AETHERPOINT TECHNOLOGIES

Azure Cloud Security Assessment Report



Prepared by TrillCyber Security

Security Researcher: Jacob Hill

September 21, 2026

Version 1.0 — Final Report

Revisions

Version	Date	Author	Description
0.1	September 21, 2026	Jacob Hill	Initial Draft
1.0	September 25, 2026	Jacob Hill	Final report incorporating technical and mentor review

Table Of Contents

Revisions 2

Table Of Contents 3

1. Executive Summary 4

1.1 Overall Assessment 4

1.2 Key Findings 4

Excessive Container Privileges — High 4

Exposed Git Repository and Credentials — High 4

Server-Side Request Forgery — High 4

Weak Authentication Controls — Medium 5

Plaintext Credentials in Microsoft 365 — Medium 5

1.3 Priority Actions 5

1.4 Assessment Scope 5

1.5 Using This Report 6

2. Vulnerability Findings 7

2.1 Vulnerability Summary 7

2.2 Detailed Findings 8

3. General Comments and References 13

3.1 References 13

3.2 General Comments 13

4. Appendices 14

4.1 Appendix 001 — Excessive Container Privileges 14

4.2 Appendix 002 — Exposed Git Repository and Credentials 15

4.3 Appendix 003 — Server-Side Request Forgery 17

4.4 Appendix 004 — Weak Authentication Controls 18

4.5 Appendix 005 — Plaintext Credentials in Microsoft 365 20

1. Executive Summary

1.1 Overall Assessment

AetherPoint Technologies engaged TrillCyber Security to evaluate the security of its Microsoft cloud services and application-hosting environment. The assessment identified five security weaknesses: three rated High severity and two rated Medium severity.

The results show that an attacker could take advantage of weaknesses in account protection, password storage, application configuration, development-file exposure, and system permissions. These issues could allow unauthorized access to employee accounts, internal company information, stored passwords, cloud credentials, and systems that were not intended to be accessible from outside the organization.

The most serious issue involved an application that had more control over its hosting system than it required. If that application were compromised, an attacker could potentially expand their access beyond the application and affect the underlying server or other applications running on it.

The assessment also found cloud credentials within development history that was accessible through a web application, an application feature that could be used to reach internal services, inconsistent use of additional login verification, and passwords stored in ordinary Microsoft 365 documents and messages.

These weaknesses were identified in different parts of the environment and should be addressed individually. However, they also show how a single compromised account or application could expose additional information that may help an attacker expand their access.

1.2 Key Findings

Excessive Container Privileges — High

An application was operating with more system access than it required. It could run with administrator-level permissions and access files belonging to the underlying server. If an attacker gained control of the application, these permissions could allow the compromise to extend beyond the application itself.

Exposed Git Repository and Credentials — High

Development files were accessible through a web application. Although cloud credentials had been removed from the current version of the code, they remained recoverable from the project's previous versions. An outside attacker could retrieve these files without first signing in and recover potentially reusable cloud credentials.

Server-Side Request Forgery — High

A web application accepted a user-provided destination and contacted that destination on the user's behalf without sufficient restrictions. This allowed the application to be used to identify and access

services that were intended to remain internal. Testing confirmed that a sensitive value could be retrieved from one of those internal services.

Weak Authentication Controls — Medium

A valid Microsoft 365 account was identified and its password was discovered through a limited password-spray test. The credentials could then be used across several Microsoft cloud services without consistently requiring an additional verification step. This increased the likelihood that a stolen or guessed password could result in unauthorized access to company information and cloud resources available to that account.

Plaintext Credentials in Microsoft 365 — Medium

Passwords and other login information were stored in ordinary Microsoft 365 documents and Teams messages. An account with permission to search that content could retrieve the stored credentials. If an employee account were compromised, these passwords could provide an attacker with additional opportunities to access other documents, accounts, or internal services.

1.3 Priority Actions

AetherPoint Technologies should prioritize the following corrective actions:

1. Reduce application and container permissions to the minimum required and remove unnecessary access to the underlying server.
2. Remove development files from public-facing systems, rotate the exposed cloud credentials, and review relevant activity logs for possible misuse.
3. Restrict which destinations web applications are permitted to contact and prevent them from accessing unauthorized internal services.
4. Require multifactor authentication for all users and strengthen protections against password-spraying attacks.
5. Remove passwords from documents and chat messages, rotate the exposed credentials, and store future credentials in an approved password or secrets-management system.

