Software Supply Chain Risks to DevSecOps Programs

Aaron Reffett

Richard Laughlin

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM21-0465

Agenda

- Who are we and what are we talking about?
- What is the software supply chain?
- Recent trends and incidents
- Tail risk: Are these sorts of incidents *really* rare and unpredictable?
- DoD DevSecOps Reference Architecture
- Software supply chain weak points
- What can programs do to protect themselves?

Who are we?

- Aaron Reffett
  - Senior Software Engineer, Security Automation Directorate, CERT
  - Full-Stack Software Engineer
  - 17 years supporting DoD software engineering, 10 years at CERT
  - Directly support DoD programs adopting DevSecOps

- Richard Laughlin
  - Software Engineer, Security Automation Directorate, CERT
  - Research Interests in DevSecOps and Kubernetes.
  - Directly support DoD programs adopting Kubernetes in pursuit of DevSecOps.

DoD DevSecOps is Relatively New

- New model for software engineering and system operations for DoD
  - Still being developed and refined as we speak!
- DevSecOps-based weapons systems have not seen significant use in highly adversarial settings
  - What happens when they face a confluence of adverse events?
- Adoption of open-source software vs COTS/GOTS
- What are the weak points in DevSecOps software supply chain?
- What are the worst-case scenarios if these weak points are exploited?

What is the Software Supply Chain?

- Traditional supply chain focuses on physical rather than logical
- Software Supply Chain is anything and everything that touches or affects your software:
  - Dependencies (configuration, code, binaries, containers, etc.)
  - Build tools (compilers, code analyzers, code repositories, build orchestrators)
  - Operational tools (security, monitoring, logging, alerting, etc.)
  - People, Organizations and Processes (internal and upstream)
  - Underlying platforms and infrastructure (physical or virtual)
- An organization inherits the software supply chain of all components that comprise it's software
- Average GitHub repository contains 203 open source dependencies
- **Enduring, Continuous and Real-time**

5 Trends in Software Supply Chain Attacks

1. *Deep Impact*: State actors target the software supply chain and do so to great effect.
2. *Abusing Trust*: Code signing is a deeply impactful tactic for compromising software supply chains as it can be used to undermine other related security schemes, including program attestation.
3. *Breaking the Chain*: Hijacked updates are a common and impactful means of compromising supply chains and they recurred throughout the decade despite being a well-recognized attack vector.
4. *Poisoning the Well*: Attacks on OSS are popular, but unnervingly simple in many cases.
5. *Downloading Trouble*: App stores represent a poorly addressed source of risk to mobile device users as they remain popular despite years of evidence of security lapses.

Atlantic Council: Breaking Trust: Shades of Crisis Across an Insecure Software Supply Chain

Recent Examples of Supply Chain Incidents

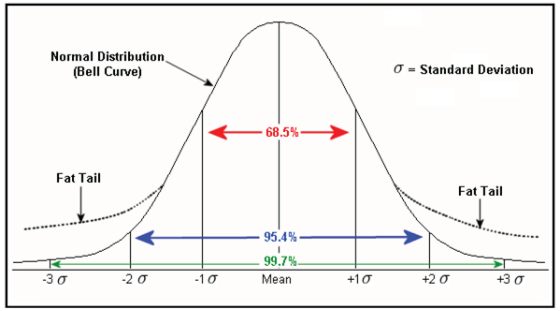
- Various AWS Outages
  - 2017: us-east-1 S3 outage took down EC2 and other services
  - 2019: Power failure (main and backup) in us-east-1 resulted in data loss
  - 2020: Kinesis outage in us-east-1 disrupted service to many apps and services
- AWS Route 53 BGP Hijack
  - Attackers hijacked IP prefixes for AWS DNS service
  - Re-routed traffic destined for a cryptocurrency site to attacker-controlled servers
- SolarWinds
  - Attack against build process
  - Targeting victims' operational infrastructures
- Pulse Secure
  - "Normal" vulnerability
  - Pre-auth RCE (SMB, RDP are also recent examples)

Recent Examples of Supply Chain Incidents (2)

