📄 Proof of Concept (PoC) : Cloud Storage Threat Matrix (Microsoft)

Intern Name: Aditya Yashwant Borade **Intern ID : 131**
Project Title: Cloud Storage Threat Matrix (Microsoft)
Organization: Digisurakhsha Foundation

🔰 Objective

The goal of this Proof of Concept is to explore how attackers target **cloud storage services** using various tactics and techniques as defined in Microsoft's Cloud Storage Threat Matrix. This research aligns with the larger topic of **Threat Intelligence**, focusing on identifying, understanding, and mitigating cloud-specific threats.

🛡️ Part 1: All Tactics in Microsoft's Cloud Storage Threat Matrix

The Cloud Storage Threat Matrix aligns closely with the **MITRE ATT&CK framework** and is structured around **tactics** — the attacker's **goals during an operation**. Below are the tactics relevant to cloud storage:

Tactic	Description
Reconnaissance	Gathering information about cloud storage (e.g., buckets, blob containers)
Initial Access	Gaining a foothold in a cloud environment
Execution	Running malicious code or scripts in the cloud
Persistence	Maintaining access in cloud storage services
Privilege Escalation	Gaining higher-level permissions
Defense Evasion	Avoiding detection and security tools
Credential Access	Stealing or accessing login information
Discovery	Identifying services, configurations, and data
Lateral Movement	Moving from one resource to another
Collection	Gathering sensitive data
Exfiltration	Sending stolen data out of the cloud
Impact	Damaging, deleting, or encrypting data

💉 Part 2: Techniques (At least 3)

✅ Technique 1: Unsecured Cloud Storage Buckets

- **Tactics:** ***Reconnaissance, Initial Access***

🔍 **Description:**

Many organizations use public cloud providers like Amazon S3, Azure Blob Storage, or Google Cloud Storage to host assets. If these storage buckets are misconfigured to allow anonymous (unauthenticated) access, attackers can find and browse their contents without credentials.

Using automated scanners or brute-forcing bucket names from wordlists, attackers can discover open buckets and extract sensitive files such as:

- Environment configs (.env, config.json)
- Backups (.bak, .zip)
- Source code or internal documentation
- Personally Identifiable Information (PII)

⚠️ **Real-World Context:**

Numerous data breaches have occurred due to public S3 buckets, including the Accenture, Booz Allen, and Tesla cases.

🛡️ **Defense:**

- Apply least privilege access (private by default)
- Enable logging and monitoring
- Use CSPM (Cloud Security Posture Management) tools to detect misconfigurations

---

✅ **Technique 2: Exploiting Shared Access Signature (SAS) Tokens**

- **Tactic: *Defense Evasion***

🔍 **Description:**

Azure's SAS tokens allow time-limited and permission-specific access to blob storage and other services. However, if a SAS token:

- is leaked in GitHub repositories, emails, or logs
- has an unnecessarily long expiration
- lacks IP or protocol restrictions

...then an attacker can silently access or modify storage objects without triggering standard IAM auditing.

⚠️ **Real-World Context:**

Developers often hardcode SAS tokens into apps or scripts. A threat actor with access to that token can access the data without triggering traditional identity-based logging or MFA protections, making it an effective evasion mechanism.

🛡️ **Defense:**

- Set short expiry times
- Use signed user delegation SAS instead of account SAS
- Restrict by IP and HTTPS-only
- Monitor for token usage in logs

✅ **Technique 3: Cloud Credential Harvesting**

- **Tactic: *Credential Access***

🔍 **Description:**

Once attackers gain access to a cloud storage bucket, they search for secrets left behind—often due to poor developer hygiene. These include:

- .env files with API keys or passwords
- .aws/credentials or .azure/config files
- OAuth tokens, Slack tokens, or SSH keys
- Service principal credentials

By harvesting these credentials, attackers can:

- Elevate privileges in cloud accounts
- Access other services (CI/CD, DBs, internal tools)
- Bypass MFA when service credentials are used

⚠️ **Real-World Context:**

Hardcoded secrets are a major source of compromise in supply-chain attacks. Threat actors like LAPSUS$ have used stolen cloud credentials to infiltrate multiple organizations.

🛡️ **Defense:**

- Never store secrets in cloud storage; use Key Vault, Secrets Manager, or Parameter Store
- Use secret scanning tools (e.g., GitGuardian, TruffleHog)
- Implement least privilege access for service accounts

✅ **Technique 4: Enumeration of Storage Services**

- **Tactic: *Discovery***

🔍 **Description:**

After gaining some access, attackers attempt to map the environment to discover:

- Existing storage accounts

- Containers or buckets
- Naming patterns
- Access control policies
- File share mounts (Azure Files, AWS EFS)

They may use tools like:

- az storage account list
- PowerShell-based MicroBurst scripts
- Internal APIs or misconfigured DNS names

This helps them plan next moves, like privilege escalation or exfiltration.

