ThoughtWorks®

Cabinet Office

# Tin Tulip - Blue team

Showcase #3 - May 5

# Agenda

What we achieved

What’s next?

# Summary

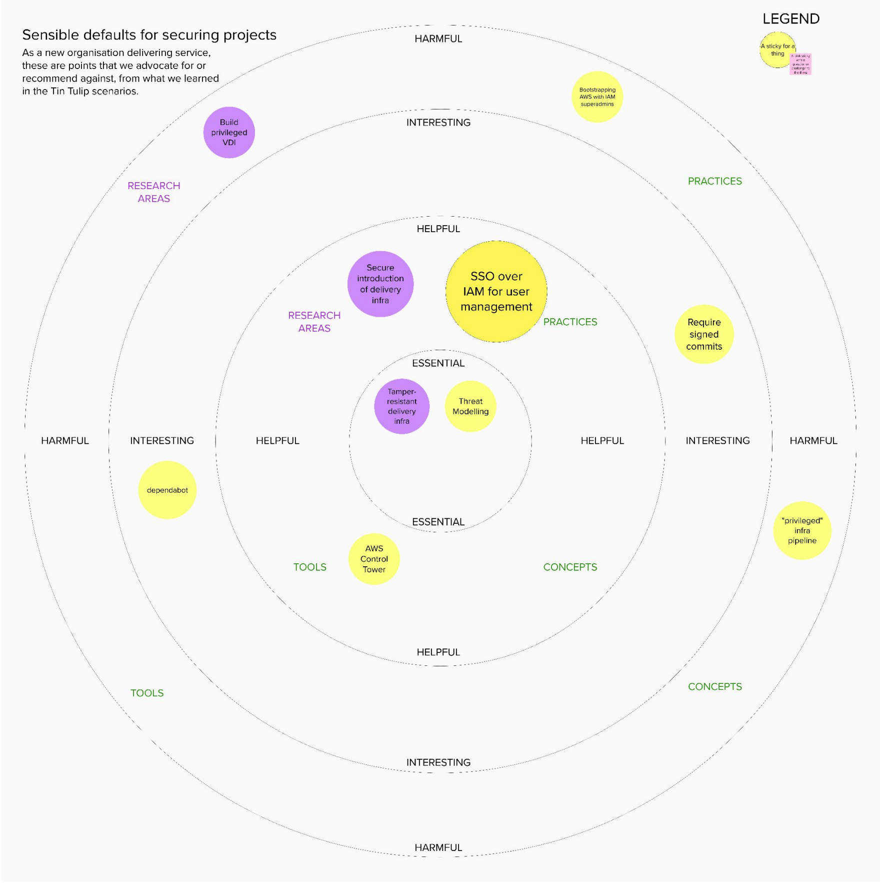
*The web team continues to build out the test AWS platform according to guiding principles.*

*The key next step is making a web front end available, in order to complete an end to end environment in time for the red team joining.*

# What we achieved

# What we worked on

- Replacing AWS IAM "owners" with SSO



Sensible defaults for securing projects

As a new organisation delivering service, these are points that we advocate for or recommend against, from what we learned in the Tin Tulip scenarios.

*Zoomable diagram available on Mural at this link (signup required)*

# Replacing AWS IAM "owners" with SSO

**What we built:**

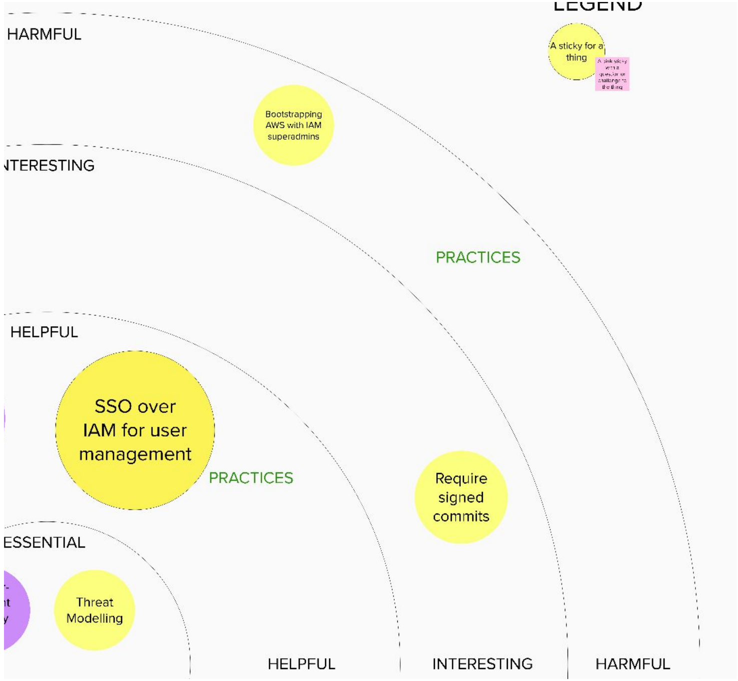
Following up on Control Tower, used its SSO

**Why we built it:**

Adopting better tooling for better PAM in AWS

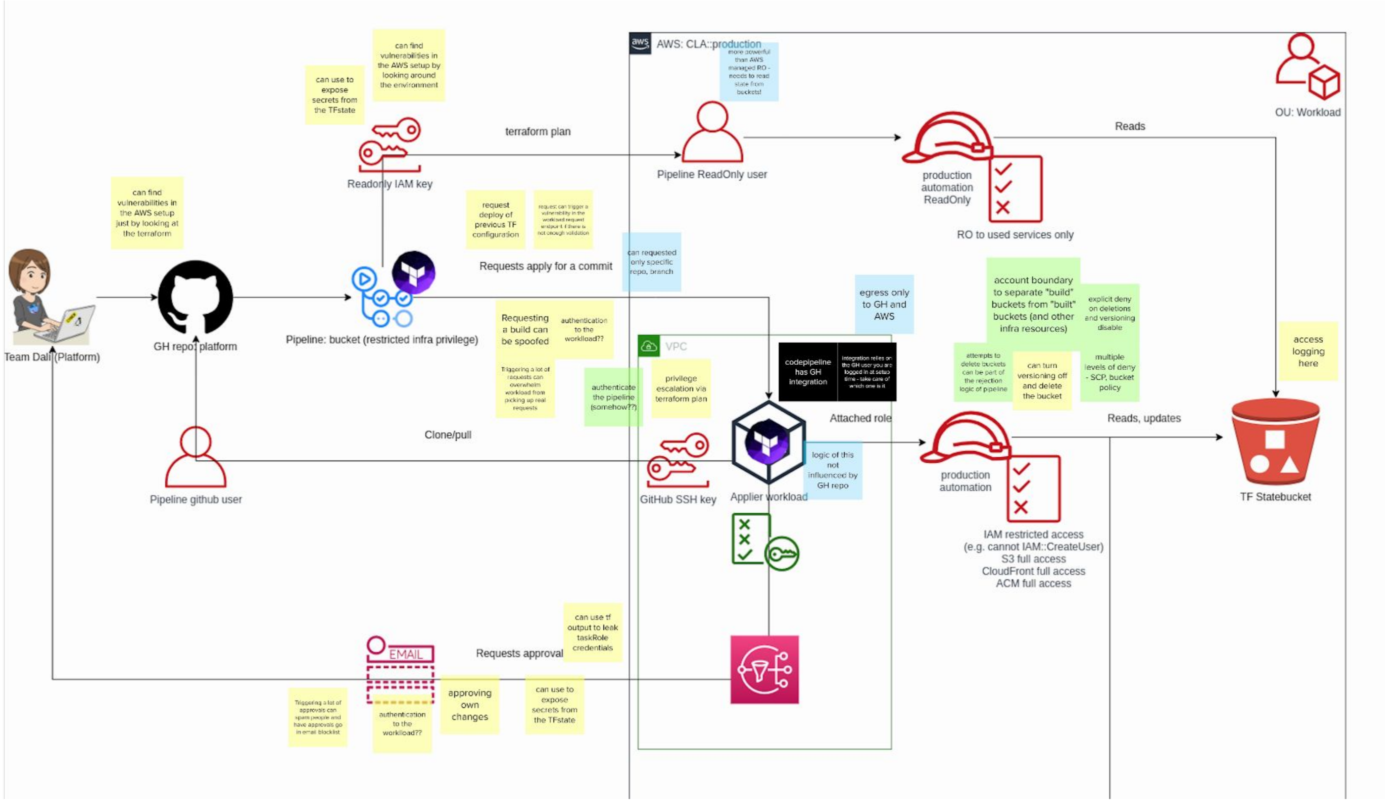
**What we learned from it:**

- There is no good reason to "bootstrap" an AWS org with privileged IAM users if you're adopting Control Tower - go straight to SSO instead



# Threat modelling #3

# Threat Modelling



can use to expose secrets from the TFstate

can find vulnerabilities in the AWS setup by looking around the environment

more powerful than AWS managed RO - needs to read state from buckets!

can find vulnerabilities in the AWS setup just by looking at the terraform

request deploy of previous TF configuration

request can trigger a vulnerability in the workload request endpoint if there is not enough validation

can requested only specific repo, branch

egress only to GH and AWS

Requesting a build can be spoofed

authentication to the workload??

Triggering a lot of requests can overwhelm workload from picking up real requests

authenticate the pipeline (somehow??)

privilege escalation via terraform plan

codepipeline has GH integration

integration relies on the GH user you are logged in at setup time - take care of which one is it

logic of this not influenced by GH repo

account boundary to separate "build" buckets from "built" buckets (and other infra resources)

explicit deny on deletions and versioning disable

attempts to delete buckets can be part of the rejection logic of pipeline

can turn versioning off and delete the bucket

multiple levels of deny - SCP, bucket policy

access logging here

can use tf output to leak taskRole credentials

Triggering a lot of approvals can spam people and have approvals go in email blocklist

authentication to the workload??

approving own changes

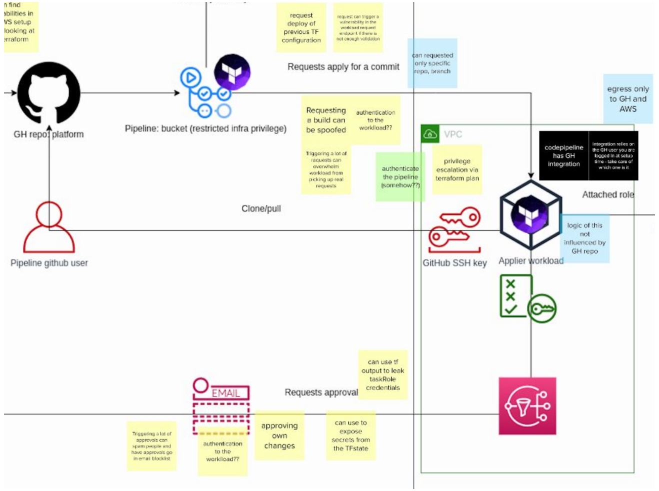
can use to expose secrets from the TFstate

*View full article on GitHub*

# Threat Modelling

Key takeaways:

- "Inverting" the interaction between GitHub and AWS mitigates many pipeline threats mentioned so far
- Requires a "TF applier workload" within AWS
  - allows to introduce trusted evaluation of policy and checks
  - needs to be locked down itself
- Account isolation from workload and prod would mitigate main risks identified


n find
abilities in
VS setup
looking at
erraform
request deploy of previous TF configuration
request can trigger a vulnerability in the workload request endpoint if there is not enough validation
Requests apply for a commit
can requested only specific repo, branch
egress only to GH and AWS
GH repo: platform
Pipeline: bucket (restricted infra privilege)
Requesting a build can be spoofed
authentication to the workload??
Triggering a lot of requests can overwhelm workload from picking up real requests
authenticate the pipeline (somehow??)
VPC
privilege escalation via terraform plan
codepipeline has GH integration
integration relies on the GH user you are logged in at setup time - take care of which one is it
Attached role
Clone/pull
Pipeline github user
GitHub SSH key
Applier workload
logic of this not influenced by GH repo
EMAIL
Requests approval
can use tf output to leak taskRole credentials
Triggering a lot of approvals can spam people and have approvals go in email blocklist
authentication to the workload??
approving own changes
can use to expose secrets from the TFstate

# What’s next?

# Current focus

- Introduce security tooling in pipeline (writeup in progress)
- Complete pipelines for team Dalí and Mozart
- CLA website published

# Appendix: Guiding Principles

# Guiding principle for the project

*Does this teach us something new about a security control, or how to defeat it?*

# Guiding principle for platform implementation

*In order to research the known security boundaries, the blue team will implement a test platform based on published best practices, including those published by the NCSC*

# Guiding principle for communicating learnings

*The key audience for learnings are government departments, who want to empower their local technology teams to deliver secure systems*