BLOCKSEC

Security Testing Report for MegaETH's Web and Cloud Infrastructure

Date: March 17, 2026 **Version:** 1.0

Contact: contact@blocksec.com

Contents

Report Manifest

Item	Description
Client	MegaETH
Target	MegaETH's Web and Cloud Infrastructure

Version History

Version	Date	Description
1.0	March 17, 2026	First release

Signature

Digitally signed by: BlockSec (https://blocksec.com)

SHA1 digest of public key: BAB106D3708BC6DE0963CE05E9478C24B326AE1C

Date: 2026-03-18 07:50:16+0000

About BlockSec BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at Email, Twitter and Medium.

Chapter 1 Introduction

1.1 About Target Contracts

Information	Description
Type	Cloud Penetration Testing
Approach	Semi-automatic and manual verification

MegaETH's Web and Cloud Infrastructure on Google Cloud Platform (GCP), its externally exposed access channels, and its core architectural design and implementation collectively introduce three critical security risks. Each risk maps to a specific penetration-testing focus:

- **Cloud compromise enabling lateral movement and sensitive asset exposure**. MegaETH relies heavily on GCP for development and deployment. If a key cloud component, such as a jump server or cloud VM, is compromised, an attacker could exploit misconfigured IAM roles, stolen credentials, or unpatched instance vulnerabilities to move laterally within the cloud environment. This may lead to unauthorized access to high-value resources, including private keys and transaction data, and disruption of overall network operations.
- **Expanded external attack surface from internet-facing assets**. MegaETH exposes multiple external access channels, including public services, domains, and IP addresses. This broad footprint increases the likelihood of initial compromise through weaknesses in internet-facing endpoints, such as RPC services, observability stacks, web applications, subdomain configurations, or mismanaged certificates and proxy layers. Any exploitable flaw in these assets could provide a direct entry path into the cloud environment, making attack-surface mapping and remote-exploitation testing essential.
- **Architectural trust-boundary and access-control weaknesses in the deployed system**. MegaETH's core architectural design and implementation determine how trust boundaries and privileges are defined and enforced across the deployed environment. If these boundaries are weak or inconsistently implemented, for example through insufficient separation between admin and operations planes and runtime services, overly permissive service-to-service access paths, insecure secret or key-handling flows, or limited defense in depth, then an attacker with any foothold could escalate access beyond what individual vulnerabilities would normally allow. Penetration testing must therefore validate that segmentation, authentication and authorization paths, and privilege scoping remain robust under adversarial conditions.

Accordingly, the testing targets emphasize simulating real-world attacker behavior after an initial breach, such as beginning from a compromised jump server or cloud VM, to determine whether and how unauthorized access can propagate across the cloud infrastructure. This approach validates defenses against lateral movement techniques, including abuse of IAM misconfigurations, credential theft, and exploitation of unpatched instances. The objective is to identify gaps in network segmentation, access controls, and threat-detection mechanisms, and to confirm that sensitive cloud resources remain protected even after an attacker gains an initial foothold.

The engagement was conducted over a 12-day period from February 23, 2026 to March 06, 2026. The scope included three Google Cloud VMs (two RPC sync nodes and one OP Stack node), as well as 32 web domains and 23 externally exposed IP addresses.

1.2 Disclaimer

The scope of this security testing is limited to the code mentioned in Section 1.1. This penetration test does not guarantee the discovery of all security issues in the target system, the assessment results do not guarantee the absence of any other security vulnerabilities. This report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. Because no single penetration test can be completely comprehensive, we always recommend conducting independent penetration tests and implementing a public bug bounty program to ensure the security of the network infrastructure.

1.3 Methodology

To ensure a comprehensive security evaluation of the target systems, BlockSec consultants follow multiple industry standard methodologies and frameworks, including the Penetration Testing Execution Standard (PTES) [1], the MITRE ATT&CK framework [2], and the Open Web Application Security Project (OWASP) [3].

1.3.1 Overall Workflow

Our engagement follows a structured penetration testing lifecycle.

First, we conduct reconnaissance to identify targets and information that would be valuable to a real adversary. We use open source intelligence (OSINT) to understand the target's services, mission, and publicly available data that could be leveraged for initial access. In parallel, we map externally reachable assets and enumerate machines and services within the accessible network boundary. This builds a clear view of the attack surface and trust relationships across the environment.

