infosecurity NORTH AMERICA TECHTALKS

Brandon Edwards, Chief Scientist
Pete Markowsky, Chief Architect

How to Detect & Defend Against Attacks on Modern Infrastructure

CAPSULE8

Introduction + Topic Overview

- Who are we?
- What do we mean by Modern Infrastructure
- How old attacks apply
- What are the new attacks
- Modern defense
- **Warning:** a LOT of content, 20 minutes means we can only go so deep on each concept

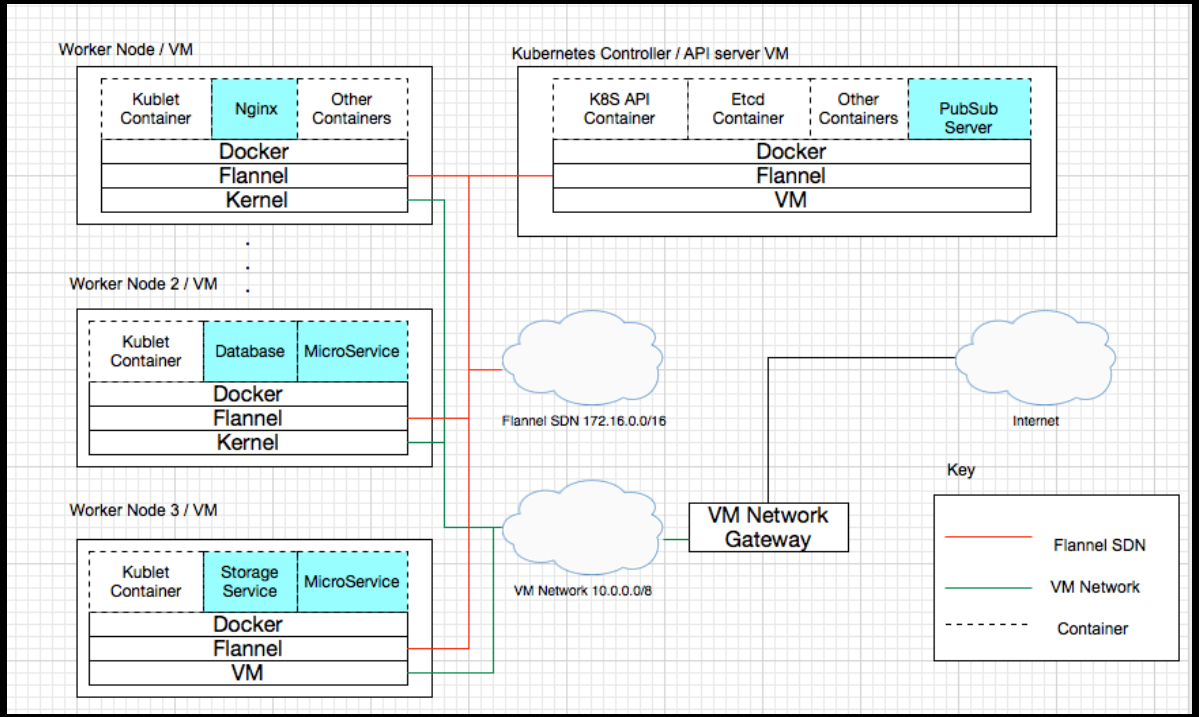
Who are we

- Hackers
- Co-founders of Capsule8
- New Yorkers--Welcome to New York!
- Interested in the intersection of where attacks meet modern infrastructure

What Do We Mean By Modern Infrastructure?