These actions address the conditions most likely to allow initial unauthorized access or help an attacker expand access after compromising an account or application.

1.4 Assessment Scope

The assessment covered AetherPoint Technologies' Microsoft Azure environment, associated Microsoft 365 services, and cloud-hosted application and container infrastructure.

Authorized testing included account and domain reconnaissance, authentication testing, cloud-resource and permission review, Microsoft 365 data-access testing, development-file and credential analysis, internal application-request testing, and container-security validation.

The assessment did not include denial-of-service testing, destructive changes, social engineering, persistence beyond what was necessary to validate a finding, testing of Microsoft-managed infrastructure, or testing of unapproved production and third-party environments.

1.5 Using This Report

This Executive Summary is intended for organizational leadership and provides the primary business risks and remediation priorities in plain language.

Section 2 provides the technical information needed by security, cloud, application, and infrastructure teams to understand and correct each weakness. It includes severity ratings, affected components, technical explanations, recommended controls, and supporting references.

Section 4 contains the testing evidence and reproduction information used to validate the findings. The appendices are intended primarily for the technical personnel responsible for reviewing or correcting the affected systems.

2. Vulnerability Findings

The security vulnerabilities are a list of potentially exploitable issues discovered during the assessment for which a Common Weakness Enumeration (CWE) identifier can be attributed. The severity of each finding is measured using Common Vulnerability Scoring System version 3.1 metrics and considers factors such as attack vector, attack complexity, privileges required, user interaction, scope, and potential effects on confidentiality, integrity, and availability.

CVSS v3.1 uses a numerical score from 0.0 through 10.0. The qualitative severity ratings used in this report are shown below.

Severity	CVSS v3.1 Score
Low	0.1–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

2.1 Vulnerability Summary

Listed in report order.

Finding	CWE	Severity	CVSS
Excessive Container Privileges	CWE-250	High	8.8
Exposed Git Repository and Credentials	CWE-798	High	7.5
Server-Side Request Forgery	CWE-918	High	7.5
Weak Authentication Controls	CWE-308	Medium	6.5
Plaintext Credentials in Microsoft 365	CWE-256	Medium	6.5

2.2 Detailed Findings

2.2.1 Excessive Container Privileges	
Severity	High — CVSS v3.1: 8.8 (AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
CWE	CWE-250: Execution with Unnecessary Privileges
Affected Component	Kubernetes pod security context, Linux capabilities, host filesystem mounts, and the container runtime.
Impact	An attacker who gains code execution within the affected workload could escape the container boundary and obtain root-level access to the underlying host. This could expose other containers, node configuration, credentials, application data, and administrative interfaces, and could enable modification, disruption, or expansion into the wider cluster.
Details	The affected workload operated with privileges and host access beyond those required for its intended function. Root execution, powerful Linux capabilities, and a mounted host filesystem substantially weakened the isolation boundary between the container and the node. Because the container could access the host filesystem and invoke host binaries through a changed root context, compromise of the workload could extend to the underlying host operating system.
Reproduction Steps	Refer to Appendix 001.
Recommendations	• Remove privileged execution and grant only the minimum Linux capabilities required by the workload. • Run containers as a non-root user, set allowPrivilegeEscalation to false, and drop unnecessary capabilities. • Remove host filesystem mounts and container-runtime socket access unless strictly required. • Apply the Kubernetes Restricted Pod Security Standard or an equivalent admission policy. • Monitor workload definitions for privileged mode, host mounts, host namespaces, and dangerous capabilities.
Additional References	MITRE CWE-250 — Execution with Unnecessary Privileges Kubernetes Documentation — Configure a Security Context for a Pod or Container Kubernetes Documentation — Pod Security Standards

