SAND2023-05340C

Sandia National Laboratories

Exceptional service in the national interest

THE SECURE CONTAINER IMAGE MIRROR

Jonathan Grzybowski

Matthew Morse

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly artment of Energy's National Nuclear Security Administration under contract DE-NA0003525.

PROBLEM

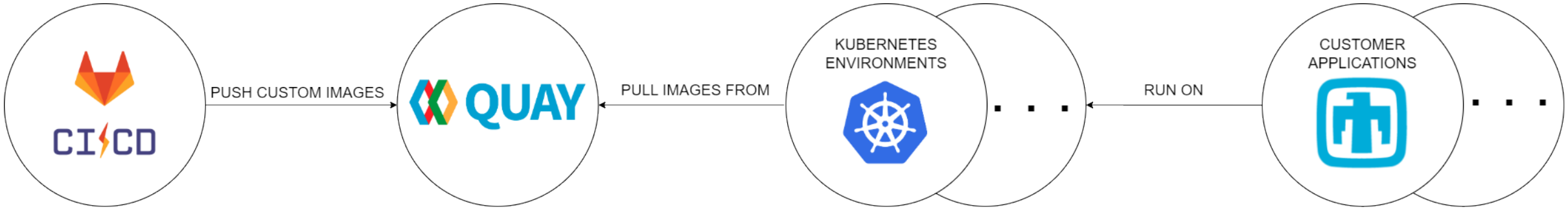
Our team provides application hosting across two different networks.

We aim to provide a mechanism to automatically mirror customer container images from one network to another for more consistent environments.

The image transfer mechanism travels over an unreliable channel.

CUSTOMER ENVIRONMENTS

NETWORK ENVIRONMENT



OUR GOAL

- Want environments consistent across networks
  - Matching versions of customer applications
  - Versioning of infrastructure
  - Manage config drift
- Reduce time from compliance to deployment
  - Different compliance levels across networks
- Minimal human interaction desired

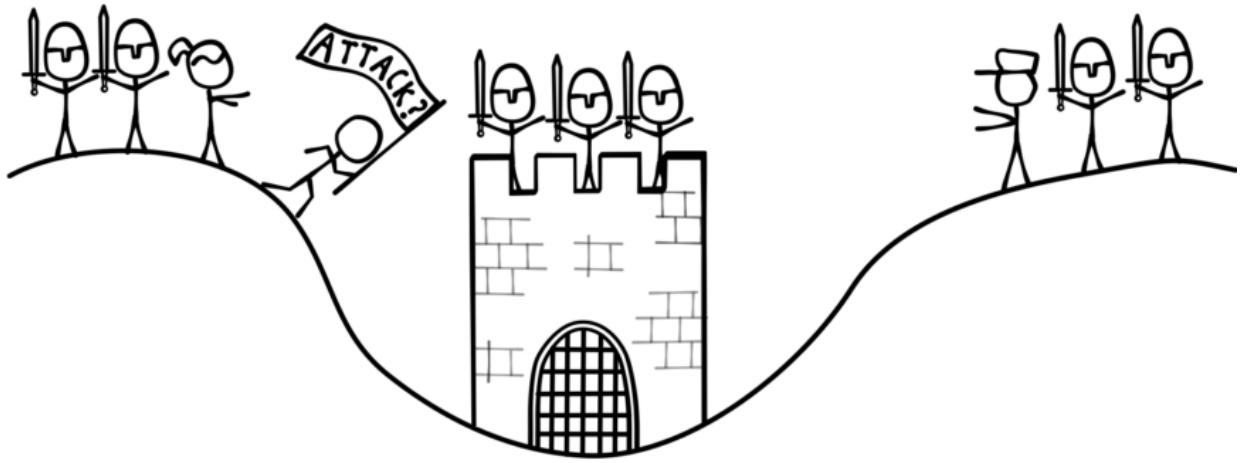
FILE TRANSFER PROCESS

CROSS-NETWORK FILE TRANSFER SERVICE

- Bottleneck between networks
- API webserver on each network
- One-way diode
  - No acknowledgement of receipt
  - Insufficient integrity guarantees — manual integrity check
- Potential errors
  - Intermittent data corruption
  - No receipt of data
  - Downtime and degradation of diode
  - API unresponsive within request timeout interval

