BLOCKSEC

Security Testing Report for MegaETH's Web and Cloud Infrastructure

Date: November 27, 2025 **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	November 27, 2025	First release

Signature

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

SHA1 digest of public key: BAB106D3708BC6DE0963CE05E9478C24B326AE1C

Date: 2025-12-04 10:42:46+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 14-day period from October 11, 2025 to October 24, 2025. The scope included three GCP virtual machines (one sequencer node, one RPC sync node, and one observability node), as well as 25 web domains and six 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.

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

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

[3] https://owasp.org/

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 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.

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

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

- **Partially Fixed** The item has been confirmed and partially fixed by the client.
- **Fixed** The item has been confirmed and fixed by the client.

Chapter 2 Findings

In total, we found **six** potential security issues. Besides, we have **two** recommendations.

- High Risk: 3
- Medium Risk: 2
- Low Risk: 1
- Recommendation: 2

ID	Severity	Description	Category	Status
1	High	Insecure SSH configuration	Host security	Fixed
2	High	Insecure third-party dependency: Log4j	Host security	Confirmed
3	High	Insecure third-party dependency: Redis	Host security	Confirmed
4	Medium	Exposed management interface	Web Asset Security	Fixed
5	Medium	Leaked GCP service account private keys	GCP security	fixed
6	Low	Leaked build and compliation information	Web Asset Security	Fixed
7	-	Enhance threat intelligence integration and EASM coverage	Recommendation	Fixed
8	-	Apply pending security patches	Recommendation	Confirmed

The details are provided in the following sections.

2.1 Security Issue

2.1.1 Insecure SSH configuration

Severity High

Status Fixed

Description Within the assessment scope, five Linux servers expose SSH services directly to the Internet. These services allow username and password authentication with root login enabled. If an attacker compromises valid credentials, the affected servers are at high risk of unauthorized access and takeover. In addition, SSH is running on the default port 22, increasing exposure to automated scanning, service fingerprinting, and brute-force attempts.

Server	Port	SSH Login
..9.101	22	Login with username and password, root login enabled
..150.69	22	Login with username and password, root login enabled
..87.3	22	Login with username and password, root login enabled
..99.207	22	Login with username and password, root login enabled
..184.126	22	Login with username and password, root login enabled

Impact This Internet-facing SSH configuration lacks sufficient hardening and is susceptible to reconnaissance and credential-based attacks. Successful exploitation could lead to full

compromise of the affected Linux servers and provide a foothold for further lateral movement within the environment.

Suggestion Disable password-based SSH authentication and require key-based authentication. Disable direct root login and enforce least-privilege administrative access. Restrict SSH exposure through network-level controls such as a VPN or bastion host, rather than relying on port changes alone.

Feedback from the project The following measures have been implemented:

- All SSH services previously exposed to the public internet have been removed and replaced with access via JumpServer and SecureLink.
- All servers have been brought under centralized management and are now subject to unified auditing and monitoring.

2.1.2 Insecure third-party dependency: Log4j

Severity High

Status Confirmed

Description The target environment uses Apache Log4j 1.2.17. This version is not directly affected by the *Log4Shell* vulnerability (CVE-2021-44228), which impacts Log4j 2.x versions 2.0-beta9 through 2.14.1. However, Log4j 1.2.17 belongs to the legacy 1.x line that reached end of life in 2015 and is no longer supported by the Apache Software Foundation. Vulnerabilities reported for Log4j 1.x after its EOL are not fixed, and no security updates are provided.

Log4j 1.2.17 also has known security weaknesses, including a deserialization-based remote code execution risk in the SocketServer component (CVE-2019-17571). If SocketServer or related network logging features are enabled and exposed to untrusted traffic, an attacker could exploit this flaw to execute arbitrary code.

Because the library is deprecated and unmaintained, any newly discovered issues in the 1.x series will remain permanently unpatched, making its continued use a standing security risk.