2.2.2 Exposed Git Repository and Credentials	
Severity	High — CVSS v3.1: 7.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)
CWE	CWE-798: Use of Hard-coded Credentials; related exposure: CWE-552
Affected Component	Publicly accessible web repository, exposed Git metadata, repository history, and AWS access credentials.
Impact	An unauthenticated attacker could reconstruct the repository and recover cloud credentials from historical commits. The demonstrated impact is disclosure of sensitive source-control content and reusable authentication material. The assessment did not validate the recovered AWS credentials or establish their effective permissions.
Details	Sensitive AWS credentials were committed to source control and remained recoverable from Git history after removal from the current working tree. The web application also exposed its .git directory, allowing the repository to be reconstructed remotely. Removing a secret from the latest revision does not remove it from prior commits; therefore, exposed repository metadata and retained credential history combined to disclose reusable cloud authentication material.
Reproduction Steps	Refer to Appendix 002.
Recommendations	• Immediately deactivate and rotate the exposed AWS access keys, then review associated CloudTrail and service logs. • Remove the credentials from the complete Git history and invalidate all published copies or forks where possible. • Use a managed secrets service, temporary credentials, or workload identities instead of static credentials in source code. • Prevent deployment of .git directories and other development artifacts to web-accessible locations. • Enable automated secret scanning in local development, pull requests, repositories, and deployment pipelines.
Additional References	MITRE CWE-798 — Use of Hard-coded Credentials MITRE CWE-552 — Files or Directories Accessible to External Parties AWS IAM Documentation — Manage Access Keys for IAM Users GitHub Documentation — Removing Sensitive Data from a Repository

2.2.3 Server-Side Request Forgery	
Severity	High — CVSS v3.1: 7.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)
CWE	CWE-918: Server-Side Request Forgery
Affected Component	Kubernetes-hosted web application request functionality, cluster networking, service discovery, and an internal metadata service.
Impact	An unauthenticated attacker could cause the affected application to make requests to services that were not directly exposed externally. The demonstrated attack reached an internal metadata service and retrieved a sensitive value from a secrets endpoint. This could disclose internal configuration or secrets and provide information that supports further access within the cluster.
Details	The application accepted a user-controlled destination and issued server-side requests without adequately restricting the target host, port, or internal address range. Because the application could communicate with other services through the Kubernetes network, an attacker could use it as a proxy to enumerate internal ports and access cluster-only endpoints. Base64 encoding does not protect a disclosed secret because the value can be decoded without a key.
Reproduction Steps	Refer to Appendix 003.
Recommendations	• Restrict server-side requests to an explicit allowlist of required schemes, hosts, and ports. • Block unapproved loopback, link-local, private, and Kubernetes cluster destinations, and repeat validation after DNS resolution and redirects. • Apply default-deny Kubernetes NetworkPolicies and permit only required workload egress. • Require authentication and authorization for internal metadata and microservice endpoints, and do not expose secrets through unauthenticated routes. • Rotate exposed secrets and monitor for internal port scanning, unusual service discovery, and unexpected metadata access.
Additional References	MITRE CWE-918 — Server-Side Request Forgery OWASP — Server-Side Request Forgery Prevention Cheat Sheet Kubernetes Documentation — Network Policies

2.2.4 Weak Authentication Controls	
Severity	Medium — CVSS v3.1: 6.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N)
CWE	CWE-308: Use of Single-factor Authentication; related weakness: CWE-521
Affected Component	Microsoft Entra ID authentication, password policy, multifactor authentication, and Conditional Access.
Impact	An external attacker could identify a valid cloud identity, discover its password through password spraying, and authenticate to Microsoft Graph, Azure management, Microsoft Teams, SharePoint, and OneDrive without an additional authentication factor. This could expose organizational data and provide access to cloud resources assigned to the compromised user. Separate lab evidence documents Key Vault Reader and Automation Account Contributor assignments. These illustrate potential post-compromise access, but the captures do not establish that the password-sprayed identity held those roles.
Details	The environment permitted authentication paths that relied on a reusable password without consistently requiring a second factor. Valid-account discovery and password-spray susceptibility reduced the effort required to compromise an identity, while inconsistent MFA enforcement allowed the recovered credentials to be used across several Microsoft cloud services. A compromised identity can exercise its existing cloud permissions; the resulting impact depends on that identity's actual role assignments and resource access. This places the security of the account and its accessible resources on the strength and secrecy of a single password.
Reproduction Steps	Refer to Appendix 004.
Recommendations	• Require multifactor authentication for all users through Conditional Access policies. • Require phishing-resistant authentication methods for administrative and other high-impact accounts. • Deploy Microsoft Entra Password Protection with banned-password controls. • Configure smart lockout protections and block legacy authentication protocols. • Monitor sign-in logs for password spraying, unfamiliar locations, anomalous clients, and repeated failures.
Additional References	MITRE CWE-308 — Use of Single-factor Authentication MITRE CWE-521 — Weak Password Requirements Microsoft Entra Documentation — Require Multifactor Authentication for All Users Microsoft Entra Documentation — Password Protection Microsoft Entra Documentation — Smart Lockout

