test_9.5dd5421a.link
In-toto-run -> ███ pipeline completed. All evidence files generated...Done
root@241c52e2f113:/usr/csaade#
```

- 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

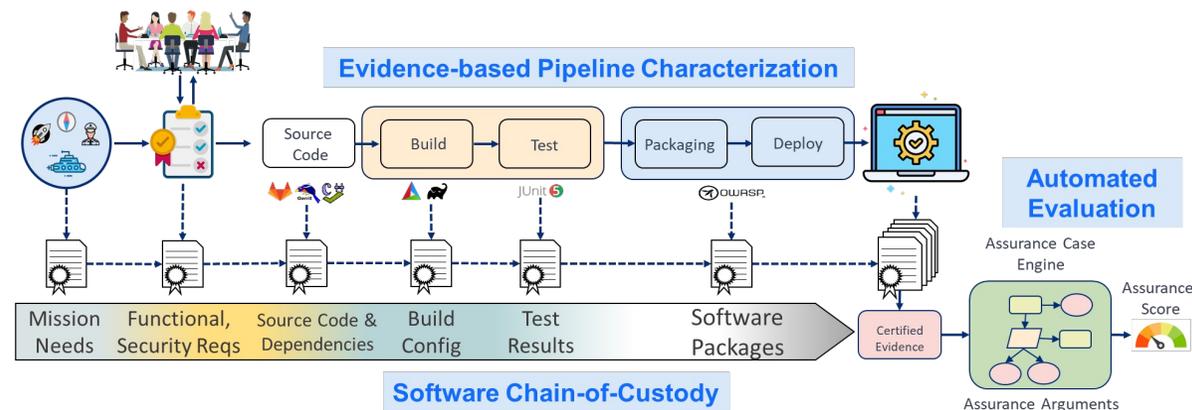
| Key Metric | Result(s) |
|----------------------------------|---|
| Project Developer Load | <ul style="list-style-type: none">Total commitment: 5 hoursLimited set of inputs required to replicate the entire legacy pipeline |
| Legacy Pipeline Characterization | <ul style="list-style-type: none">CSAADE configuration file generated in seconds (~22K of lines) |
| Automated Provisioning | <ul style="list-style-type: none">Total time cut from hours to minutesMirror 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?

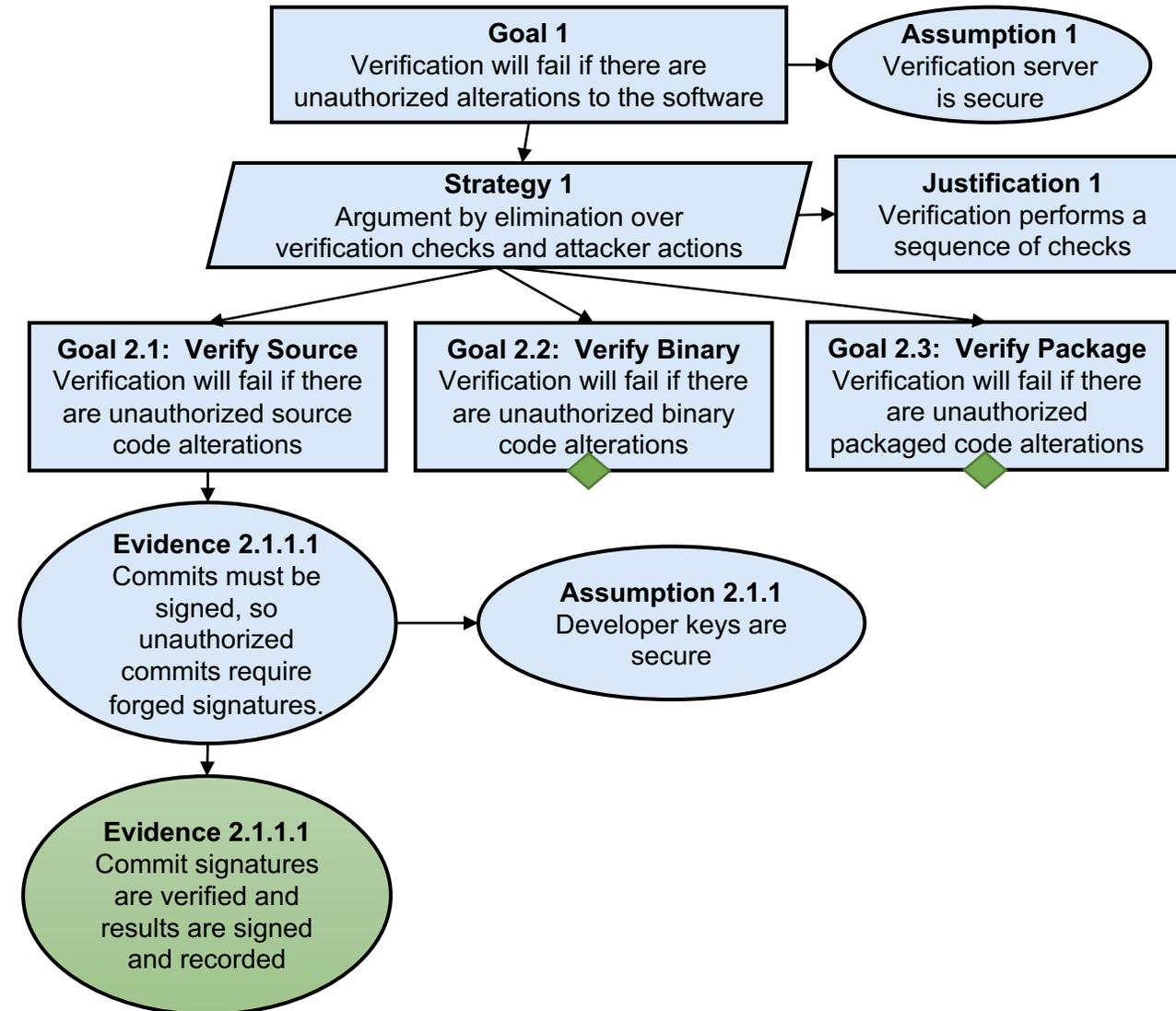
- Sensors span the software development pipeline. **Every sensor contributes to the final assurance score**
- **Platform integrity sensors included**

| Sensor | Evidence | Phase | Threat Addressed |
|---|---|---------------------------|---|
| GitLab | <ul style="list-style-type: none"> • Commit signatures | Dev | <ul style="list-style-type: none"> • Malicious source submission by unauthorized actor |
| CLOC | <ul style="list-style-type: none"> • Source Lines of Code (SLOC) • Number of files | Dev | <ul style="list-style-type: none"> • Malicious source submission with stolen credentials |