- Uses virtualization
- Uses the cloud and cloud APIs / tools
- Uses orchestration (e.g. Kubernetes, Docker Swarm)
- Employs Continuous Integration / Continuous Delivery
- Automation / use of software defined features
- Assembles pieces together using VMs or containers

Exemplar: Network



Traditional Targets

- User federation, orchestration, and inventory
  - Domain Controller
  - Kerberos
  - OpenView/NNMi
  - VMWare vSphere ESXi
- Many of these still apply, for example Active Directory in the cloud is a thing with Azure

The New Era Targets

- Modern Orchestrators
  - Kubernetes
  - Swarm
  - Puppet/Chef/Ansible
- Glue-wear
  - Enterprise Service Bus / Event Sourcing
  - Caching Layers
  - Key-Value Stores
  - Container Repos/Registries
- Cloud Provider APIs and Tooling

The New Era Target Examples

- Key-value stores are critical components in many modern environments, as getting control over the key-value associations often means control over how and what is deployed, in addition to keys & values being sensitive data

- Modern targets in this category:
  - etcd
  - memcached
  - KV SecretsEngine

The New Era Target Examples

- Cloud-provider APIs are rich and powerful, but that means they're also important to protect as they become primary targets for attack

- For example, the AWS metadata endpoint is available locally to all instances at http://169.254.169.254

- This will (or has) become one of the first places an attacker looks to pivot / migrate laterally after initial compromise of an AWS instance

Previously Impractical Attacks

In addition to new software targets, the speed and strength of modern computing lends itself to attacks which were previously considered impractical outside of a lab

- **Timing attacks** can exploit precision of modern infrastructure
- The speed, reliability, and network proximity of cloud instances makes exploiting textbook string comparison timing vulnerabilities a viable network-based attack:

```
if (suppliedpassword == storedpassword) {... }
```


Previously Impractical Attacks

In addition to new software targets, the speed and strength of modern computing lends itself to attacks which were previously considered impractical outside of a lab

- **Memory-corruption** attacks get more chances in highly-available environments, especially when they're configured to autoscale
- **Integer wraps** requiring 32gb of memory **CVE-2018-14634**
- **Side-channels** to exploit co-tenancy (Spectre, Meltdown)

Modern Infrastructure Trade-Offs

- CI/CD makes your environment dynamic, which means you're a moving target, combined with agility a speedy recovery means you're more resilient
- But it also means that there are new moving pieces to manage
- Each piece of glue-wear is nuanced, all of which could be insecurely misconfigured
- There are pros and cons, but our stance is that if properly managed, the tooling for modern infrastructure is generally better from a security perspective than legacy environments

OPP: “Other People’s Pieces” Problems

- Complicate authorization models
  - E.g. PubSub / Event Sourcing
- Limited authorization
  - Often accept any certs or passwords
  - Limited ability to monitor / disparate data
    - Cloud trail users
- Varying levels of granularity and control

Examples of Modern Attacks

- Token theft
- Overly Permissive Resources
  - S3 buckets open to the world
  - Mounting or mapping privileged resources into containers (/proc, Docker socket, etc)
- API Abuse (Kubernetes)
- Orchestrator-Specific Attacks