2.2.5 Plaintext Credentials in Microsoft 365	
Severity	Medium — CVSS v3.1: 6.5 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N)
CWE	CWE-256: Plaintext Storage of a Password
Affected Component	Microsoft SharePoint, OneDrive, and Teams content accessible through Microsoft Graph.
Impact	A compromised Microsoft 365 account could discover reusable credentials stored in documents and Teams messages. This exposes sensitive authentication material and may enable access to other resources if the credentials remain valid. The assessment did not validate the recovered credentials against the referenced systems.
Details	Passwords and authentication information were stored in plaintext within collaboration documents and Teams messages. An account authorized to read or search that content could also retrieve the stored credentials. Although Microsoft 365 provides access controls and auditing, storing reusable secrets alongside ordinary collaboration content exposes them to the same account-compromise and sharing risks. Credentials require dedicated secrets-management controls, including restricted retrieval and rotation.
Reproduction Steps	Refer to Appendix 005.
Recommendations	• Rotate the exposed credentials and review authentication and resource-access logs for misuse. • Remove plaintext credentials from Microsoft 365 content and store secrets in an approved secrets manager. • Use Microsoft Purview Data Loss Prevention policies to detect and restrict credentials in Teams, SharePoint, and OneDrive. • Apply least-privilege permissions, sensitivity labels, and external-sharing restrictions to sensitive collaboration content. • Train users not to transmit or store passwords in documents, chat messages, or shared workspaces.
Additional References	MITRE CWE-256 — Plaintext Storage of a Password Microsoft Azure Documentation — Azure Key Vault Security Microsoft Purview Documentation — Data Loss Prevention for Microsoft Teams Microsoft Purview Documentation — Data Loss Prevention Overview

3. General Comments and References

3.1 References

- MITRE Common Weakness Enumeration (CWE)
- NIST National Vulnerability Database CVSS Calculator
- FIRST Common Vulnerability Scoring System v3.1
- Microsoft Azure Security Documentation
- OWASP Server-Side Request Forgery Prevention Cheat Sheet
- Kubernetes Security Documentation
- AWS IAM Security Best Practices

3.2 General Comments

The assessment identified multiple security risks related to identity protection, server-side request validation, credential management, collaboration services, source-code security, and container configuration rather than defects in the underlying cloud platforms themselves. In several cases, broad access available to compromised identities, insufficient network restrictions, and misplaced trust increased the likelihood of unauthorized access and sensitive-data exposure.

While individual findings varied in severity, the overall risk was elevated because weaknesses across identity, application, credential, and container layers could independently expose sensitive data or increase access. Weak authentication controls demonstrated that a Microsoft 365 account could be compromised and used across multiple cloud services. Separately documented Azure role assignments illustrate the level of access an identity might possess, but the retained evidence does not establish that the password-sprayed account held those roles. Plaintext credentials stored in collaboration services created additional opportunities for credential reuse and unauthorized access.

The cloud-native environment demonstrated multiple attack paths. Server-side request forgery exposed internal Kubernetes services and a secrets endpoint. Exposed source-control history disclosed cloud credentials, while excessive container privileges and host filesystem access weakened the isolation boundary between an application workload and its underlying host. These conditions could allow compromise of a single application or workload to expand into broader cloud or container-infrastructure access.

Effective risk reduction requires layered security controls across identities, data, applications, and infrastructure. AetherPoint Technologies should prioritize multifactor authentication, least-privilege access, secure secrets management, recurring permission reviews, container hardening, and centralized monitoring for suspicious identity and resource activity.