⚠️ **Real-World Context:**

APT groups and Red Teams often enumerate cloud resources to find less-monitored storage containers or shadow infrastructure.

🛡️ **Defense:**

- Audit management plane access
- Use role-based access control (RBAC) tightly
- Monitor for unusual enumeration via logging (e.g., AzureActivity, AWS CloudTrail)

✅ **Technique 5: Data Staging in Storage Accounts**

- **Tactic: *Collection***

🔍 **Description:**

Before exporting large datasets (exfiltration), attackers may stage files in temporary storage they control (within or outside the victim's account). For example:

- Zip sensitive files into one blob
- Upload to a private container
- Schedule exfiltration via a script or API

This technique helps avoid detection during data access and allows attackers to control timing and bandwidth usage of exfiltration events.

⚠️ **Real-World Context:**

Staging is commonly used in APT operations, where long-term persistence and stealth are more important than speed.

🛡️ **Defense:**

- Monitor storage write operations and large data movements

- Enable DLP (Data Loss Prevention) policies
- Alert on new container creations

✅ **Technique 6: Deleting or Encrypting Storage Data (Impact)**

- **Tactic: *Impact***

🔍 **Description:**

Attackers may intentionally delete or encrypt files in cloud storage buckets as part of a ransomware campaign or to disrupt operations. Since storage buckets often hold:

- Backup files
- Web application assets
- Documents or datasets

Deleting or corrupting them can cause critical outages or business continuity failures.

Some ransomware groups now directly target cloud-native storage using APIs, rather than endpoints.

⚠️ **Real-World Context:**

In 2022, a cloud ransomware variant targeted Amazon S3 buckets using stolen IAM tokens.

🛡️ **Defense:**

- Use soft-delete and versioning in cloud storage
- Implement immutable blob protection (e.g., WORM policies)
- Regularly test disaster recovery and backup restore processes

---

⚙️ **Part 3: Procedures (At least 2)**

🔧 **Procedure 1: Discover and Access a Misconfigured Azure Blob Storage Container**

Goal: Simulate how a threat actor may discover and access an open cloud blob container.

Steps:

1. Open command line or PowerShell.
2. Use a tool like **Microburst** or **AZScanner** to scan for open containers.
3. Invoke-EnumerateAzureBlobs -Base wordlist.txt
4. Identify open containers allowing anonymous access.
5. Use browser or Azure CLI to download/view contents.
6. az storage blob list --container-name publiccontainer --account-name myaccount --output table

Result: If the container is public, data can be listed and downloaded without authentication.

🔧 Procedure 2: Exploiting a Leaked Azure SAS Token

Goal: Show how an exposed SAS token can be used to access protected data.

Steps:

1. A developer mistakenly commits a **SAS token** in GitHub:
2. https://myblob.blob.core.windows.net/containername/blob.txt?<sas-token>
3. The attacker copies the URL and accesses the blob directly using a browser or curl:
4. curl -O "https://myblob.blob.core.windows.net/containername/blob.txt?<sas-token>"
5. Since the SAS token has read privileges, the file is downloaded without login.

Result: Confidential files are accessed using the shared token.

📌 Mapping of Tactic → Technique → Procedure

Tactic	Technique	Procedure
Reconnaissance	Unsecured Buckets	Scanning blob containers using tools
Initial Access	Unsecured Buckets	Anonymous access to blobs
Defense Evasion	Leaked SAS Tokens	Accessing protected blobs via token
Credential Access	Credential Harvesting	Searching for .env, .aws in blobs

---

🔒 Suggested Mitigations

Threat	Mitigation
Public blob access	Enable private access only, disable anonymous access
SAS token misuse	Use short expiry, restrict IPs, monitor logs
Credential leaks	Never store credentials in storage buckets, use Key Vault

---

📚 References

- Microsoft Cloud Threat Matrix: Microsoft Docs
- MITRE ATT&CK for Cloud: https://attack.mitre.org/

- Azure Storage SAS tokens: https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview
- Tools: Microburst, AZScanner

📌 Conclusion

This PoC demonstrates how attackers can exploit **cloud storage misconfigurations and weak access controls** using well-known techniques. By understanding tactics like **reconnaissance, defense evasion, and credential access**, security teams can better detect, prevent, and respond to cloud threats. Threat Intelligence plays a crucial role in defending cloud environments, and frameworks like the Microsoft Cloud Storage Threat Matrix provide a clear roadmap for threat analysis.