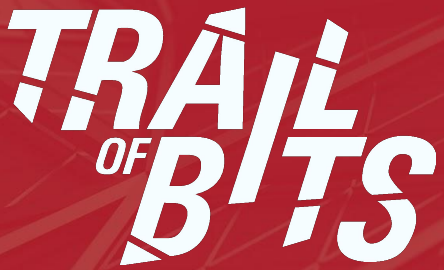


Piloting Around the Rocks: Avoiding Threats in Kubernetes

KubeCon 2019, Robert Tonic & Stefan Edwards

About us

- **Robert Tonic - @b0bbytabl3s**
  - Security Engineer at Trail of Bits
  - Focuses on fintech, blockchain, cloud infrastructure, and distributed systems
  - Background in web application security, infrastructure orchestration, development
- **Stefan Edwards - @lojikil**
  - Assurance Practice Lead at Trail of Bits
  - Focuses on defense, fintech, blockchain, and compiler related projects
  - Background in adversarial simulation, network assessments, web application security

Notable mention

- **Dominik Czarnota**
  - Security Engineer at Trail of Bits
  - Focus on low-level systems & design, distributed systems, and software engineering
  - Background in fintech, application security, and development
  - Unfortunately at another conference right now

Disclaimer

Although we performed this assessment, we do not use Kubernetes every day. As such, we are likely not as proficient with the 3rd-party additions to Kubernetes, such as CNI/CRI providers.

Outline

- Assessment background
- Trust zones
- Architectural concerns
- Reviewing your environment

Assessment background

Goals

- **Find the low hanging fruit**
  - Only new bugs
  - Only in Kubernetes
- **Evaluate the threat model of Kubernetes**
  - Architectural issues
  - Documentation and implementation inconsistencies
- **Identify security-relevant configuration options**
  - Which configuration options have security implications?
  - How hard is it to “mess up”

~~Book writing~~ Results

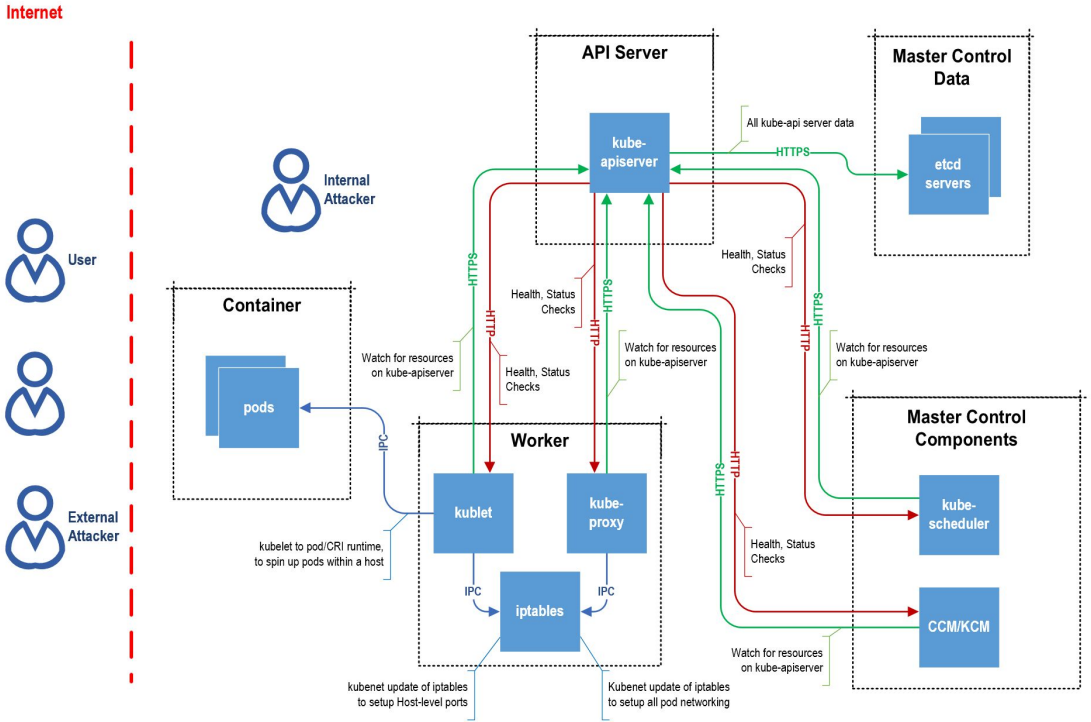
- **Source review: 37 findings**
  - Mostly implementation-related problems
- **Threat model: 17 findings, including a control analysis**
  - Mostly design-related problems
- **White paper: Ergonomics of Kubernetes**
  - Considerations the users should be aware of

Trust zones

Deriving zones from dataflow

- **Kubernetes is composed of many small components**
  - API Server
  - Controllers
  - Schedulers
  - Etc…
- **Tracing communications and interactions allows for quick definitions of trust zones**
  - What components talk to it? What components does it talk to? What protocols?
  - What logic controls inbound and outbound interactions?