Next, we perform vulnerability analysis based on the reconnaissance results to identify likely attack vectors. We validate these vectors through targeted exploitation, then develop and execute an attack plan aligned with PTES phases and MITRE ATT&CK tactics. After obtaining an initial foothold on cloud virtual machines, we assess post-compromise behavior, focusing on lateral movement using acquired VM privileges. We then apply privilege escalation techniques to determine whether higher-privilege access can be achieved and whether that access could enable broader impact. This workflow evaluates both individual weaknesses and realistic attack chains.

In addition to manual testing, we review high-risk vulnerabilities identified through *Bitdefender* scan results. Findings are filtered to prioritize issues with strong exploitability during

[1] http://www.pentest-standard.org/index.php/Main_Page

[2] https://attack.mitre.org/

[3] https://owasp.org/

adversarial operations. These vulnerabilities are treated as high priority because they often represent practical entry points or escalation paths. In an incident scenario, quickly identifying these risk areas helps defenders determine plausible attack routes, contain threats efficiently, and deploy appropriate mitigations.

1.3.2 Security Testing Coverage

Taking into account the MegaETH's industry context and the security-testing resources provided by the project, this assessment was conducted across the following three areas:

- **Web Asset Security**: Conducting in-depth web security testing on the provided Internet-facing assets, including the listed domains as well as assets discovered through our own enumeration.
- **Host Security**: Performing host-side security reviews of the provided VM instances, including comprehensive security checks within the VMs.
- **GCP Security**: Assessing the security of the provided GCP environment and accounts, including privilege-escalation paths and attempts at unauthorized resource access.

In parallel, we also collected **threat intelligence** related to MegaETH, including project-level and personnel-related intelligence. These results were delivered separately to MegaETH and are not included in this report due to privacy considerations.

1.3.3 Applied Frameworks and Checkpoints

- **Penetration Testing Execution Standard (PTES)**. We use PTES as the primary methodology because it models the full sequence of actions a real adversary would take to compromise a network. PTES defines an end-to-end penetration testing lifecycle, including pre-engagement interactions, intelligence gathering, threat modeling, vulnerability analysis, exploitation, post-exploitation, and reporting.
- **MITRE ATT&CK**. We align our testing with MITRE ATT&CK to ensure the assessment reflects real-world attacker behavior. ATT&CK is a globally accessible knowledge base of adversary tactics and techniques derived from observed intrusions and widely adopted for threat modeling and security validation. Each testing activity is mapped to relevant ATT&CK tactics and techniques to ensure comprehensive coverage and high-fidelity evaluation.
- **OWASP Top 10**. We apply the `OWASP Top 10` to assess the security posture of the target's web-facing components in accordance with industry best practices. OWASP Top 10 represents broad consensus on the most critical risks to web applications and serves as a standard reference for identifying and validating these issues. The categories include:
  - A01 - Broken Access Control
  - A02 - Cryptographic Failures
  - A03 - Injection
  - A04 - Insecure Design
  - A05 - Security Misconfiguration
  - A06 - Vulnerable and Outdated Components
  - A07 - Identification and Authentication Failures

* A08 - Software and Data Integrity Failures
* A09 - Security Logging and Monitoring Failures (which includes insufficient detection and response capabilities)
* A10 - Server-Side Request Forgery (SSRF)

Note *The checkpoints listed above represent the primary assessment focus. Additional checkpoints may be applied during security testing as needed, depending on the project's specific functionality and implementation.*

1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [4] and Common Weakness Enumeration [5]. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.

Table 1.1: Vulnerability Severity Classification

Impact \ Likelihood	*High*	*Low*
High	High	Medium
Low	Medium	Low

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following five categories:

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Partially Fixed** The item has been confirmed and partially fixed by the client.
- **Fixed** The item has been confirmed and fixed by the client.

[4] https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[5] https://cwe.mitre.org/

Chapter 2 Findings

In total, we found **six** potential security issues.

