**Name : Bethi Roshni**

**Intern ID : 425**

**What is Threat Intelligence?**

Threat Intelligence is like knowing your enemy before they attack — in the world of computers and cybersecurity.

It means collecting information about:

* Who the hackers are
* What tools and tricks they use
* What kind of damage they cause
* How to stop them before they succeed

**TTPs Framework (Tactics, Techniques, and Procedures)**

This framework helps us understand how cyber attackers operate, step by step.

1. **Tactics** – *What the attacker wants to achieve*

2. **Techniques** – *How the attacker tries to do it*

3. **Procedures** – *Exactly how they do it*

**DevOps Threat Matrix**

**1. Reconnaissance**

Gathering information to prepare for an attack on DevOps infrastructure.

**Techniques:**

1. T1592 – Gather Victim Host Information

Identify build servers, SCM (GitHub/GitLab), or exposed tools by the subdomain brute-forcing or DNS lookup.

2. T1595 – Active Scanning

Scan for open ports on Jenkins, Docker daemons, or cloud APIs to identify vulnerable services.

3. T1596 – Search Open Sites

Use GitHub, Pastebin, or search engines to find leaked secrets or CI config files.

**Procedures:**

Use truffleHog or Gitleaks to scan public GitHub repositories for leaked secrets or tokens.

Scrape .git/config & CI pipeline definitions in public repos to discover toolchains and endpoints.

**2. Resource Development**

Preparing infrastructure, code, or tools before executing the attack.

**Techniques:**

1. T1583.001 – Acquire Infrastructure: Cloud Account

Create malicious cloud resources (e.g., fake CI/CD runners or S3 buckets).

2. T1587.003 – Develop Capabilities: Payloads

Write custom malware or implant in build scripts or libraries (e.g., postinstall in NPM).

3. T1588.002 – Obtain Code Signing Certificates

Set up GitHub repos or containers that mimic legitimate projects to gain trust.

**Procedures:**

Publish a typosquatted NPM package that contains malicious installation logic.

Set up a fake GitHub repo that mirrors a popular CI template & includes malicious YAML or Dockerfiles.

**3. Initial Access**

Breaking into DevOps environments like pipelines or source control.

**Techniques:**

1. T1078 – Valid Accounts

Use leaked or phished credentials to access SCM, pipelines, or cloud consoles.

2. T1190 – Exploit Public-Facing Applications

Exploit CI plugins, exposed Jenkins consoles, or misconfigured API endpoints.

3. T1566.001 – Spearphishing Attachment

Send crafted CI-related documents with malicious macros or scripts.

**Procedures:**

Use a leaked GitHub PAT to modify a repository’s GitHub Actions workflow for code execution.

Send a spoofed email claiming a failed build, including a fake link to a “log viewer” containing malware.

**4. Execution**

Running attacker-controlled code inside DevOps infrastructure.

**Techniques:**

1. T1059 – Command and Scripting Interpreter

Use PowerShell, Bash, or Python inside CI jobs or remote scripts.

2. T1204.002 – User Execution: Malicious File

Users unknowingly run a malicious artifact from the pipeline.

3. T1651 – Cloud Administration Command

Abuse cloud-native tools (e.g., AWS SSM, Azure RunCommand) to run payloads on VMs.

**Procedures:**

Inject a curl | bash payload into .gitlab-ci.yml or GitHub Actions YAML.

Use az vm run-command or aws ssm send-command to launch malware on compromised cloud instances.

**5. Persistence**

Ensuring continued access inside CI/CD or related infrastructure.

**Techniques:**

1. T1547.001 – Registry Run Keys/Startup Folder

Persist malware by altering Windows startup entries on self-hosted runners.

2. T1053 – Scheduled Task/Job

Schedule malicious scripts via CRON on Linux CI agents.

3. T1505.003 – Web Shell

Place a web shell in artifact storage or internal portals for access.

**Procedures:**

Create a scheduled CRON job on a CI runner that downloads and executes a reverse shell on reboot.

Add a persistence mechanism into a shared library or script (e.g., build.sh) used across jobs.

Great! Here’s the continuation of the MITRE ATT&CK tactics for DevOps environments, each with 3 techniques and 2 procedures with descriptions, starting from Tactic 6.

**6. Privilege Escalation**

Gain higher-level permissions within DevOps tools or cloud infrastructure.

**Techniques:**

1. T1068 – Exploitation for Privilege Escalation

Exploit vulnerabilities in CI containers or host systems to gain root/admin.

2. T1543.003 – Create or Modify System Process (systemd services)

Set malicious systemd services to auto-execute payloads on CI/CD runners.

3. T1611 – Abuse Elevation Control Mechanism (e.g., IAM misconfigurations)

Exploit weak IAM policies in cloud CI setups to elevate privileges.