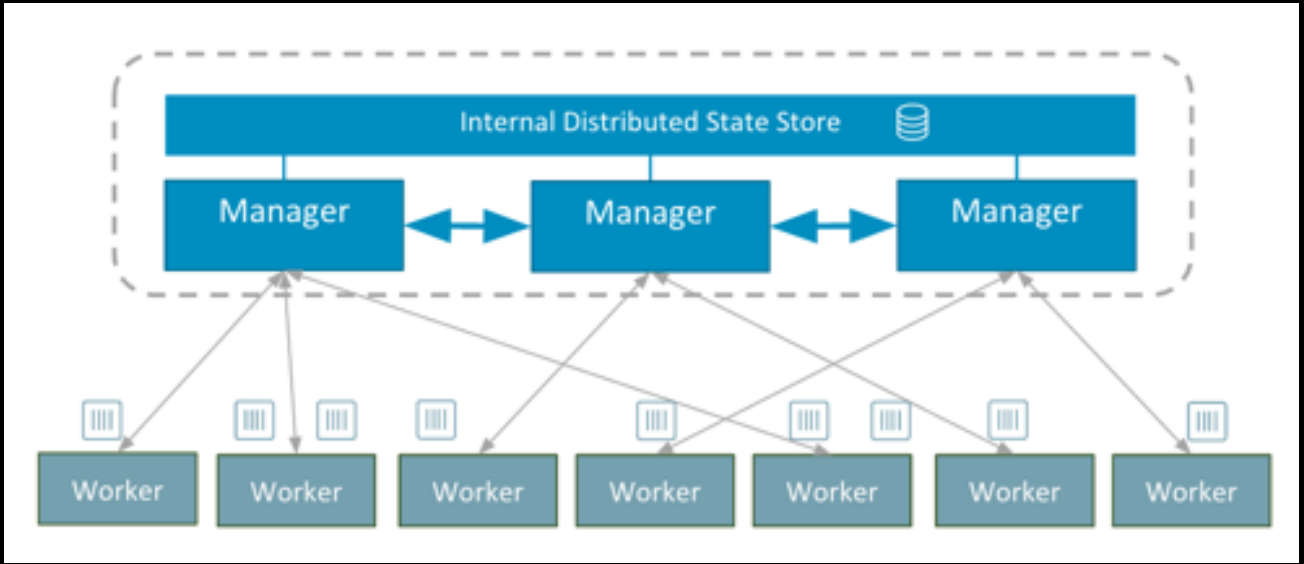
Examples of Modern Attacks

- Conceptually, modern attacks exploit the same weaknesses that have always been present, but the nuances of modern environments mean the impact is different

- Attack surface has become more nebulous to model as automation tooling abstracts away relationships between services, workloads, data, and users