- High Risk: 1
- Medium Risk: 4
- Low Risk: 1

ID	Severity	Description	Category	Status
1	High	Publicly accessible ClickHouse panel with insufficient authentication	Security Issue	Fixed
2	Medium	Hardcoded Lark webhook URL in plaintext within GCP VM configuration	Security Issue	Fixed
3	Medium	Exposed origin server IP address due to CDN protection circumvention	Security Issue	Fixed
4	Medium	Uncleared shell command history exposing sensitive credentials on GCP VMs	Security Issue	Fixed
5	Medium	Stale GCP service account token stored in plaintext on disk	Security Issue	Fixed
6	Low	Hardcoded plaintext Loki password in vector service configuration file	Security Issue	Fixed

The details are provided in the following sections.

2.1 Security Issue

2.1.1 Publicly accessible ClickHouse panel with insufficient authentication

Severity High

Status Fixed

Description The `ClickHouse` web panel is accessible from the internet and protected exclusively by username and password authentication. If an attacker successfully compromises these credentials, they can gain full control over the `ClickHouse` database, leading to potential data exfiltration, unauthorized data manipulation, or service disruption. Moreover, the lack of multi-factor authentication (`MFA`) on the management interface substantially increases the risk of unauthorized access.

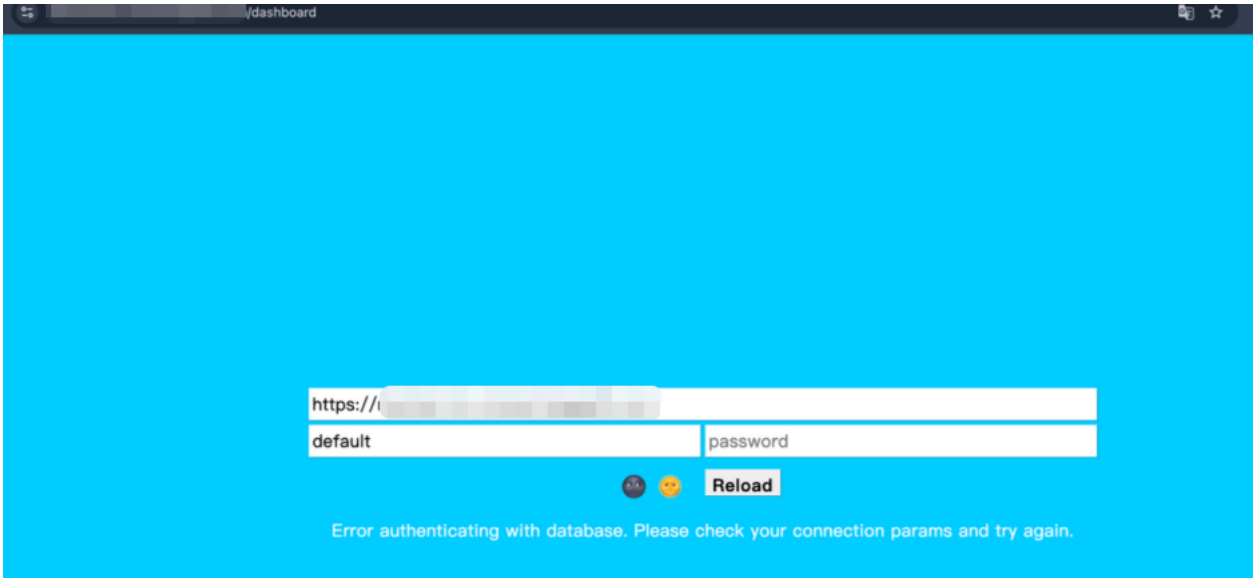

Figure 2.1: ClickHouse Dashboard

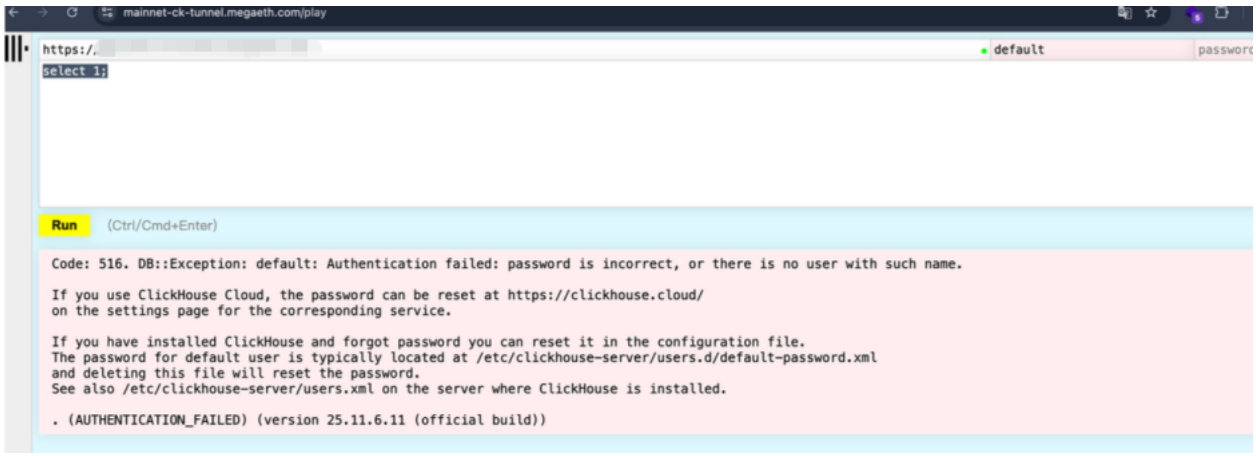

Figure 2.2: ClickHouse Index Page

Impact The access configuration of the `ClickHouse` web panel contains security weaknesses that render it highly susceptible to brute-force attacks and social engineering campaigns aimed at credential acquisition. Successful exploitation could result in unauthorized access to the `ClickHouse` database, leading to data exfiltration or service disruption.

Suggestion For the `ClickHouse` administrative console, it is recommended to enable multi-factor authentication (`MFA`) to strengthen security and restrict access exclusively to authorized `IP` addresses or designated network segments. Additionally, regularly reviewing and rotating user credentials, along with continuous monitoring of authentication activities, can further mitigate the risk of compromise.