Impact Using Log4j 1.2.17 introduces several risks:

1. Exposure to known exploits. CVE-2019-17571 allows deserialization-based RCE under certain configurations, creating a practical compromise path if network logging features are used.
2. No future security fixes. As an end-of-life dependency, the target remains indefinitely exposed to newly discovered Log4j 1.x vulnerabilities.
3. Operational and compliance risk. Unsupported components can violate internal security baselines or external compliance expectations, increasing audit and incident-response burden.
4. System compromise potential. Successful exploitation may enable unauthorized access, data theft, malware deployment, or service disruption.

Suggestion To mitigate the risks associated with log4j 1.2.17, the following actions are strongly advised:

- Upgrade to a supported Log4j 2.x release. Migrate to the latest stable Log4j 2.x version to obtain active security support and modern hardening. Ensure the chosen version includes fixes for Log4Shell-related issues (CVE-2021-44228, CVE-2021-45046) as applicable.
- Apply interim mitigations if upgrade is delayed. Confirm that SocketServer and other network-based appenders are disabled and not exposed to untrusted traffic, since these features are prerequisites for CVE-2019-17571 exploitation.
- Increase monitoring for exploitation attempts. Add detection rules and alerting for anomalous Java logging behavior, unexpected outbound connections, or signs of deserialization payloads, until the dependency is fully replaced.

Feedback from the project The team has confirmed the finding and acknowledged the risk, with alternative mitigation and a fallback plan in place. Specifically:

- The service dependency chain is relatively deep, and a short-term replacement would introduce tangible stability risks.
- The affected functionality does not enable *SocketServer* and exposes no external invocation path; therefore, it does not form an exploitable attack chain.
- Compensating controls are in place via *Bitdefender* EDR/XDR exploit mitigation.

2.1.3 Insecure third-party dependency: Redis

Severity High

Status Confirmed

Description The target environment uses Redis 6.0.16 and Redis 7.0.15. These are stable releases and are generally secure when properly configured. However, recent high-risk vulnerabilities have been disclosed for these versions. Specifically, CVE-2025-49844 and CVE-2024-46981 describe Lua scripting use-after-free issues that can be triggered by specially crafted Lua scripts, potentially leading to remote code execution in affected deployments.

Impact Using vulnerable or insufficiently hardened Redis versions may lead to the following:

1. Remote code execution (RCE). Exploitation of CVE-2025-49844 and CVE-2024-46981 may allow attackers to abuse Lua scripting and garbage-collector behavior to trigger a use-after-free condition and potentially execute arbitrary code on the Redis server.
2. System compromise and lateral movement. A compromised Redis instance can be used as a pivot point for lateral movement, privilege escalation, or malware and ransomware deployment.

Suggestion To mitigate the risks associated with Redis 6.0.16 and 7.0.15, the following actions are strongly advised:

- Upgrade to secure versions. Upgrade Redis 6.0.16 to 6.2.20 or later, and Redis 7.0.15 to 7.2.20 or later, following the latest security advisories.
- Apply temporary hardening if upgrade is delayed.
  - Set a strong password using `requirepass` in `redis.conf` or via `CONFIG SET requirepass`.
  - Use ACLs (Access Control Lists), available since Redis 6.0, to enforce fine-grained permissions and limit Lua scripting capabilities.

  - Rename or disable high-risk commands (e.g., `FLUSHDB`, `FLUSHALL`, `CONFIG`, and `SHUTDOWN`) using rename-command.
- Limit exposure. Ensure Redis is not directly accessible from untrusted networks and is reachable only by required internal services.

Feedback from the project The team has confirmed the finding and considers the residual operational risk acceptable. Specifically:

- All Redis instances are deployed within the internal network, with no externally reachable access paths.
- ACLs, `requirepass`, and rename-command hardening have been applied.
- Compensating controls are provided via *Bitdefender* exploit blocking.