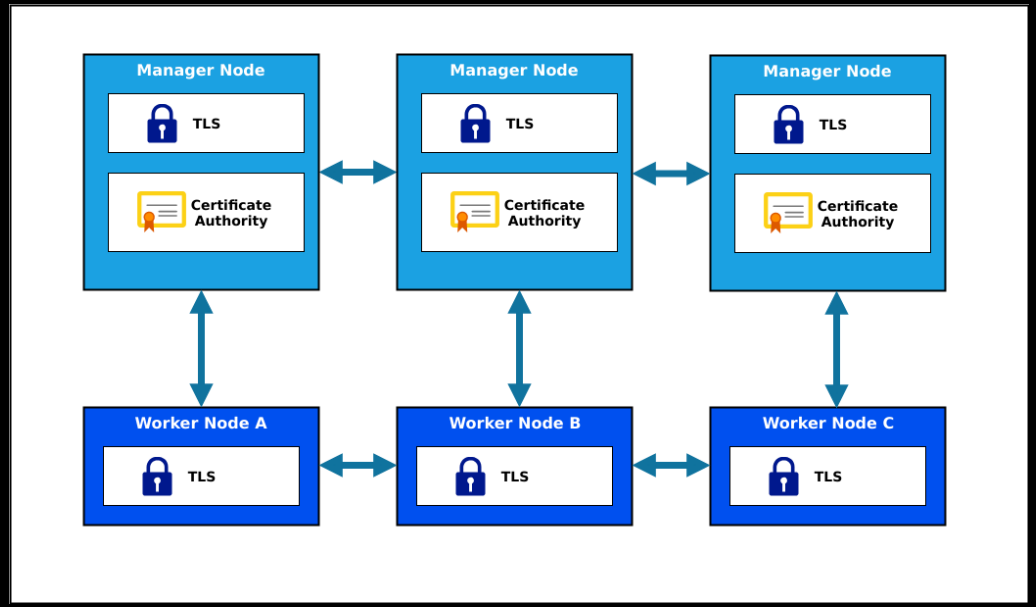
Case Study: Docker Swarm

Docker's built-in orchestration, available since 1.12



Case Study: Docker Swarm

Also the first orchestrator to introduce a security model

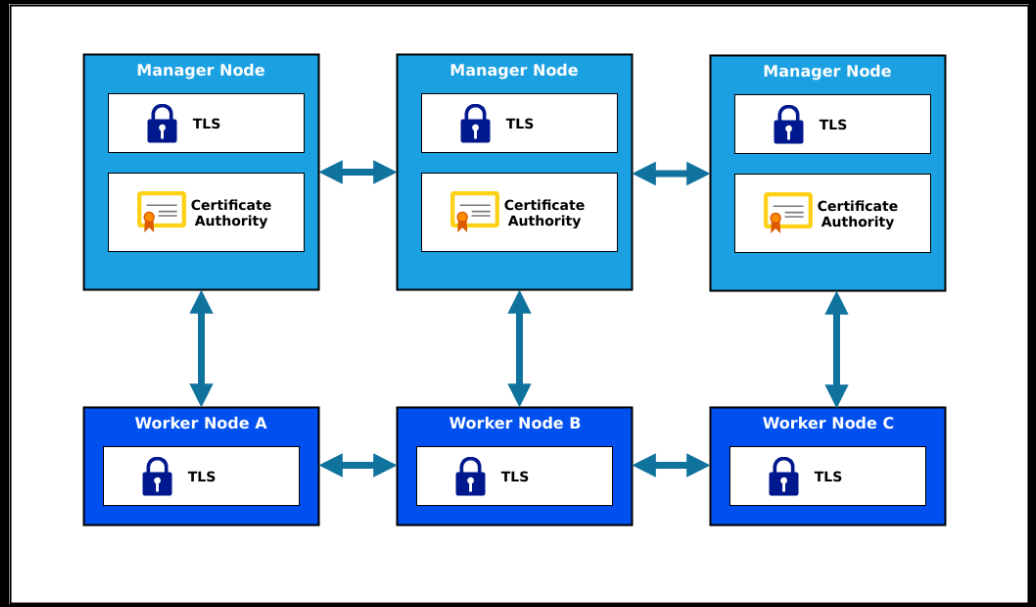


Case Study: Docker Swarm

- Example of Orchestrator-Specific Attack
- The attack exploits a combination of
  - Lack of finely-grained worker authorization
  - ARP-based DoS (for worker deregistration)
  - Credential replay
  - Optionally local credential extraction (not required)
- Results in harvesting cluster-wide application **secrets** managed by the orchestrator
- Closest analogy is “pass-the-hash” for orchestrators