2.1.2 Hardcoded Lark webhook URL in plaintext within GCP VM configuration

Severity Medium

Status Fixed

Description A `Lark` webhook URL was discovered in plaintext within a configuration file on

a Google Cloud VM instance. This practice introduces a critical security vulnerability: if an attacker gains access to the URL, they can send malicious requests to Lark, potentially leading to data exfiltration, service disruption, or other security incidents.

```
root@mnet-fra-rpc-2:/opt/block_monitor_agent# cat config.py
SERVER_ROLE = "mnet-fra-rpc-2"

CAST_RPC = "http://127.0.0.1:9547"

RPC_METRICS_URL = "http://127.0.0.1:8002"

PORT = 18081

LARK_WEBHOOK = "[redacted]"
```

Figure 2.3: Lark Webhook URL leaked

Impact If an attacker gains access to the URL, they can send malicious requests to Lark, potentially leading to data exfiltration, service disruption, or other security incidents.

Suggestion It is strongly recommended to immediately remove the Lark webhook URL from the configuration file and instead store and retrieve it using environment variables or a secure secrets management solution. Additionally, IP allowlisting should be implemented to restrict access to the webhook URL exclusively to trusted sources. The webhook URL should also be rotated periodically to maintain its security posture, and relevant logs must be monitored continuously to detect any anomalous or unauthorized activity.

Feedback from the project The Lark webhook URL is configured with notification-only permissions, which prevents attackers from directly retrieving sensitive data or performing malicious operations through it.

Clarification from BlockSec The project has addressed this issue by implementing IP allowlisting on the webhook URL, restricting access to the following trusted IP addresses: ***.***.113.120, ***.***.117.129, ***.***.18.72, and ***.***.48.217.

2.1.3 Exposed origin server IP address due to CDN protection circumvention

Severity Medium

Status Fixed

Description The IP addresses ***.***.116.96 and ***.***.114.186 can be used to circumvent CDN protection and access the origin server directly.