| Cppcheck | <ul style="list-style-type: none"> • List of source code warnings and errors | Dev | <ul style="list-style-type: none"> • Vulnerable source submission by well-intentioned developer |
| CodeDNA  | <ul style="list-style-type: none"> • Binary fingerprint • Malware similarity score | Dev, Build | <ul style="list-style-type: none"> • Malicious source submission with stolen credentials |
| gcov | <ul style="list-style-type: none"> • Test source code coverage | Dev | <ul style="list-style-type: none"> • Vulnerable source code submission by well-intentioned developer |
| Integrity Measurement Architecture (IMA) | <ul style="list-style-type: none"> • Hashes of critical files • Hashes of booted software | Dev, Build, Test, Package | <ul style="list-style-type: none"> • Dev, Build, Test, or Package environment compromise |
| Linux Kernel Integrity Measurer (LKIM)  | <ul style="list-style-type: none"> • Linux Kernel structure and data values | Dev, Build, Test, Package | <ul style="list-style-type: none"> • Dev, Build, Test, or Package environment compromise |
| Tracer  | <ul style="list-style-type: none"> • Trace of syscalls triggered by the build process | Dev, Build, Test, Package | <ul style="list-style-type: none"> • Dev, Build, Test, or Package environment compromise |
| OWASP Dependency Check | <ul style="list-style-type: none"> • List of known dependency vulnerabilities | Dev, Package | <ul style="list-style-type: none"> • 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