4. Appendices

4.1 Appendix 001 — Excessive Container Privileges

An interactive shell was obtained within the Kubernetes workload named system-monitor-deployment. The current working directory and effective identity were verified using pwd and whoami, confirming that commands were executing as the root user inside the container.

```
pwd
whoami
```

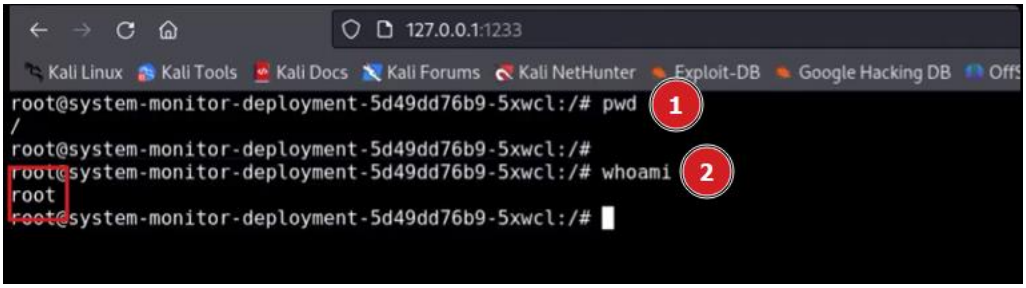


Figure 1.1 — Root-level command execution within the affected container.

The container's Linux capabilities were reviewed using capsh --print. The resulting capability set confirmed that the workload retained elevated capabilities beyond those normally required by an application container, increasing the impact of code execution within the workload.

```
capsh --print
```



Figure 1.2 — Elevated Linux capabilities assigned to the container.

Mounted filesystems were then enumerated with mount, and the host-system mount was inspected. The container could access the underlying node filesystem through /host-system, demonstrating that the workload was not effectively isolated from host resources.

```
mount
ls /host-system/
```



Figure 1.3 — Underlying host filesystem exposed inside the container.

The documented validation used chroot against the mounted host filesystem, followed by identity and runtime checks. The captures establish root execution, elevated capabilities, and host filesystem access. Output from chroot, id, and docker ps was not retained, so successful execution in the host context is not claimed as captured evidence.

```
chroot /host-system bash
id
docker ps
```

End of Appendix 001.

4.2 Appendix 002 — Exposed Git Repository and Credentials

Content discovery against the web application identified an externally accessible .git/config path. This confirmed that Git repository metadata had been deployed within the web-accessible application directory.

```
gobuster dir -u <REDACTED_TARGET> -w <WORDLIST>
```

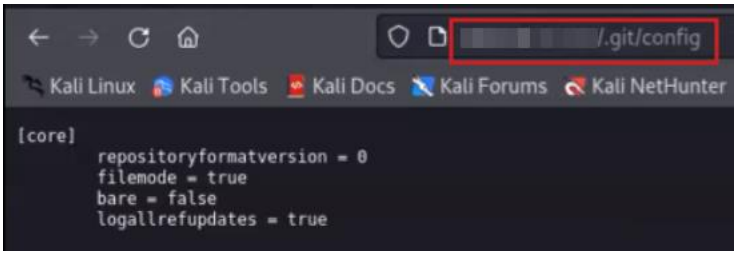


Figure 2.1 — Exposed Git repository metadata identified through content discovery.

Git-Dumper was used to retrieve the exposed repository objects and reconstruct the project locally. Review of the recovered Git log identified a historical commit associated with the addition of custom environment variables.

```
git-dumper <REDACTED_TARGET>/.git/ <OUTPUT_DIRECTORY>
cd <OUTPUT_DIRECTORY>
git log --oneline --all
```