2.1.4 Exposed management interface

Severity Medium

Status Fixed

Description The following management-facing domains are directly accessible from the Internet:

- `testnet-dashboard.megaeth.com`
- `testnetv2-dashboard.megaeth.com`

Impact Because these management interfaces are Internet-reachable, attackers can attempt direct access and probing. If a web-based attack bypasses authentication or authorization controls (the observed 403 response indicates access control is present but may be insufficient), unauthorized parties could gain access to administrative resources, potentially exposing sensitive information or system data through the management interface.

Suggestion Place management-facing web resources behind a privileged access environment and enforce network segmentation to prevent direct Internet access. Require secure access paths such as VPN or bastion entry, and implement strict access controls including zero-trust authentication and multi-factor verification so that only authorized personnel can reach these assets through authenticated, encrypted channels.

Feedback from the project The following measures have been implemented:

- All management-plane entry points have been migrated to the dedicated *SecureLink* network.
- Direct access has been forcibly disabled at the *Cloudflare* layer.
- Unauthorized access previously returned 403 responses and is now fully unreachable.

2.1.5 Leaked GCP service account private keys

Severity Medium

Status Fixed

Description During file system review of the provided Google Cloud VM instances, JSON-formatted GCP service account private key files were found persistently stored on disk. These credentials appear to be used for operational tasks such as data synchronization and log export.

On the `nonprod-observability-node` VM which is used for log collection and monitoring, a service account key file named `gcs-key.json` was located in `/opt/loki/config`. This file contains credentials for

`nonprod-loki-gcs-access@devops-production-464602.iam.gserviceaccount.com`

, which is presumed to support data synchronization.

Figure 2.1: Leaked `gcs-key.json` file

In addition, another service account key file named `gcp-key.json` was identified under `/opt/stackdriver_exporter`. It contains credentials for

`stackdriver-exporter-sa@testnet-v2-467703.iam.gserviceaccount.com`

, likely used for exporting log data and interacting with GCP services.

Figure 2.2: Leaked `gcp-key.json` file

Impact An attacker who gains access to these key files could authenticate as the associated service accounts and enumerate the resources and permissions granted to them. If the accounts are over-privileged or misconfigured, this could enable unauthorized access to GCP resources, privilege escalation, or lateral movement within the cloud environment, potentially extending to additional VM instances or management-plane operations.

Suggestion If persistent storage of these service account key files is not strictly required, remove them from disk and revoke or rotate the affected keys. If local retention is operationally necessary, implement compensating controls, including encrypting the files at rest, enforcing strict file permissions and ACLs to restrict read access to only required processes and privileged users, and monitoring for unauthorized access. Rotation of long-lived keys and migration to workload identity or short-lived credentials are also recommended where feasible.

Feedback from the project The following measures have been implemented:

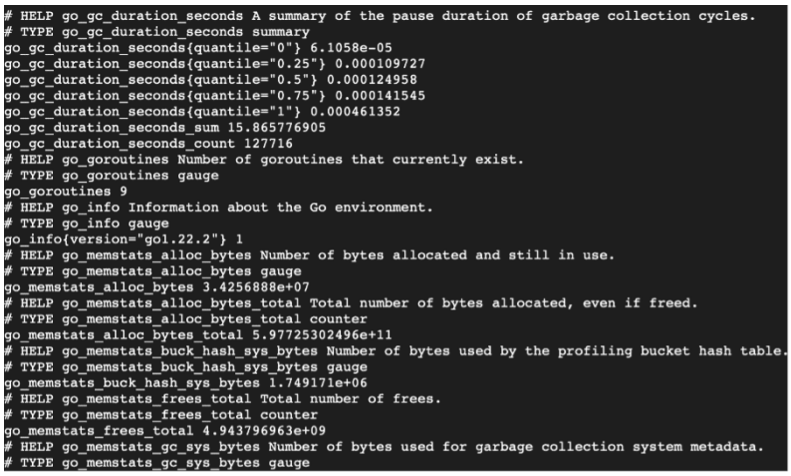
- File names have been obfuscated.
- File storage paths have been modified.
- UAll VMs have been inspected to ensure no residual SA keys remain.
- File permissions, ACLs, and access auditing have been fully re-hardened.