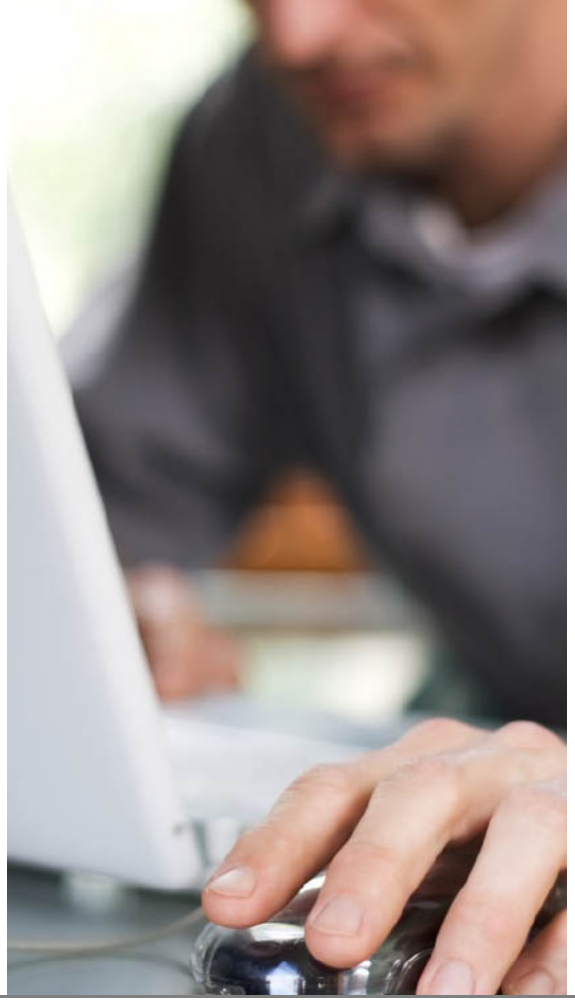
- PyPi package name typo squatting
  - urllib vs urllib3
  - crypt vs crypto
  - setup-tools vs setuptools
- event-stream Node.js library
  - New dev imported new unused dependency: flatmap-stream
  - Malicious code injected into flatmap-stream, which users of event-stream transitively inherited
  - Because of transitive indirection, most users of event-stream did not notice the change
- left-pad Node.js library
  - Developer removed the library from npm (Node Package Manager)
  - Downstream dependents failed to build (e.g. React) causing a snowball effect
  - 22 lines of code
- ShadowHammer
  - Targeted ASUS computers by hijacking ASUS' automated update feature

Tail Risk and the Kurtosis Effect

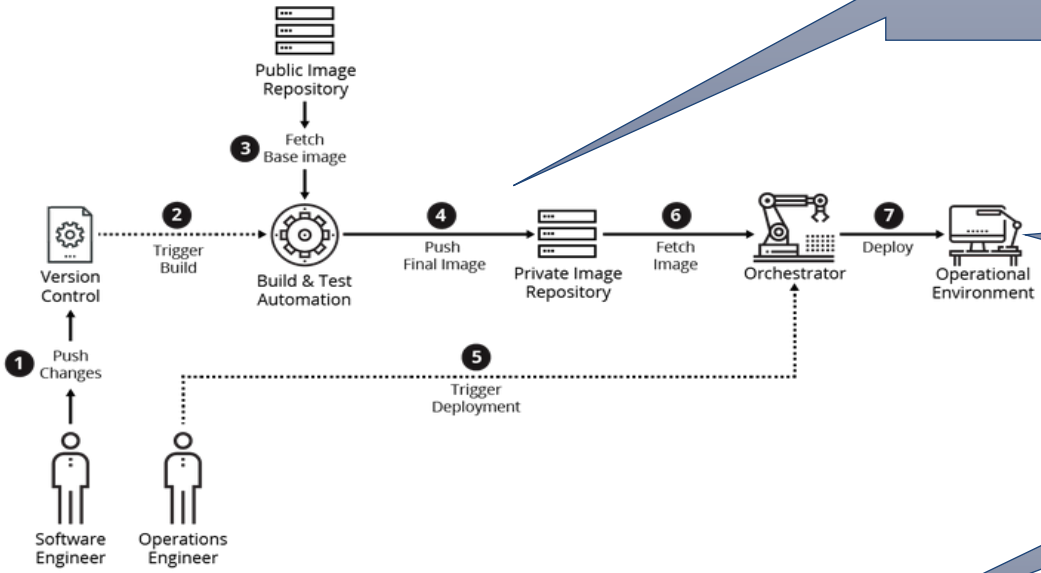
- Many models of risk assume that the probability events follows a normal distribution
- Fat tails occur when the probability of events outside 2 standard deviations of the mean is greater than that of a normal distribution
- Consequence is that low-probability high-impact events are discounted or ignored completely
- **DoD programs should assume that their risks follow a left fat tail distribution**
- **Simple reason: Don't assume black swans are random as opposed to orchestrated**