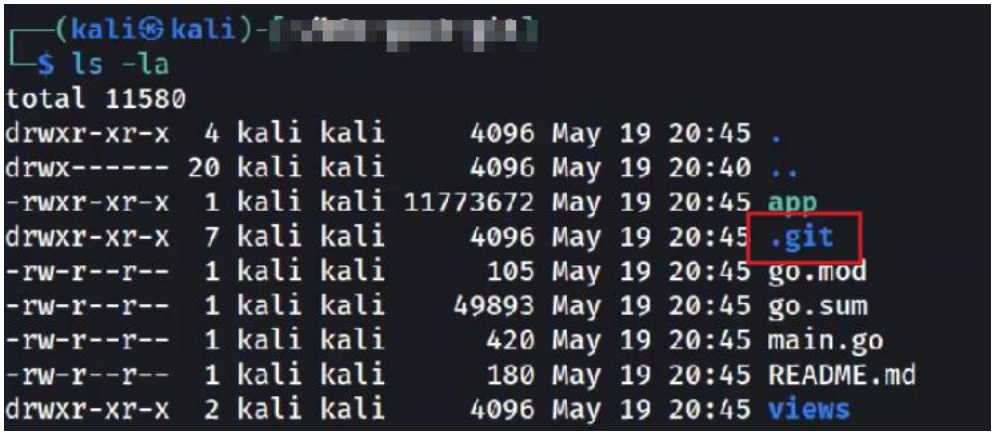


Figure 2.2 — Exposed repository reconstructed and historical commits enumerated.

The historical commit was checked out and its environment file was reviewed. The prior revision contained an AWS access-key identifier and secret access key that were no longer present in the current working tree but remained recoverable from repository history. The actual credential values are intentionally redacted from this report.

```
git checkout <REDACTED_COMMIT>
cat .env
```

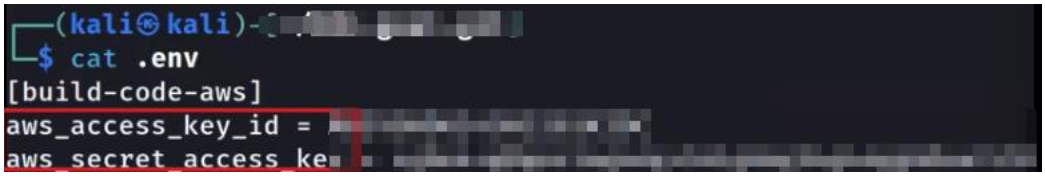


Figure 2.3 — Redacted AWS credentials recovered from historical source-control content.

A subsequent TruffleHog scan from the container environment independently identified the same class of secret exposure, corroborating that sensitive cloud authentication material was present within the repository history.

```
trufflehog git file://<REDACTED_REPOSITORY_PATH>
```

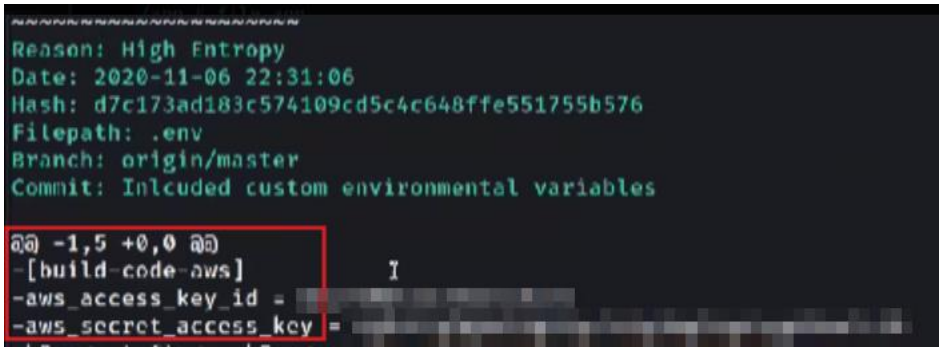


Figure 2.4 — Secret-scanning results corroborating credential exposure.

End of Appendix 002.

4.3 Appendix 003 — Server-Side Request Forgery

The Kubernetes-hosted application accepted a user-controlled destination and issued a request from the server. The request was intercepted and supplied to Burp Suite Intruder to test access to internal services.

```
Burp Suite Intruder payloads: 1000, 2000, 3000, 4000, 5000, 6000, 7000
```



Figure 3.1 — Internal port enumeration through the server-side request identified a responsive service on port 5000.

The responsive service exposed an internal metadata application identified as metadata-db. The server-side request was directed through the service's latest and secrets paths until the Kubernetes Goat secret endpoint was reached.

```
http://metadata-db/latest/secrets/kubernetes-goat
```



Figure 3.2 — The SSRF condition provided access to an internal metadata secrets endpoint.

The endpoint returned a Base64-encoded value. Local decoding confirmed disclosure of the internal service's secret value.

```
echo -n "<REDACTED_BASE64_VALUE>" | base64 -d
```



Figure 3.3 — The value retrieved from the internal secrets endpoint was successfully decoded.