High-level Kubernetes data flow



What are trust zones?

- A trust zone is the logical boundary between each component in a system
- Trust zones encompass component location, controls, and policies
- Each zone can be evaluated for criticality
- Nested zones are expected

Evaluating zone criticality

- **What would happen if zone interactions were to stop?**
  - Inbound - Denial of service?
  - Outbound - Cascading failures?
- **What impact would component compromise have?**
  - Credentials?
  - Tenancy?
- **How does placement affect the component?**
  - Should multiple **different components** exist the **same location**?
  - Should multiple instances of the **same component** exist in the **same location**?

Architectural concerns

Policies are applied... Right?

- **TOB-K8S-TM01: Policies may not be applied**
  - Container networking providers are not required to enforce network policies
  - There aren't any warnings when a policy is not enforced
- **TOB-K8S-038: hostPath PersistentVolumes enable PodSecurityPolicy bypass**
  - Persistent volume claims are not restricted by the allowed paths (PodSecurityPolicy), allowing the `hostPath PersistentVolume` to mount arbitrary paths
  - Note: Unless the admission controller for pod security policies are enabled, it won't apply! There is no warning when a policy isn't applied!

How about TLS?

- **TOB-K8S-TM03: Most components accept inbound HTTP**
  - There is no strict enforcement of using HTTPS
- **TOB-K8S-TM02: Insecure TLS by default**
  - The components which do support TLS often use insecure settings
- **TOB-K8S-028: Kubernetes does not support certificate revocation**
  - If a certificate is compromised, it can be extremely hard to rotate.

Are secrets safe?

- **TOB-K8S-TM08: Secrets not encrypted at rest by default**
  - Concerns over the security of etcd (including backups)
  - Provider configuration can be tricky
- **TOB-K8S-001: Bearer tokens revealed in logs**
  - Logging on the API Server could be configured in a way which leaked bearer tokens to the logging location
- **TOB-K8S-005: Environment variables expose sensitive data**
  - Some components rely on environment variables to supply secrets

Reviewing your environment

Evaluate your component data flow

- **Identify:**
  - Inbound & outbound component interactions
  - Supported component protocols
  - Nesting of components

Define trust zones

- **Define component existing boundaries:**
  - Controls
  - Policies
  - Placement
- **Define component interaction requirements:**
  - Is a component a dependency of another?
  - What are the failure modes of a component?

Determine criticality

- **What would happen if zone interactions were to stop?**
  - Would other components fail?
  - Would workloads fail?
- **What impact would component compromise have?**
  - Does the component store sensitive information?
  - Does the component parlay access to a previously inaccessible zone?
- **How does placement affect the component?**
  - Can components placed in the same location exhaust available resources?
  - Are certain component locations more sensitive than others?

Discuss assumptions across teams

- **Plan a meeting with the team in charge of each zone**
  - Share the data-flow, zone definitions, and criticalities
- **Attempt to find inconsistencies between team knowledge and your assumed knowledge**
  - Document any that are identified
- **Confirm that the data-flow and zone definitions are comprehensive**
  - Components often have features with very little documentation which the team may have insight on

Confirm assumptions

- **Review the implementation of each assumption to ensure no undocumented functionality exists, and the documented functionality is operating as assumed**
  - Failure modes
  - Component dependants/dependencies
  - Data retention
  - Logging
  - etc

Remediate

- **File issues for each discrepancy**
  - Try to make short and long term goals for remediation
  - Try to include relevant teams - a fix in one zone could be a flaw for another
- **Ensure documentation reflects changes made for discrepancies**
- **Consult with each team to discuss findings across all zones**
  - There may be deeper design concerns related to the findings

And just like that you reviewed…

- Networking
- Cryptography
- Logging
- Authorization
- Tenancy
- And probably a lot more!

Want document templates? We made some!

- **The threat model has:**
  - An appendix with a “Rapid Risk Assessment” template
  - Example cluster data flow
  - Example cluster trust zones
  - Example findings (found from the assessment!)

Thank you!

Robert Tonic
Security Engineer

robert.tonic@trailofbits.com
trailofbits.com

Stefan Edwards
Assurance Practice Lead

stefan.edwards@trailofbits.com
trailofbits.com