**Procedures:**

Exploit Docker socket exposure (/var/run/docker.sock) in CI runner to escalate to host root.

Modify IAM role binding to allow admin access across cloud CI jobs.

**7. Defense Evasion**

Avoid detection by security tools or human reviewers in DevOps pipelines.

**Techniques:**

1. T1562.001 – Disable or Modify Tool Logging

Disable cloud logs (CloudTrail, Activity Logs) or delete CI job history.

2. T1027 – Obfuscated Files or Information

Encode malicious commands using base64 in YAML or scripts.

3. T1070.004 – File Deletion

Clear logs or artifacts after execution to avoid detection.

**Procedures:**

Use rm -rf to delete CI logs and artifacts post-execution.

Encode payload using base64 in GitHub Actions and decode at runtime to hide malicious intent.

**8. Credential Access**

Steal or abuse secrets from source code, environment variables, or cloud.

**Techniques:**

1. T1552.001 – Unsecured Credentials

Access .env, .npmrc, or pipeline secrets in repositories.

2. T1555.003 – Credentials in Log Files

Find secrets in CI output or verbose debug logs.

3. T1528 – Abuse Access Tokens

Use leaked OAuth tokens, AWS keys, or GitHub PATs.

**Procedures:**

Use automated secret scanners to detect credentials hardcoded in repos.

Monitor CI job logs for accidental printing of environment variables.

**9. Discovery**

Explore internal systems and environments within CI/CD pipelines.

**Techniques:**

1. T1083 – File and Directory Discovery

Search for keys, configs, or scripts in mounted paths.

2. T1087 – Account Discovery

Identify cloud accounts, user roles, or CI agents.

3. T1135 – Network Share Discovery

Discover connected services, build agents, or internal APIs.

**Procedures:**

List files in home and secrets directories during pipeline execution (ls -al ~/ ~/.aws/).

Use env, whoami, or curl to inspect CI/CD environment and network metadata.

**10. Lateral Movement**

Move between systems in the DevOps ecosystem.

**Techniques:**

1. T1021.001 – Remote Services: SSH

Use exposed SSH keys from CI/CD runners to access other systems.

2. T1550.002 – Use of Stolen Cloud Tokens

Reuse AWS/GCP tokens to access unrelated cloud resources.

3. T1072 – Remote Desktop Protocol

Access GUI-based consoles for CI/CD tools or cloud.

**Procedures:**

SSH into internal build servers using leaked private keys from repo history.

Use compromised AWS session token from CI secrets to assume another IAM role.

**11. Collection**

Gather sensitive data for later use or exfiltration.

**Techniques:**

1. T1005 – Data from Local System

Collect build artifacts, logs, secrets, config files.

2. T1074 – Data Staged

Zip or tar sensitive data for exfiltration.

3. T1113 – Screen Capture

(For GUI-based tools) Take screenshots from remote desktops.

**Procedures:**

Collect environment files (.env, .git-credentials, .docker/config.json) from CI agents.

Zip secrets and move them to a staging directory inside the build agent.

**12. Command and Control**

Maintain remote access to CI/CD systems.

**Techniques:**

1. T1071 – Application Layer Protocol

Use HTTPS for communication with command and control (C2) servers.

2. T1105 – Ingress Tool Transfer

Download second-stage tools during build process.

3. T1095 – Non-Application Protocol (e.g., DNS tunneling)

Use alternative protocols to evade detection.

**Procedures:**

Use curl or Invoke-WebRequest in CI jobs to pull shell scripts from a C2.

Establish reverse shell that communicates over HTTPS with attacker’s server.

**13. Exfiltration**

Extract sensitive data outside of the DevOps environment.

**Techniques:**

1. T1041 – Exfil over C2 Channel

Use established C2 connection to send stolen files.

2. T1567.002 – Exfiltration to Cloud Storage

Upload data to attacker-controlled S3 bucket or Google Drive.

3. T1048 – Exfiltration Over Alternative Protocol

Use FTP, SCP, or rsync to extract files quietly.

**Procedures:**

Use AWS CLI in a pipeline to upload stolen files to an attacker’s S3 bucket.

Transfer collected data via scp to external server controlled by the attacker.

**14. Impact**

Disrupt, damage, or destroy DevOps operations or reputation.

**Techniques:**

1. T1486 – Data Encrypted for Impact

Encrypt build artifacts, secrets, or critical deployment configs.

2. T1499 – Endpoint DoS

Overwhelm CI/CD runner resources by flooding job queue.

3. T1491.002 – Defacement

Modify CI dashboards or project README to deface reputations.

**Procedures:**

Add ransomware logic into build.sh to encrypt deployment binaries.

Push endless fake PRs or scheduled jobs to crash CI/CD systems.