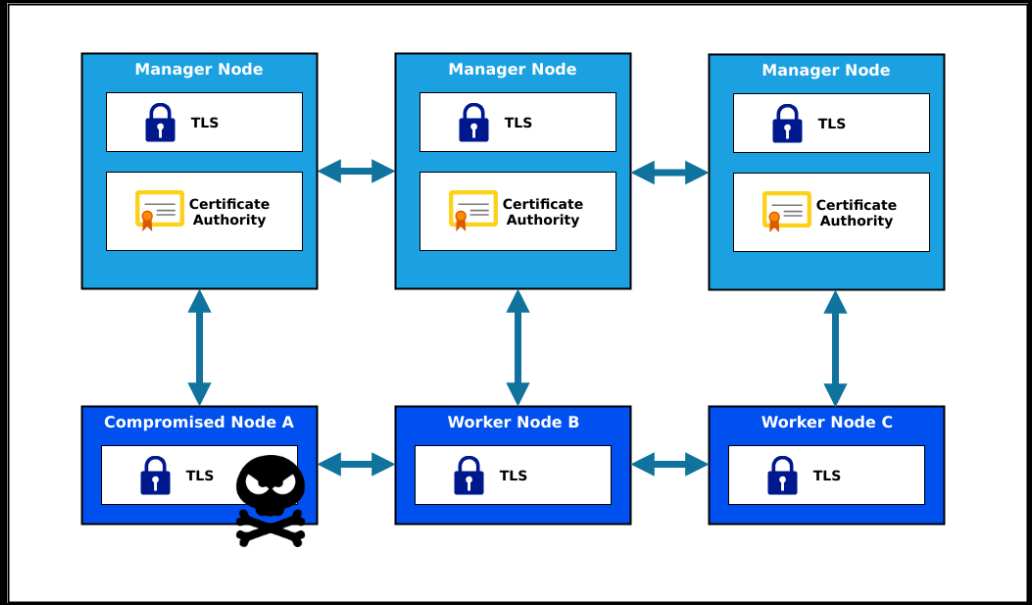
Case Study: Docker Swarm

SWARM manages delivery of secrets as needed by workers

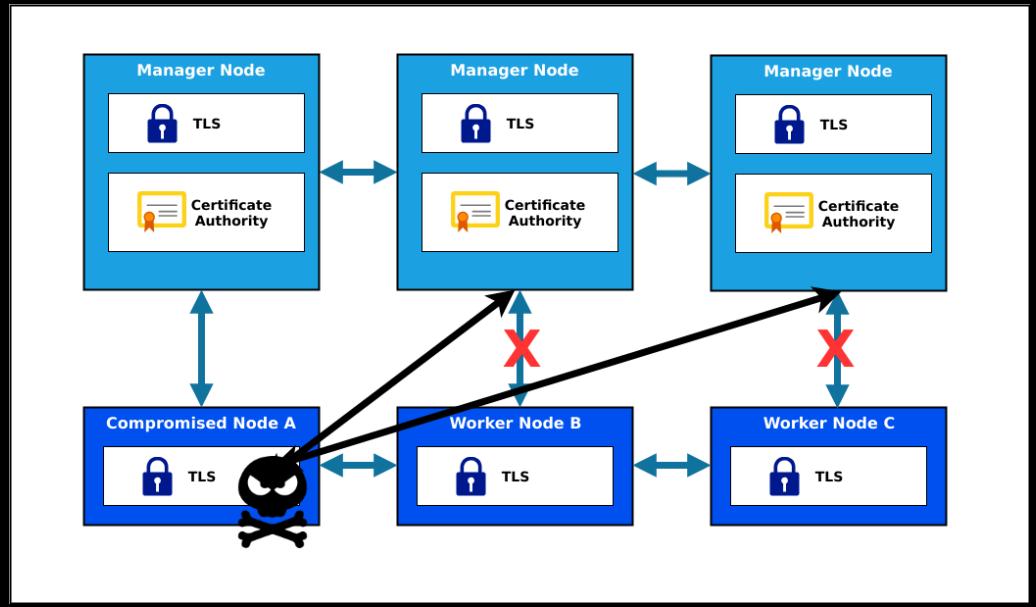


Case Study: Docker Swarm

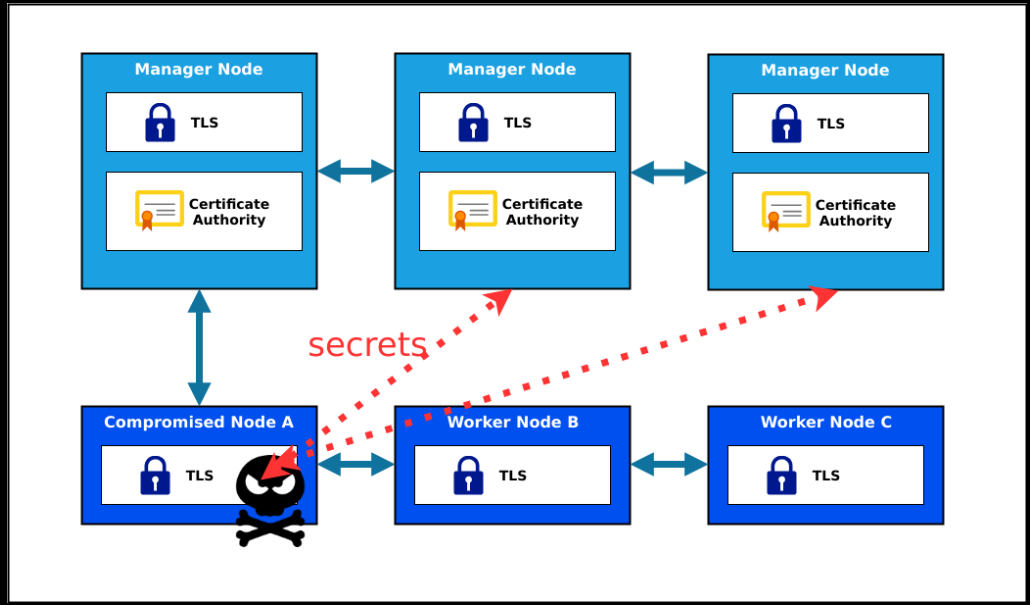
...but a single compromised worker can subvert secrecy