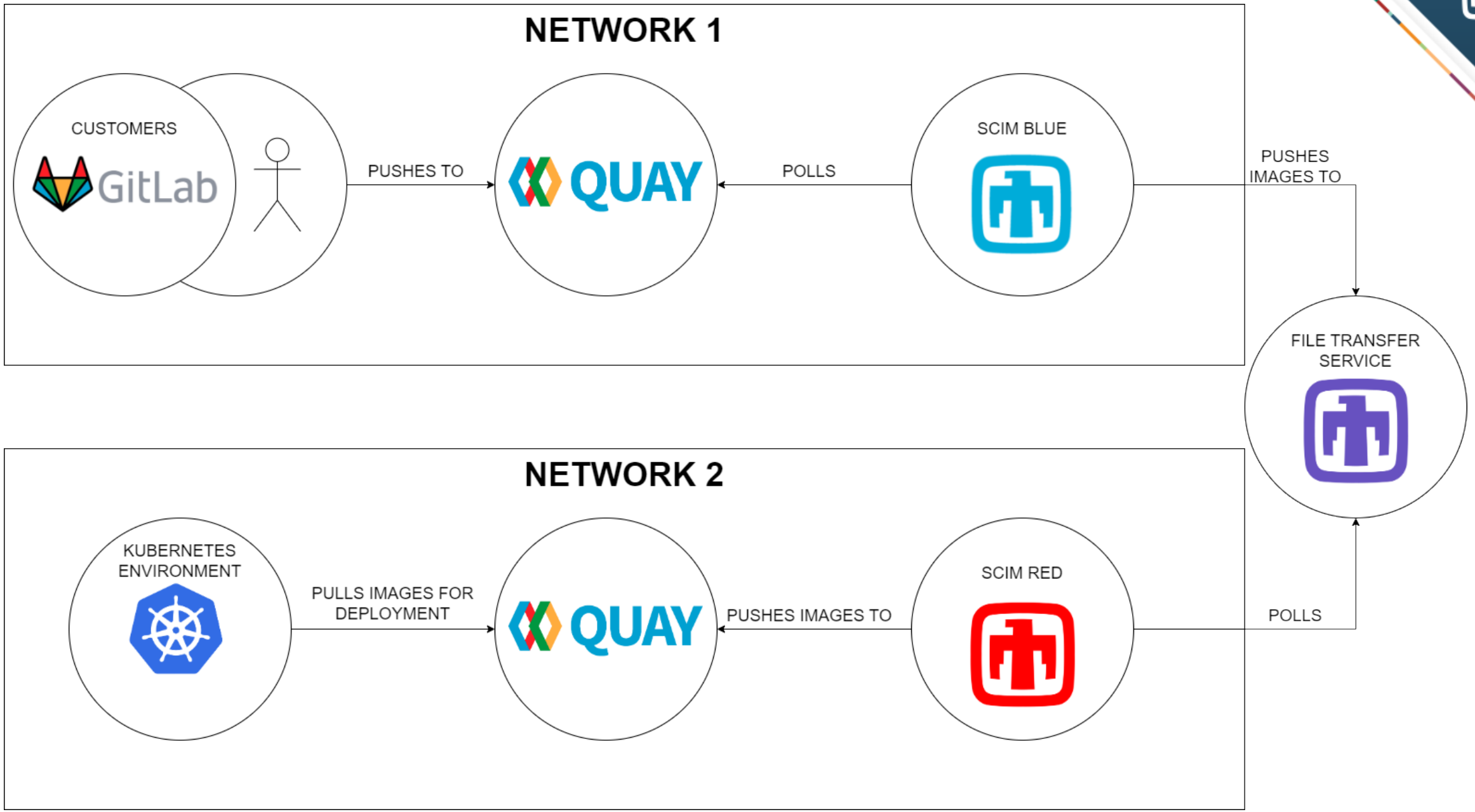
TWO GENERALS

- Two attacking generals
- Fort between them
- Rules
  - If 1 attacks, both fail
  - If both attack, they succeed
  - Communication is potentially unreliable
- Generals communicate back-and-forth



https://haydenjames.io/the-two-generals-problem/

THE SECURE CONTAINER IMAGE MIRROR (SCIM)


NETWORK 1
CUSTOMERS
GitLab
PUSHES TO
QUAY
POLLS
SCIM BLUE
PUSHES IMAGES TO
FILE TRANSFER SERVICE
NETWORK 2
KUBERNETES ENVIRONMENT
PULLS IMAGES FOR DEPLOYMENT
QUAY
PUSHES IMAGES TO
SCIM RED
POLLS

SCIM MANIFEST

- Rules based
- Image regex
  - Quay image form
    - namespace/repository:tag
- Vulnerability gate
  - gate – highest allowable severity
- Allow special unknown/unsupported severities

```
namespaces:
   - namespace
   - namespace2

rules:
   - regex: namespace/repo.*:tag
     gate: High
   - regex: namespace2/repo.*:tag
     gate: Low
     allow_unknown: true
     allow_unsupported: true
```

VALID IMAGES

- Database of processed images
  - Name
  - Image ID
  - Image manifest digest
  - Upload attempts
  - Max retries

```
6  class Image(Base):
7      __tablename__ = 'images'
8
9      id = Column(Integer, primary_key=True)
10     name = Column(String)
11     image_id = Column(String)
12     manifest_digest = Column(String)
13     retries = Column(Integer)
14     max_retries = Column(Integer)
```