Insufficient destination validation enabled internal port enumeration, access to the cluster-only metadata service, and disclosure of a secret.

End of Appendix 003.

4.4 Appendix 004 — Weak Authentication Controls

External Microsoft 365 reconnaissance was performed to identify valid accounts associated with the target tenant. A candidate email list was evaluated with Oh365UserFinder, which identified an active Microsoft 365 identity and saved the validated address for authorized authentication testing.

```
python .\oh365userfinder.py -r <REDACTED_PATH>\emails.txt -w validemails.txt
-t 10
```

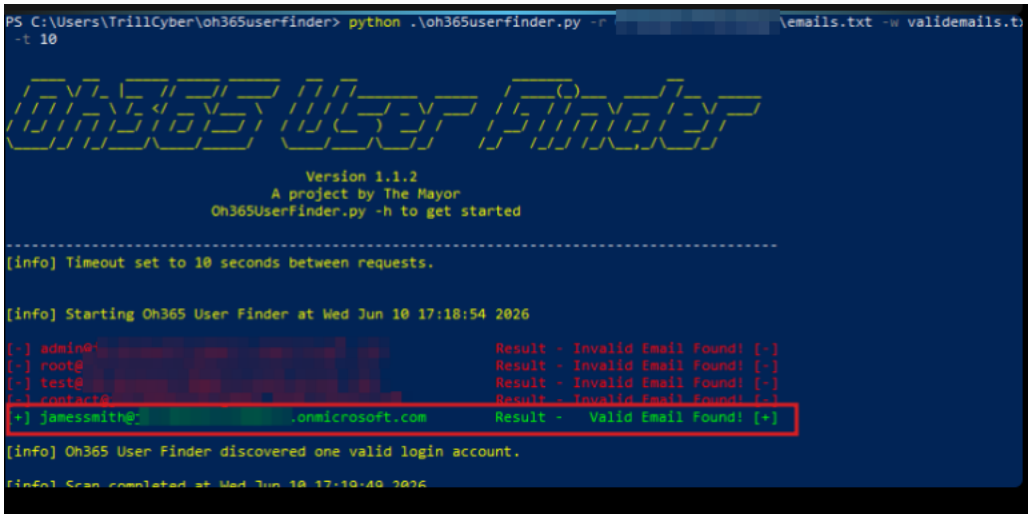


Figure 4.1 — Valid Microsoft 365 account identified through Oh365UserFinder.

The validated account list was supplied to MSOLSpray to test a single authorized password against the identified Microsoft 365 account. The tool returned a successful authentication result, confirming that a valid account password had been discovered through password spraying. The tested password and affected identity are redacted.

```
. .\MSOLSpray.ps1
Invoke-MSOLSpray -UserList <REDACTED_PATH>\validemails.txt -Password
<REDACTED> -Verbose
```

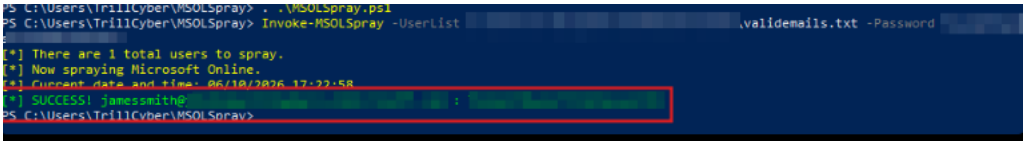


Figure 4.2 — Successful authentication identified through MSOLSpray.

MFASweep was then used to evaluate whether the compromised credentials triggered an additional authentication requirement across Microsoft cloud services. Authentication succeeded without an MFA challenge against Microsoft Graph, the Azure Service Management API, Microsoft Teams, and Microsoft Office applications associated with OneDrive and SharePoint.

```
python .\mfasweep.py --username <REDACTED> --password <REDACTED>
```

