Building a Security Program for SaaS Product Development

Christian Bauer
BSides Munich, 16.05.2022

About Me

- **Have been working on cloud security since ~2017**
- **Currently working as a security engineer for a company with an as-a-service offering.**

This talk summarizes my experiences collected over these years.

Story line of this talk

- **Today is your first working day at the startup ACME!**
- **You are the first security engineer hire. It’s your job to help improving the security of their product.**
- **Their product is:**
  - A platform/software-as-a-service (PaaS/SaaS) offering
  - Deployed on the well known hyperscalers (AWS, Azure, GCP, ...)
  - Basic tech stack: containers, Kubernetes, some cloud provider native services

Day 1 + 1 hour

Where to start?

How to approach this in a structured way?

What are the low hanging fruits?

What provides the biggest impact?

Picture from https://unsplash.com/photos/5eThdzpVqyE

A Product Security Roadmap for ACME (Day 2)

- Group security activities into different phases
- Phase 1: the basics
- Phase 2+: additional activities, grouped by (perceived/opinionated) priorities

- Please note: your mileage may vary. Composition of phases depends on: your background, current security posture, in-company support, etc.

A Product Security Roadmap /2

Cloud Inventory | SSO & MFA | IAM | Network Perimeter | CSPM | Centralized Logging
Vulnerability Management | Incident Response (IR)

} Phase 1

Security Baseline | SDLC | K8S Security improvements | Pentests
Security Automation Infrastructure | K8S Security Monitoring

} Phase 2

Risk assessments, threat modeling | DevSecOps & xAST | Network Egress Filtering
Hardening: CI/CD infrastructure | Central log data sink monitoring | IR playbooks | BIA

} Phase 3

Security Trainings | Bug Bounty program | Hardening: VM & container images
Security Metrics Dashboard | Secure-By-Default Building Blocks | IR: Game Days

} Phase 4

Phase 1: Cloud Inventory (the first 1-2 weeks)

- **Obtain a list of all existing cloud environments (AWS accounts, GCP projects, ...)**
- **What cloud provider services are in use? What infrastructure is deployed? How is that infrastructure configured?**
- **There are tools that can help with that: ScoutSuite (AWS, Azure, GCP), Steampipe (AWS, Azure, GCP), Cloudmapper (AWS), Cloudgraph (AWS), Prowler (AWS), ...**

Phase 1: Perimeter Protection

Identity & Access Mgmt (IAM)

- **This is super critical for Cloud provider API: *"Identity is the new perimeter"***
- **Eliminate long-term credentials s.a. AWS IAM user access keys or passwords, GCP service account user-managed keys, etc.**
  - For humans: only use SSO, with MFA
  - For service accounts (SAs): use AWS IAM roles for service accounts (for EKS), GCP Workload Identity (for GKE), etc.
  - Cross-provider IAM access for SAs: use identity federation
- **Review IAM policies of humans + service accounts for overly permissive privileges**

Network Perimeter

- **Review resources with public IPs and move them to private network zones, where appropriate.**
- **Firewall review: identify and eliminate ports open to the Internet (ingress 0.0.0.0/0). In particular SSH & RDP.**

! Setup automated monitoring for all of this!

Phase 1: Security Monitoring

Cloud Security Posture Management (CSPM): monitor your entire cloud environment for security violations (IAM users or user-managed GCP service account keys, public storage buckets, AWS IMDSv1, etc.). You can use

- **Cloud provider native service (AWS Security Hub/Config, Azure Security Center/Policy, GCP Security Command Center), OR**
- **Commercial 3rd party SaaS/tool, OR**
- **Open source tools (Cloud Custodian, Steampipe, …)**

Phase 1: Define Incident Response Process

- **Formalize the incident response process:**
  - **Define the phases and their activities: Detection, Analysis, Containment, Eradication, Recovery, Post-Incident**
  - **Setup repository for storing evidence**
  - **Use issue tracking system to coordinate and indicate progress**
  - **Nominate contact persons: operations, engineering, IT, customer support, legal, etc.**

Phase 1: Centralized Logging

- **You will need a source of truth for security incident investigations.**
- **Setup a central log data sink with proper access control (object storage such as AWS S3, GCP GCS, etc. might be the most suitable; SIEM is another option).**
- **As a first step, forward cloud provider management API logs to this sink (AWS CloudTrail, GCP admin activity logs, etc.)**
- **For querying this data sink, use AWS Athena, GCP BigQuery or something similar. Serverless is the goal – the only security engineer of ACME does not have (much) time for infrastructure maintenance & operations.**

Phase 1: Vulnerability Management

- **Setup vulnerability scanning for**
  - **Containers: commercial solutions, or open source based (Anchore, Clair, Trivy)**
  - **Host machines: commercial solutions, usually based on agents running on those machines. Some cloud providers also offer a solution.**
  - **Prefer continuous scanning over CI/CD based scans: because a one time scan at build time is not sufficient.**
- **Agree on patching SLAs with operations / engineering!**

Survived the first few months.
What are the next steps?

Phase 2 Activities

- **Define security baseline (secrets management, encryption policy, …)**
  - **Also reuse existing frameworks s.a. CIS benchmarks that are available for AWS, Azure, GCP, Kubernetes, Docker, etc.**
  - **Work on establishing compliance to your baseline (monitor with CSPM for violations!)**
- **Kubernetes security improvements: *securityContext* for pod definitions (*runAsUser*, *allowPrivilegeEscalation*, …) and network security policies (network perimeter inside K8S)**
- **Integrate Kubernetes security monitoring into your CSPM solution.**
- **Adopt a workflow orchestration framework for regularly executing your security tools (e.g. repository credential scans, web application scanner, ...)**
  You can use a cloud provider service (AWS Step Functions, Azure Logic Apps, GCP Cloud Workflows) or Kubernetes native frameworks (Argo Workflows, Tekton). Or a cron job in a VM ;)
- **Integrate security into the software development lifecycle (SDLC): perform security reviews in the design phase.**
- **Perform regular pen tests (at least 1x / year)**

Continue with phases 3 and 4.
Adapt and extend as required.

Some final thoughts

- **ACME is serving business customers, and some of them are saying *“we will only do business with you if you have [SOC2, ...] compliance certifications”***
  - **Compliance requirements can help you get security work done**
  - **And this is even a business driver generating more revenue!**
- **Build relationships with other departments, in particular with engineering. They should see you as a valuable support function and not as a blocker.**

Some Useful Reading Material

- **Evan Johnson: “Starting a security program at a startup”, AppSecCali 2019, Youtube Link**
- **Marco Lancini: “On Establishing a Cloud Security Program”, May 18th 2021, Blog Link**
- **Scott Piper: “AWS Security Maturity Roadmap”, January 2021, PDF Link**
- **CNCF TAG Security: Cloud Native Security Whitepaper, v2, May 2022**

Thank you for your attention.

contact@christianb.net

Picture from https://unsplash.com/photos/5eThdzpVqyE

Backup Slides

Phase 1: SSO (more of a corporate/enterprise security topic)

- **Introduce Single-Sign On (SSO) and multifactor authentication (MFA)**
  - **This really helps with employee off-boarding**
  - **Replaces static user credentials (e.g. AWS IAM users with access keys) used to access your cloud environments**
  - **Consider hardware-based authenticator as 2nd factor at least for production systems (e.g. FIDO2)**

Phase 3 Activities

- **Setup network traffic egress filtering, usually by deploying a network proxy in your VPCs (e.g. Squid)**
- **Setup automated security monitoring for log data ingested into the central log data sink (see phase 1)**
- **Harden your CI/CD infrastructure, it is a critical infrastructure component!**
- **Introduce Business Impact Assessments (BIAs) to determine component/service criticality level (based on CIA ratings). This determines how much security attention a component/service will need.**
- **Define playbooks for the most common (or expected) security incidents**
- **Introduce threat modeling and a risk management process. The CIS Risk Assessment Method (RAM) is a good starting point.**
- **Shift security left: integrate SAST tools into your CI/CD pipelines. Or run DAST tools from your security automation platform. Some examples:**
  - **Secrets scanning: gitLeaks, git-secrets, detect-secrets, TruffleHog, ...**
  - **Container vulnerability scanning tools (see phase 1)**
  - **Infrastructure code scanning: Checkov, kics, terrascan, tfsec, ...**
  - **Generic source code scanning: semgrep, ...**
  - **Web security scans: https://github.com/psiinon/open-source-web-scanners**

Phase 4 Activities

- **Setup a security training program for engineering (you should know the common problems by now)**
- **Perform game days to test your incident response procedures**
- **Hardening of virtual machine and container images:**
  - **Follow CIS benchmarks for OS level hardening**
  - **Think about using distroless container base images**
  - **Provide hardened base images that can be used by engineering**
- **Collect security related metrics and present them in a dashboard**
- **Setup a bug bounty program**
- **Provide secure-by-default building blocks to engineering: e.g. infrastructure code for network reference architecture, misuse-resistant JWT validation library, etc.**