- Check image manifest digest and name doesn't exist in DB
  - Image isn't already processed
  - 5 or 500 requests per batch?
- Check image in manifest namespaces list
- Check image matches at least one rules regex

IMAGE COMPLIANCE

- Clair
  - Integrated Quay container scanner
- Aggregates vulnerability information from 3rd Parties
- Assigns severity of image vulnerabilities (Low, Medium, High, Critical)
  - Issues with Unknown or Undefined severities

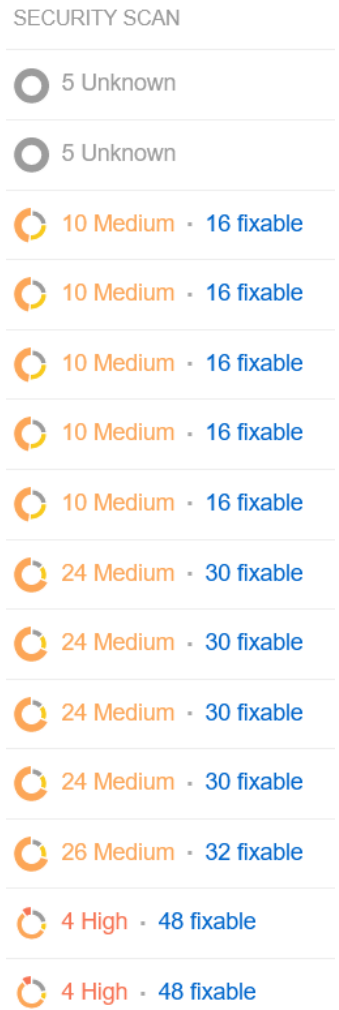


IMAGE COMPLIANCE

- SCIM Blue queries Quay
- Polls images
  - Includes vulnerabilities and associated severities
- Sorts in descending order list of severities
- Given the matching rule in manifest, determine if maximum severity passes gate threshold
  - Temporary issues with “clean” images
  - Unexpected issues with multi-architecture images and child manifests

FILE TRANSFER PROCESS

- Save container image as tarball
- Package with versioned metadata file
  - Quay namespace/repository:tag
  - Network 2 Quay destination URL
  - Image ID
  - Manifest digest
- Image tarball HMAC
- Metadata file HMAC
- Transfer service-generated metadata
  - Creation time
  - Receipt time

```
4   @dataclass(frozen=True)
5   class Image:
6       namespace: str
7       repository: str
8       tag: str
9       image_id: str
10      manifest_digest: str
```

```
20  @dataclass(frozen=True)
21  class MetadataV1(Metadata):
22      image: Image
23      target_registry: str
24      version: int = field(default=1, init=False)
```

IMAGE RETRIEVAL

- SCIM Red
- Poll transfer service for new images
- Checks automation account
  - Potential for undesired uploads
  - Identify images via prefix
  - Verify images via HMAC
- Check if namespace exists on Network 2
- Create repo if it doesn’t exist
- Images pushed to same location from Network 1

RESULTS

NETWORK 1

- 6 teams currently supported
- Hundreds of images transferred
- Dependencies cause the most service interruptions
  - Quay
    - Avg 7 hours to restore after error
    - Max 16 hours
  - Transfer Service
    - Avg 2:20min to restore after error
    - Max 4:20min to restore after error
- Parallel chunk uploads
  - Currently ~5 100MB chunks uploaded in parallel
  - GC must be called after each batch to prevent interpreter OOM errors

NETWORK 2

- No instances yet of HMAC not matching
- Dependency degradation
  - Intermittent spot errors only
  - Immediate resolution
  - Presumed API hiccups
- Can download ~4 images at a time
  - Memory and filesystem space issues

TRANSFER DIODE

- Two states
  - Normal
    - Median image transfer time: ~3:30min
  - Degraded
    - Transfer diode can take multiple days (max 2.5 days)
- No instances yet of data corruption on receipt

ERRORS ENCOUNTERED

- Dependency on uptime of outside software
- Unexpected results from Quay
  - Vulnerability scans
  - API request headers
- Overloading Quay with API requests
- Balanced logging
- Lost images when Quay is degraded

FUTURE GOALS FOR SCIM

- Intermittent image reupload to mitigate issues with diode downtime
- Allow customers to self sign-up
  - Potential policy/security concern
  - Large effort to implement/integrate sign-up service
- More fail-safes for handling degraded dependencies
- Investigate larger memory footprint for Python interpreter

QUESTIONS

Jonathan Grzybowski
jgrzybo@sandia.gov

Matthew Morse
mlmorse@sandia.gov