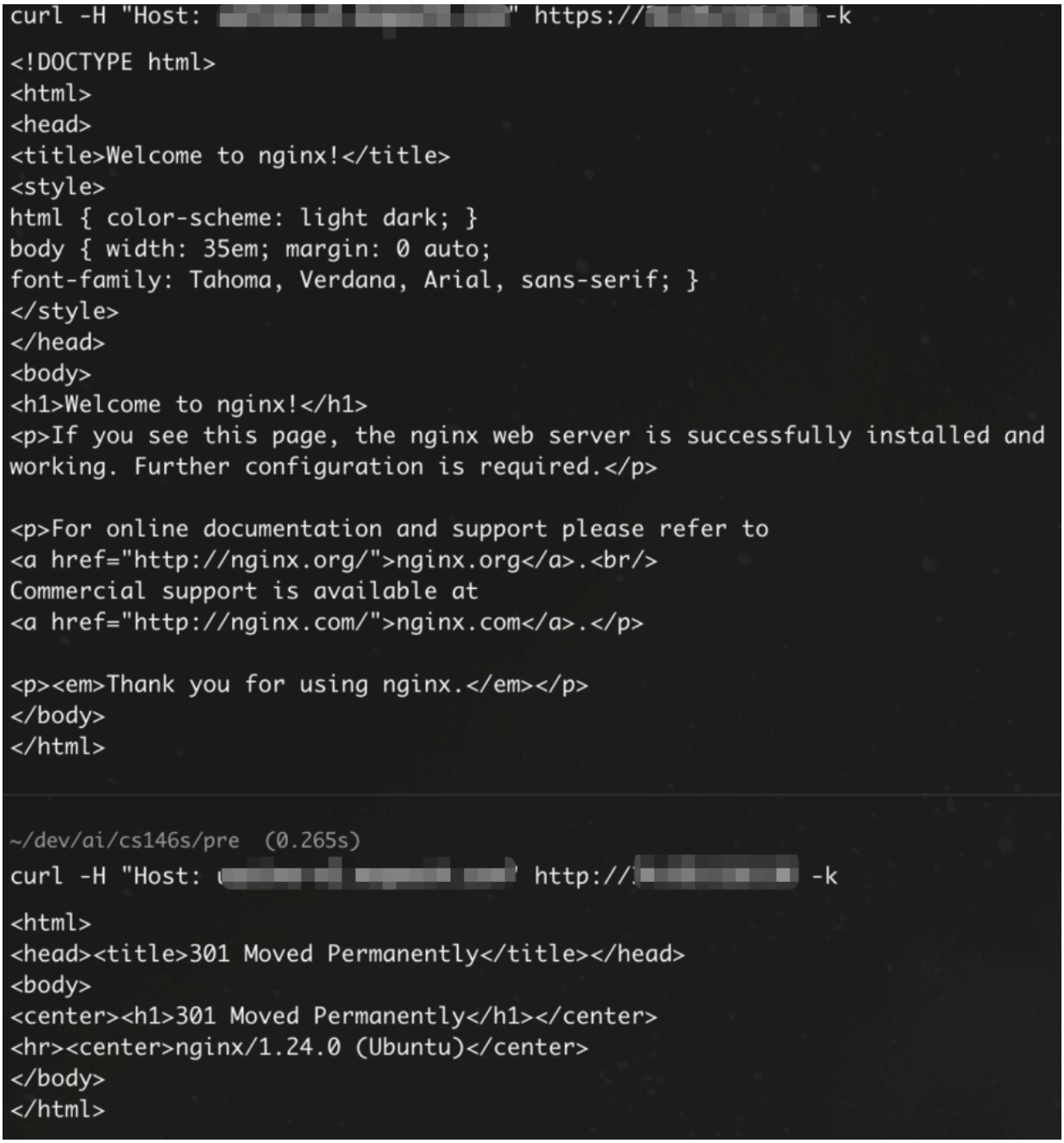

Figure 2.4: CDN circumvention 1

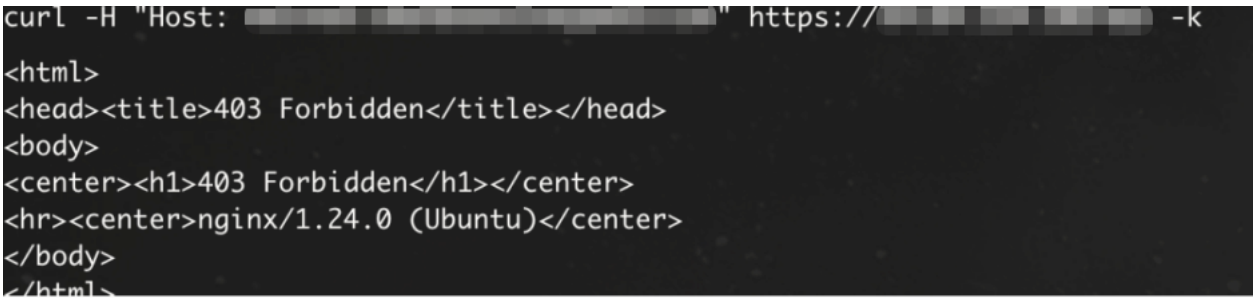

Figure 2.5: CDN circumvention 2

Impact Malicious actors can directly access the IP resources over the internet, circumventing CDN protection. If attackers successfully circumvent the CDN's security controls, they may gain direct access to services hosted on these IP addresses, potentially leading to sensitive data exposure, unauthorized operations, or service disruption. Furthermore, these exposed IPs could be leveraged as footholds for additional attacks, thereby significantly increasing the overall attack surface and security risk.

Suggestion The access controls for these two IP resources must be reassessed and hardened. The services associated with them should be migrated to a protected environment and placed behind a strictly enforced CDN security policy to prevent direct exposure to the internet.

Clarification from BlockSec The project has added CDN protection to the domains associated with these IP addresses. However, the public IP addresses themselves remain directly accessible due to business requirements. It should be noted that if these public IP addresses are obtained by an attacker, the risks described above still apply.

2.1.4 Uncleared shell command history exposing sensitive credentials on GCP VMs

Severity Medium

Status Fixed

Description During the security assessment of the three provided Google Cloud VM instances, it was identified that command history records from previously executed commands had not been cleared on any of the instances. This oversight may lead to the exposure of sensitive information, such as credentials, API keys, or internal system details embedded in command-line arguments or scripts.

Impact Malicious attackers, by collecting and analyzing these command history records, may obtain critical insights into system configurations, installed software, network topology, and other sensitive operational details. Such information can be leveraged to identify potential attack vectors and craft targeted exploitation strategies. Furthermore, embedded sensitive data (e.g., credentials or access tokens) may be directly abused to gain unauthorized access or conduct further malicious activities.

Suggestion It is recommended to immediately purge the command history on all instances and implement an automated, periodic cleanup mechanism to prevent the accumulation of command-line records in the future.

2.1.5 Stale GCP service account token stored in plaintext on disk

