RSA Conference 2018

Peer2Peer

Challenges and Solutions to Secure the DevSecOps Toolchain

P2P3-W05: Challenges and Solutions to Secure the DevSecOps Toolchain

Matthew Hur, Senior Director, Entrust Datacard

It was great to see the level of interest in DevSecOps. We had a full session of 30 active participants and had to turn away folks who wanted to attend. Of the people in the room, we had a fairly even distribution of Dev, IT, and Ops people (with everyone identifying as "security" - go figure, it is the RSA Conference after all). We also had a pretty even distribution of people who stated their stage in DevSecOps as either 1) thinking about it, 2) actively planning, and 3) already doing it.

Over the course of the session, we discussed the following topic areas:

- With respect to building in consistent trust and security through the DevOps process, what are the gaps and issues today?
- How should we build in provability from the development of apps through the deployment, management, and refresh lifecycle of DevOps?
- If you're a developer, how are you managing to establish trust across your applications?
    - Peer to peer trust between nodes in a system (ex. DB, application nodes in AWS)
    - Client apps (e.g. mobile apps) connecting to a service
- With respect to leveraging open source, how do you feel/approach latest open sources updates vs maintaining "stable/tested" open source within your software?

While we started with an acknowledgement of dev toolchains and ops toolchains, the discussion seemed to keep focusing more on the dev toolchain aspect rather than the ops toolchains, and the issue of organizational and role alignment and integration was a recurring theme.

The major discussion areas were 1) writing secure code; 2) DevOps process/inhibiters; 3) consistent trust and policy adherence; and 4) vendor tools and open source.

Key points discussed regarding writing secure code:

- Competency with tools is a limiting factor
- Having well-understood and followed standards across teams is important. At present, is appears that "standardization" seems to be limited to inter-organizational only, with little standardization that can be leveraged more broadly from the industry/market. As part of the standardization discussion, the issue of how to standardize dev tools without encumbering dev innovation was discussed. In other words, if tools/processes are standardized/required, then how can dev still innovate and explore other solutions?
- How can security keep up with development rapid release cycles? How can security NOT slow down dev? How can security policy checks be automated?
- Security expertise within dev teams is a challenge. Folks are embedding security talent within various teams, identifying individuals and teams who have specific knowledge/expertise, and creating well-understood collaborative environments. When embedding security people, it is important that these practitioners have dev experience (e.g. cannot be just policy people), otherwise it becomes problematic to accurately gauge severity or policy deviation without ability to understand the code. It is not clear how this scales in the long run (could this potentially become more "standard" if some of this is replaced with improved tooling?).

Takeaways:

- There is a lot of customization being done in order to assure secure coding practices and adherence to organizational standards.
- There is still room for the industry and state of the art to evolve to provide better standard tooling and best practices that can be applied across organizations more uniformly and with less customization.

Key points discussed regarding DevOps process/inhibiters:

- As organizations are moving from waterfall (or perhaps having to deal with waterfall due to things like highly regulated/controlled environments), they face even greater challenges regarding DevSecOps.
- DevOps speed of release varies
  - Some organizations cannot rapidly release — ex. Federal requirements for authority to operate (ATO) requires an extended schedule for deployment
  - Some organizations release on a frequent basis, but this requires good operating policies and processes - ex. Policies around amount of change (e.g. make "small" changes) and automated test/assurance to support rapid deployment.

Takeaways:

- If you're not agile/CI/CD (continuous integration / continuous deployment), then security/trust management in DevSecOps is even more challenging.
- When addressing continuous deployment, automation is super important, and there is a need to move checks as close as possible to the start of the cycle (e.g. as close to the development environment as possible).

Key points discussed regarding consistent trust and policy adherence:

- It was interesting to note that this topic took us immediately into a containerization discussion. Technologies like docker, rkt, kubernetes, saltstack, helm, ELK, and similar tools/frameworks were brought up.
- One participant mentioned how all communication external to a container is managed via an external config file so that the security team can confirm adherence to policy.

Takeaways:

- Even though there are containerization technologies available that can help, there is still an enormous amount of customization required to build and deploy within an organization.
- There is plenty of room for frameworks and tools to evolve.

Key points discussed regarding vendor tools and open source:

- Vendor tools are not plug and play. The security of vendor tools is questionable, and some organizations are taking to requiring pen testing of vendor tools in order to deploy (and will disqualify vendors if unable to pen test code). Organizations have needed to depend on their own internal security developer expertise and cannot depend on vendor tools.
- We discussed open source as part of the software supply chain, and the question was raised whether folks are comfortable keeping current with new open source updates rather than stay

with tested/proven code. The uncontested answer was that they must keep current (which was slightly surprising—there was not even one more conservative opinion voiced). The people in the room seemed to trust open source more than they trust vendor code. This makes some sense when you consider the ability to change open source code and feed it back to the community (if you care enough about the functionality then you want to fix it, and it's easier to fix open source than to rely on vendors).

Takeaways:

- Vendors need to do a better job of contributing to and integrating tools and frameworks that enable easier development and deployment across the DevSecOps toolchains.
- Open source tools are excellent options to provide organizations with control over their security posture in a DevSecOps environment.

We had a very engaging peer to peer session at RSA Conference 2018. DevSecOps is in a very early stage, and these are the type of discussions that need to continue in order to help shape the direction as the industry moves forward. There is an opportunity and a need to create repeatable patterns and appropriate abstractions to advance the state of the art.