DoD DevSecOps Reference Architecture



High Level View of Abstract System



3. Software Factory

4. Mission System

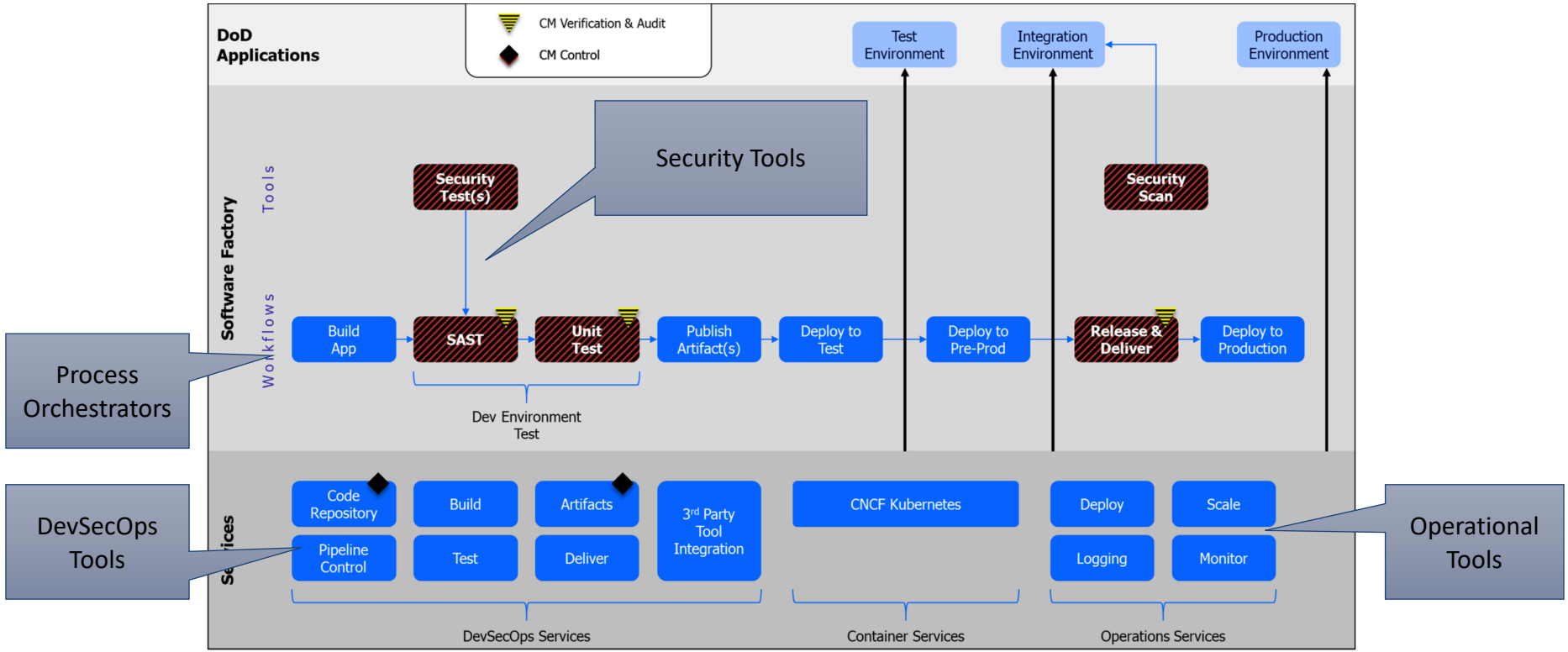
1. Software-enabled infrastructure

2. Common Platform(s)

Common Platform (e.g. Kubernetes)