Severity Medium

Status Fixed

Description On the host mnet-***-***-0, the account m***.yang@megaeth.com was identified in the directory /home/ubuntu/.config/gcloud/legacy_credentials/, and its associated historical access token was successfully retrieved, though the token has expired and is no longer valid for accessing Google Cloud resources.

Impact Malicious attackers could exploit the compromised, expired token to conduct social engineering attacks (e.g., impersonating legitimate services or support personnel) to trick the user into regenerating a new, valid token. Additionally, attackers may attempt to leverage artifacts from the expired token (e.g., refresh tokens, session identifiers, or credential patterns) in brute-force or credential-stuffing attacks to obtain valid authentication tokens or other sensitive information.

Suggestion Expired tokens and unused user account credentials should be promptly purged, and multi-factor authentication (`MFA`) must be enforced for all accounts to enhance security. Additionally, it is recommended to regularly review and rotate tokens to mitigate potential security risks.

2.1.6 Hardcoded plaintext Loki password in vector service configuration file

Severity Low

Status Fixed

Description During a security inspection of Docker on the host `mnet-***-***-0`, a plaintext password was identified in a `Loki` configuration file (`/opt/vector/vector.toml`).

Impact The exposed plaintext `Loki` password could be leveraged by an attacker to gain unauthorized access and perform malicious operations, potentially leading to data exfiltration, service disruption, or other security incidents. Furthermore, the attacker may use this credential in brute-force or password-spraying attacks against other related systems or accounts, thereby expanding the attack surface and escalating the impact.

Suggestion Storing passwords in plaintext is insecure during security assessments. If the `Loki` configuration cannot function properly using encrypted or obfuscated credentials, it is strongly recommended to either rotate the password regularly or manage it via environment variables (alternatively, consider leveraging secure secret management solutions such as Vector Secrets). Additionally, a credential reuse audit should be conducted to ensure that this plaintext password is exclusively used for the specific Loki functionality and is not shared across other systems or services.


BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS