THE CALL IS COMING FROM INSIDE THE CLUSTER

Mistakes That Lead to Whole Cluster Pwnership
WHO ARE WE AND WHAT ARE WE DOING HERE?

- Dagan Henderson
- Will Kline
- Not formal pentesters. Just Kubernetes guys who noticed some things.
- You don’t have to be a Kubernetes developer, you just have to think like one.
- No 0-Days in this presentation. Everything has been responsibly disclosed.
WTF IS A KUBERNETES?

- Take a sprinkle of Docker containers
- Mix in a handful of servers
- A pinch of orchestration
- You gots yourself a “cluster”
WTF HAPPENS WHEN YOU PWN A CLUSTER?

- Run your own workloads
- Modify existing workloads
- Hijack traffic
- Escape to a host
PHASE I: NO MORE SECRETS
ISTIO AND KIALI

- **Common mTLS service mesh**
  - Service discovery
  - Bolt-on E2E encryption
  - Incredibly robust service mesh
  - Generally a Good Thing™

- **Kiali**
  - Network visualization tool
  - Usually installed with Istio
  - Very cool!

- You should absolutely use both of these!
JWT THIS DOWN

- JSON Web Token
- Pronounced “jot”
- Defined by [RFC 7519](https://tools.ietf.org/html/rfc7519)
- Super Popular
- Includes an HMAC signature
  - What could go wrong?
  - As we all know, crypto is so easy!
One day, Dagan noticed something...

Most applications force you to log in again every time
  - Especially when you completely reinstall it

Kiali didn’t
  - That's weird

Even when he recreated the pods, his session didn't end
  - It must have persistence somewhere
  - It didn’t have any configuration for persistence

That’s weirder
DOWN THE RABBIT HOLE

1. K8s secrets
   ○ Store “private” configs
   ○ Helm conf doesn't have any
   ○ How is it signing JWTs?
2. Start looking at code...
   ○ The logic was set in the code
   ○ ... But not in the config
   ○ The only option is “kiali”
3. CVE-2020-1764
DEMO: PHASE 1
One more thing...
IT’S WHAT MISSING THAT COUNTS

```golang
func GetTokenClaimsIfValid(tokenString string) (*IanaClaims, error) {
    token, err := jwt.ParseWithClaims(tokenString, &IanaClaims{}, func(token jwt.Token) (interface{}, error) {
        return []byte{Get().LoginToken.SigningKey}, nil
    })
    if err != nil {
        return nil, err
    }

    if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
        return nil, fmt.Errorf("unexpected signing method: %s", token.Header["alg"])
    }

    if token.Valid {
        cfg := Get()
        claims := token.Claims.(*IanaClaims)

        if claims.Issuer != AuthStrategyLoginIssuer && claims.Issuer != AuthStrategyOpenshiftIssuer {
            return nil, errors.New("token has invalid issuer (auth strategy)"
        }
            return nil, errors.New("token is invalid because of authentication strategy mismatch")
        }
            return nil, errors.New("token is invalid because of authentication strategy mismatch")
        }

        return token.Claims.(*IanaClaims), nil
    }

    return nil, errors.New("invalid token")
}
```
DEMO 1 RECAP

- Gain access to an exposed Kiali
- Kiali has API access
  - Discover other services
  - Read pod-logs
  - Modify Istio Traffic Management
  - Create brand new ingress routes!
DEMO 1 RECAP

- Gain access to an exposed Kiali
- Kiali has API access
  - Discover other services
  - Read pod-logs
  - Modify Istio Traffic Management
  - Create brand new ingress routes!
- So how does a mistake like this happen?
Thank you for coming to our DEF CON talk!
DEMO 1 RECAP

- Gain access to an exposed Kiali
- Kiali has API access
  - Discover other services
  - Read pod-logs
  - Modify Istio Traffic Management
  - Create brand new ingress routes!
- Kiali can run in R/O mode
  - Least Privilege
  - It would slow us down
PHASE II: THE PIVOT
WHAT IS A FLEET?

- Rancher’s built-in GitOps tool
- Git >>> K8s
  - Reads YAML from Git
  - Makes it happen in K8s
  - Generally a Good Thing™
- Truth and Reconciliation Loop
  - Reverts changes
  - Generally a Good Thing™
LOGGING AND YOU  (ALSO, WHERE DID I LEAVE MY KEYS?)

- When logging, developers tend to either whisper or shout
- Logs also have a tendency to end up all over the place via forwarders
  - ELK
  - K8s pod-log
- Because of complex call stacks, Fleet logs *the entire* SSH *private* key on Git pull errors
- If the GitOps key is not read-only, it’s game over
- If the GitOps key *is* read-only, it’s still *very* valuable recon
DEMO PHASE 2
DEMO 2 RECAP

- Pivot from Kiali Dashboard
  - Observe Fleet logs
  - Gain *private* SSH Key
- Git access to GitOps repo
  - Deploy arbitrary workloads
  - Tend to be *very* permissive RBAC
- Deploy the Evil Workload in Fleet controlled namespace
DEMO 2 RECAP

- Pivot from Kiali Dashboard
  - Observe Fleet logs
  - Gain *private* SSH Key
- Git access to GitOps repo
  - Deploy arbitrary workloads
  - Tend to be *very* permissive RBAC
- Deploy the Evil Workload in Fleet controlled namespace
- So what went wrong here?
PHASE III: I CAN HAZ CLUSTER?
LONGHORN: MICROSERVICE BASED STORAGE

- Kubernetes doesn’t provide “Persistent Storage” by default
- It uses an API, called the Container Storage Interface
- Cluster owners choose an implementation
- Rancher created Longhorn
- Different pod types
  - Instance managers run on all worker instances (DaemonSet)
  - Longhorn managers wrangle instance managers
UNCONTAINABLE CONTAINERS

- We were trying to write a more specific PSP for Longhorn pods
  - All sorts of “bad” behavior from Longhorn
  - Some required root access to the host
  - Actually mounts the host root in the container
- Also trying to make Longhorn work with Istio’s mTLS
  - Nothing was clear
  - Ports were everywhere
  - What was evening happening?
- What is this pod even doing and why is it basically literally root?
NEW PHONE, WHO DIS?

- **Still No More Secrets**
  - There was no K8s Secret being generated/mounted/passed/encanted
  - Very similar to the Kiali situation from before

- **How were the Longhorn micro-services authenticating each other?**
  - Unlike Kiali, there wasn’t even a default key being used

- **CVE-2021-36780**

- **If you can dial a Longhorn Manager pod, it answers**
  - So what can we do with *that*?
I'M RON BURGUNDY?

- gRPC and protocol buffers are dope for devs and hackers penetration testers
- Host root dir mounted inside the pod
  - We could read/write to anything on the host. Full Stop.
- CVE-2021-36779
DEMO PHASE 3
DEMO 3 RECAP

- Find Longhorn Instance Manager
- Send commands
- Full host FS access
- chroot /host/proc/1/root and be root on the host
DEMO 3 RECAP

- Find Longhorn Instance Manager
- Send commands
- Full host FS access
- chroot /host/proc/1/root and be root on the host
- Who forgets to secure an entire API?
THE BIG TAKEAWAYS

● Don’t assume they did it right

● Good Code != Good Deployment
  ○ Microservices need to authenticate each other
  ○ Look for secrets, or lack of secrets in Helm charts

● Don’t forget to check the logs
  ○ Logs tend to aggregate secrets
  ○ A lot of things have access to read logs

● Use pod configurations to find juicy targets
  ○ Many applications have public Helm charts
  ○ Even closed-source software
  ○ Look at the pod spec
QUESTIONS?

Will Kline
https://darkwolfolutions.com/join-the-pack
defcon-outreach@darkwolfolutions.com

Get the Demo Project:
https://github.com/dagan/defcon30

Dagan Henderson
https://goraft.tech/join

We’re Both Hiring!

Twitter: TechDagan
Facebook: Dagan