2.1.6 Leaked build and compliation information

Severity Low

Status Fixed

Description The *Prometheus Named Process Exporter* service is deployed on the domain `coriander.megaeth.com` over port 9256. Its metrics endpoint is externally accessible and exposes information such as environment variables, file paths, and software version details. These disclosures can reveal aspects of the build and compilation environment.



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.1058e-05
go_gc_duration_seconds{quantile="0.25"} 0.000109727
go_gc_duration_seconds{quantile="0.5"} 0.000124958
go_gc_duration_seconds{quantile="0.75"} 0.000141545
go_gc_duration_seconds{quantile="1"} 0.000461352
go_gc_duration_seconds_sum 15.865776905
go_gc_duration_seconds_count 127716
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.22.2"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 3.4256888e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 5.97725302496e+11
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.749171e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 4.943796963e+09
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
```

Figure 2.3: Leaked build and compilation information

Impact An attacker could use the leaked build and compilation information to fingerprint deployed components, identify vulnerable software versions or configurations, and develop targeted exploits. This may increase the likelihood of host compromise, privilege escalation, or data exfiltration.

Suggestion Review whether exposing port 9256 to the Internet is operationally necessary. If not required for normal service functionality, restrict access by removing public exposure and allowing connectivity only from authorized internal or management networks through firewall rules and VPC controls.

Feedback from the project The port has been closed, and the service is now accessible only from the internal network.

2.2 Recommendation

2.2.1 Enhance threat intelligence integration and EASM coverage

Status Fixed

Description During the review of the security infrastructure, we observed that threat intelligence feeds are not yet integrated into the defensive ecosystem, and the External Attack Surface Management (EASM) capability has not been fully implemented or operationalized.

Impact Without real-time threat intelligence, the organization has limited ability to proactively detect, contextualize, and respond to emerging threats. In parallel, incomplete EASM reduces

visibility into Internet-facing assets, misconfigurations, and potential attack vectors, increasing the likelihood of undetected exposure and compromise.

Suggestion Integrate relevant threat intelligence feeds into the security defense ecosystem to support intelligence-driven detection, alert triage, and automated response. Establish and mature a continuous EASM program with regular discovery, validation, and remediation cycles to maintain accurate visibility of exposed assets and reduce external attack paths.

Feedback from the project Both threat intelligence and EASM have been deployed and are now part of our routine internal security checks.

2.2.2 Apply pending security patches

Status Confirmed

Description Analysis of the *Bitdefender* scan report indicates that several endpoint hosts are missing patches, including both security and non-security updates. Timely application of these updates is essential to maintaining system integrity and reducing the overall attack surface.

Impact Unpatched systems may retain known vulnerabilities that adversaries can exploit to gain unauthorized access, escalate privileges, or execute malicious code, increasing the risk of compromise across the environment.

Suggestion Apply patches and updates in a controlled manner that minimizes disruption to business operations, prioritizing security-related patches that address known vulnerabilities. Where patching cannot be performed due to operational constraints or compatibility concerns, implement compensating controls through the endpoint protection platform, such as exploit mitigation, behavioral blocking, and host intrusion prevention, to reduce exposure until remediation is possible.

Feedback from the project The project team stated that, after careful assessment, they recognize that immediate patch installation may impact the stability of currently operating services and the network environment. As a compensating measure, they have opted to temporarily rely on the Bitdefender endpoint protection platform for interim coverage against potential exploitation attempts. A formal patch deployment plan will be evaluated and implemented in the future.

Feedback from the project The following measures have been implemented:

- *Patch Freeze* rules have been established based on the service architecture to avoid production-level disruptions.
- All hosts that remain unpatched are covered by XDR protections, providing exploit mitigation.


BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS