DevOOPS Attacks

- ➔ Tokens in logs/dumps/configs/code snippets
- ➔ Pastebin, stackoverflow and similar sites
- ➔ Github, Bitbucket (gist, code, builds)
- ➔ Slack tokens in Github
- ➔ AWS credentials in dot files

DevOOPS Attacks

- ➔ Developer, Ops laptop lost (or) stolen
- ➔ Always admin on their systems
- ➔ Root key is king
- ➔ Security patches not updated

DevOOPS Attacks

- ➔ Older version software and applications
- ➔ Server hardening not done
- ➔ No standard AMI for infrastructure
- ➔ Container images available to public
- ➔ Hard coded keys in code
- ➔ Docker == root

DevOOPS Attacks

- ➔ Exposed Credentials (stolen or lost machine, commits with dot files and stack overflow)
- ➔ Vulnerable apps (app with security issues)
- ➔ Misconfiguration (lack of monitoring, iam policies, hardening)
- ➔ Insecurely configured services (s3 buckets, RDS)

DevOOPS Attacks Solution

- ➔ Move away from public github, pastebin (Gitlab, Gogs, Phabricator)
- ➔ Use SSH Keys only, Enable 2FA
- ➔ Security Audits
- ➔ Gitrob, Git Monitor
- ➔ Dumpmon, pastemon
- ➔ Osquery, OSSEC, ELK
- ➔ Patch Management

DevOOPS Attacks Solution

- ➔ Secure Authentication & Authorization
- ➔ Logging & Monitoring
- ➔ Private registry (docker registry, gcr, quay)
- ➔ Image scanning (clair, docker scan)
- ➔ rootless containers
- ➔ Isolation and segmentation (apparmor, seccomp etc)

Insecurity Scenarios

App insecurity scenario

- App has a Local File Inclusion bug
- The AWS root credentials are being used
- They are stored in a world readable file on the server
- Attacker reads the credentials and starts multiple large instances to mine bitcoins
- Victim saddled with a massive bill at the end of the month

Infra insecurity scenario

- ➔ MySQL Production database is listening on external port
- ➔ Developers work directly on production database and require SQL Management Software
- ➔ They log in using the root user of MySQL Database server and a simple password
- ➔ Attacker runs a brute force script and cracks the password, gains full access to the database

Data insecurity scenario

- ➔ Database is getting backed up regularly
- ➔ Due to performance reasons, database wasn’t encrypted when initial backups were done
- ➔ Dev team moves to newer type SSDs and doesn’t decommission older HDDs
- ➔ Attacker finds older HDD, does forensics for data recovery and sell the data for profit.

DevSecOps Playbook

n number of experiments to refine processes and automate where possible

Operating Model	Processes	Tooling
• Determine defect and feature flows for Security to funnel to distributed teams • Inventory work processes, guidelines, policies, experiments, data and tools • Identify groups, roles and skills required to support processes • Identify friction and measure speed of MTTR • Identify types of decisions • Identify metrics for measuring experiments and adapting processes	• Implement Code & Infrastructure Guidelines • Implement Rules Engineering Processes • Implement Security Defect Reporting • Implement Consulting and Requests Process • Implement Infrastructure Templates • Implement Red Team & SOC Processes • Implement Manual Staging Processes • Implement a Decisions Process • Implement an Escalation Process with clear stakeholders	• All systems should be run with API inspection available via a Security Fabric. (Systems without inspection require manual intervention.) • Implement Security Portal for feedback consolidation across security processes • Implement Case Management for Requests, Defects, and Incidents • Implement Testing framework • Implement Correlation engine • Implement foundational security controls • Integrate with core organizational systems

outcomes

• Identified opportunities to develop capacity without increasing risk to too high a level • Inventory provides information for Decisions board to help with risk decisions	• Decisions board with clear escalation path by type of decision • Ability to Communicate and Train on initial processes	• Consistent Ins/Outs of Dynamic Work with standard templates • SDE helps with reducing manual efforts • Ability to build up capacity for Stage Two

Expected Issues: Communication changes, adaptation of skills, decisions processes, expectations, audits and risk guidelines mismatch

Checklist

- ➔ Collaboration is key principle, make sure all teams involved throughout project life cycle.
- ➔ Now infrastructure is codified and version controlled. Add security checks into the code itself, and make some best practice checklist for your organisations
- ➔ Always add security monitoring & logging for each infrastructure, application you have

Checklist

- ➔ Once the code is committed to version control system, integrate your security checks and scanners using CI/CD
- ➔ Build centralised repositories and registries and look for security issues
- ➔ Document everything, It’s really important to know what’s happening
- ➔ Automate as much as possible, trust computers rather our memory

Checklist

- ➔ Secure by default, encrypt everything possible.
- ➔ SSH with keys, no root. HTTPS every where
- ➔ Secure storage, backups
- ➔ Perform red teaming activities
- ➔ Measure with the samples always, and take feedback from all teams and keep improve the process

Checklist

- ➔ DevSecOps is not one person job. Build security champions, gamification is the key for making more security champions in your organisations
- ➔ Build devsecops mindset and improve the culture, it’s one of the best hack to getting involved.
- ➔ Follow like minded people and contribute to the open source community