Case Study: Docker Swarm



Case Study: Docker Swarm



Case Study: Docker Swarm Defense

- Segmentation: workloads with varying levels of data sensitivity should be isolated to their own clusters and nodes
- An option, if varying sensitivity is required, is have a roll-over/HA cluster, and redeploy entire service upon some percentage of service outage (creating new Join Tokens means compromised node cannot rejoin)
- This sort of spoofing should be a pretty loud signal to detect.
  (Why is a single worker node chatting up all the manager nodes?)

Defense: Prevention

- Scan for Tokens / Open APIs / Resources
- Avoid shipping containers with secrets in them, isolate the secrets separately
- Avoid secrets in code wherever possible: this was true in the old paradigm of defense, remains true: API keys, SSH keys etc are the new hard-coded passwords
- Github now supports scanning for tokens and keys

Defense: Prevention

- Even after removing secrets from containers, avoid exposing your container repository publicly (this is still valuable recon data), or manage a separate public repo for containers intended to be consumed externally

Defense: Prevention

- Secrets Management
  - Vault / Cloud
  - KMS

- Segmentation
  - mTLS
  - Not everything in same cluster, pod, or container

Defense: Prevention

- Immutable infrastructure
  - Immutable Servers
  - Immutable Containers

- Reverse Uptime
  - Keep your servers / services fresh!
  - Force adversaries to repersist

Defense: Monitoring

- Rapid change makes monitoring hard
  - Software defined means monitoring just from the network is incomplete
- Context is king
- Diffing “likeness” of service resources
- You need metadata
  - metadata from your orchestrator
  - metadata from your cloud (cloud labels)
  - metadata from your containers / VM images

Defense: Monitoring Goals

- What process, container, and pod performed a particular suspicious action?
- What else did it do?
  - e.g. Networking in same Pod or within the Kubernetes cluster
- Why did that process perform that suspicious action?
  - Where did that process come from?
  - Did a human run a shell in the container or did nginx?
- What commands were executed from a particular shell session?

Examples of Modern Defense

- Scan public resources for tokens -- Github has a scanner now!
- Vuln Scanning of containers
- Immutable infrastructure practices
- 2FA everywhere!
- Drift detection
- Canaries
- Advanced Stuff
  - Crypto-Anchoring
  - Zero Trust Networks (auth everything!)

5 KEY TAKEAWAYS

5 Key Takeaways

- There are trade-offs for both offense and defense in the modern environment
- Modern infrastructure exposes new APIs and data to protect (tokens) – ther is a new attack surface that needs inventory / tracking
- Be cognizant of the gluewear, and third-party APIs, they can become vectors for supply-chain attack, be wary of your software dependencies and their source(s) (and recursive chain of their dependencies)
- The new APIs can also work for defense: they empower you to collect data for detection and forensics
- The dynamic environment of CI/CD means attackers get retries -- but it also means persistence is harder, and you can constantly redeploy for agility/resilience

Thank you!

Questions?

brandon@capsule8.com

pete@capsule8.com