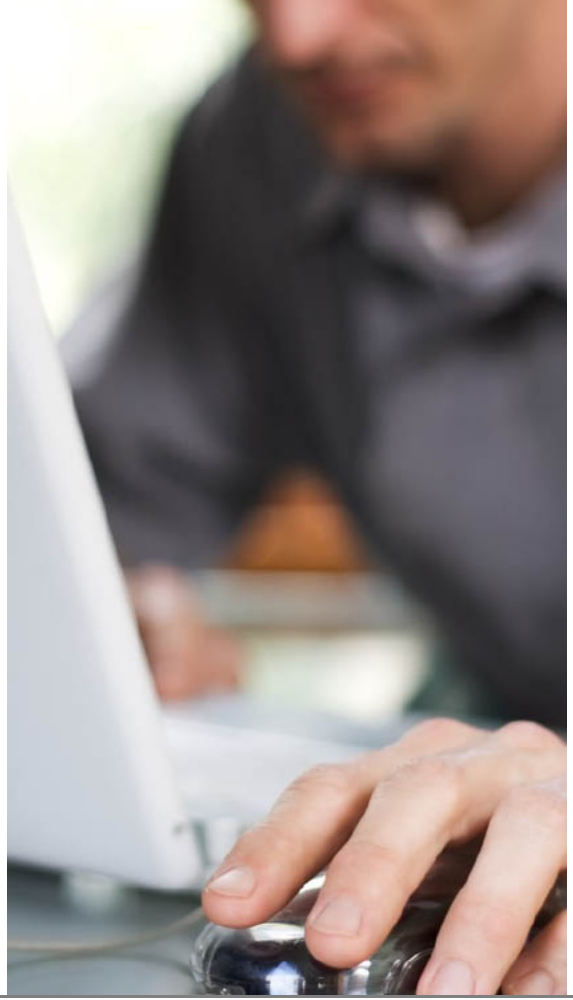
Common Infrastructure (e.g. AWS, Azure, GCP)

DevSecOps Software Factory



Supply Chain Issues in DevSecOps Programs

Software Supply Chain Weak Points



Cloud Infrastructure

- Software layer over someone else's compute, network and storage
- Example: S3 went down and crashed half the Internet
- Threat vectors:
  - Attack hardware integration homogeneity
  - Geo-distributed, but global control plane
  - Attack AMIs, Linux kernel, Xen hypervisor
- What happens if the cloud just goes away?
  - BGP hijacking

General Suggestions for **Cloud Infrastructure**

- Do not assume cloud providers are invulnerable.
  - Consider risks due to compromise or unavailability of cloud infrastructure during your risk analysis.

- For critical applications develop a Disaster Recovery / Business Continuity plan and exercise it regularly.

- Monitoring and Telemetry
  - Utilize cloud monitoring, but assume that during an attack the adversary will attempt to blind operators by attacking monitoring infrastructure as well.

Platform

- Platform One/Kubernetes
  - Homogenous, compromise this and many programs are at risk.
  - Based on open-source, adversaries have full visibility into the components and supply chain.

- Includes Application platforms: OSGi, application servers (e.g. Websphere), application stacks (e.g. Struts, Django)
  - Libraries and containers

- Cloud platforms (S3, EC2, RDS, etc.)
  - Can be reconned in commercial regions to develop attacks against government regions

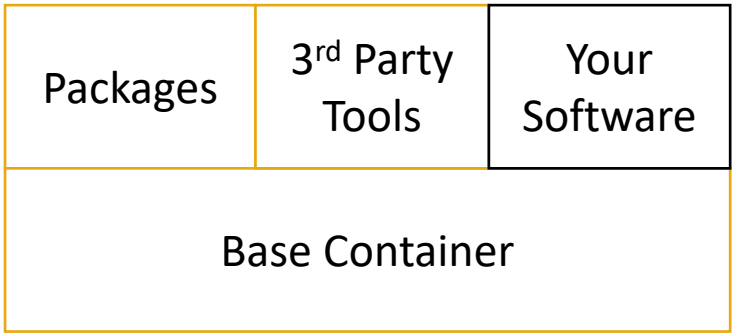
General Suggestions for **Platform**

- Do not assume that vendors or distributions of container orchestrators (e.g. Kubernetes) have got architectural tradeoffs right for your use case.

- Be aware of, and familiar with all components of vendor supplied platforms and remove any components which are not strictly necessary for your environment and use case.

Factory Inputs

- Base container images
- Distro Packages (and dependencies)
- Third-party tools
- Software Libraries/Packages



General Suggestions for **Factory Inputs**

- Manage Base Images
  - Look for signed images.
  - Consolidate on a single package ecosystem (when possible).

- Utilize scanning tools to manage the system packages within containers, and to detect and respond to CVEs.

- Carefully consider each software library which is used by your software.
  - Avoid packages with a small user base (when possible).
  - Watch out for typo-squatting

Mission System

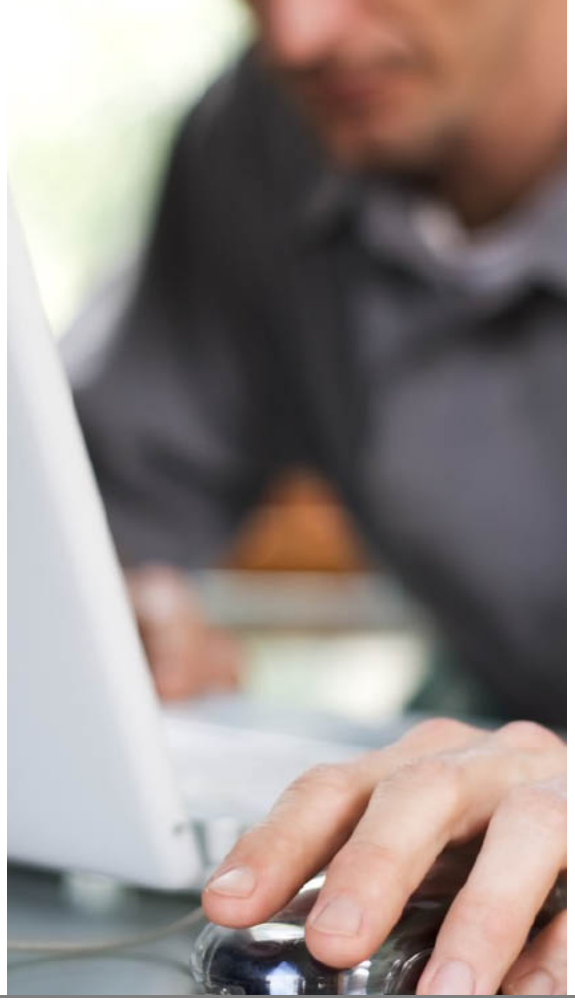
- Mission system utilizes many tools to support its operations
  - What happens if the logging and alerting system is subverted?

- Does the mission system integrate with external applications or services (e.g. via API)?

- SolarWinds: Targeted operational infrastructure
- Pulse Secure: Protects access to critical components, control plane

- How do you know your telemetry is good and accurate?
- Treat tools with ITSM stack as critically as software factory tools or mission application components

Wrapping up



Wrapping Up

- Take inventory of all software, tools, vendors, etc. that make up your supply chain
- Monitoring and Telemetry
- Comprehensive risk analysis
- Control as much of the supply chain as possible
- Be proactive, but be prepared to respond quickly

Supply Chain Issues in DevSecOps Programs

Discussion and Questions