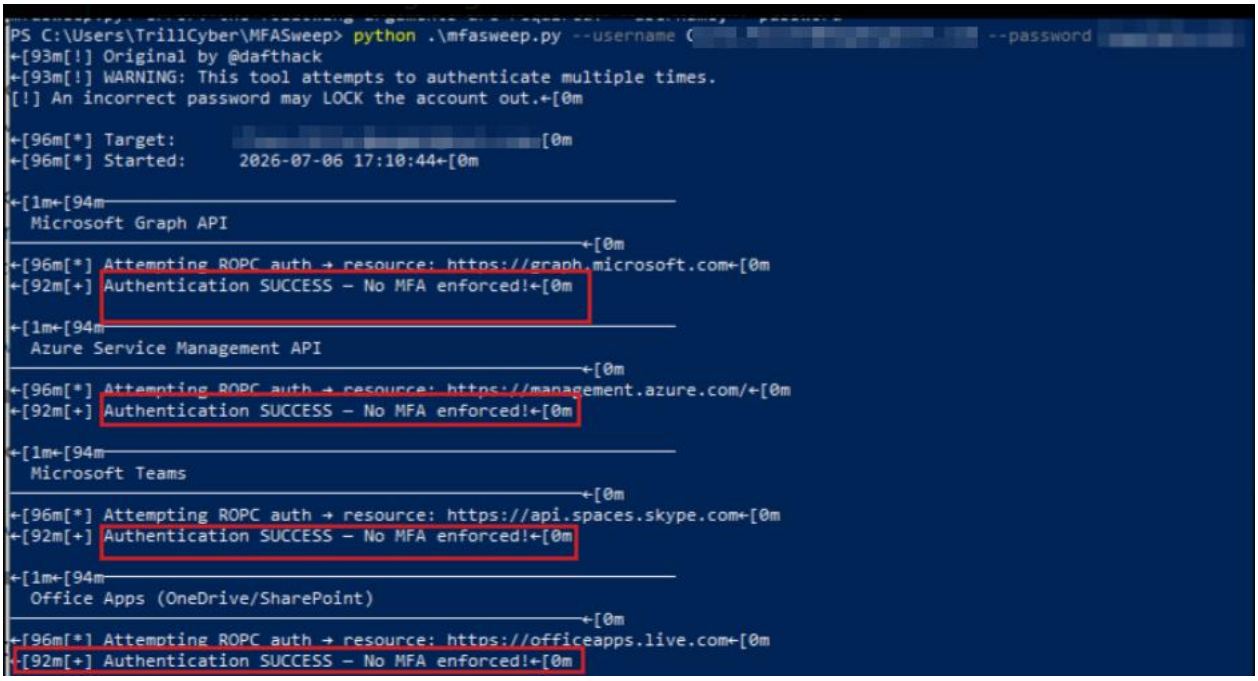


Figure 4.3 — Authentication succeeded across multiple Microsoft cloud services without MFA enforcement.

Separate Azure enumeration documented a Key Vault Reader assignment for a lab identity. The role permits visibility into vault configuration and metadata, but does not itself permit retrieval of secret values. The captures do not establish that this was the identity identified by password spraying.



Figure 4.4 — Key Vault Reader assignment documented for a lab identity.

The environment configuration documentation also recorded a Contributor assignment for the same identity on an Azure Automation Account. Contributor permits broad management-plane operations within its assigned scope, although it does not permit the identity to create Azure role assignments.

These assignments illustrate potential impact if an identity with such roles is compromised; they do not establish the password-sprayed account's permissions. The Key Vault Reader role exposes vault metadata, and Contributor can permit management operations within its assigned scope. Final Azure CLI output for the Contributor assignment was not retained.

The results demonstrate that an externally identified Microsoft 365 account could be compromised through password spraying and used across multiple cloud services with only a reusable password. Inconsistent MFA enforcement allowed the compromised credentials to be used across Microsoft 365 and the Azure management service. A compromised identity could also exercise any Azure roles assigned to that identity; however, the retained evidence does not establish that the password-sprayed account held the separately documented Key Vault Reader or Contributor roles.

End of Appendix 004.

4.5 Appendix 005 — Plaintext Credentials in Microsoft 365

GraphRunner was used with an authenticated lab identity to obtain a Microsoft Graph access token. The token provided access to Microsoft 365 resources available to that user. This evidence illustrates post-authentication credential exposure; it does not establish that the same identity was used in the password-spray test in Appendix 004.

```
$tokens = Get-GraphTokens
```

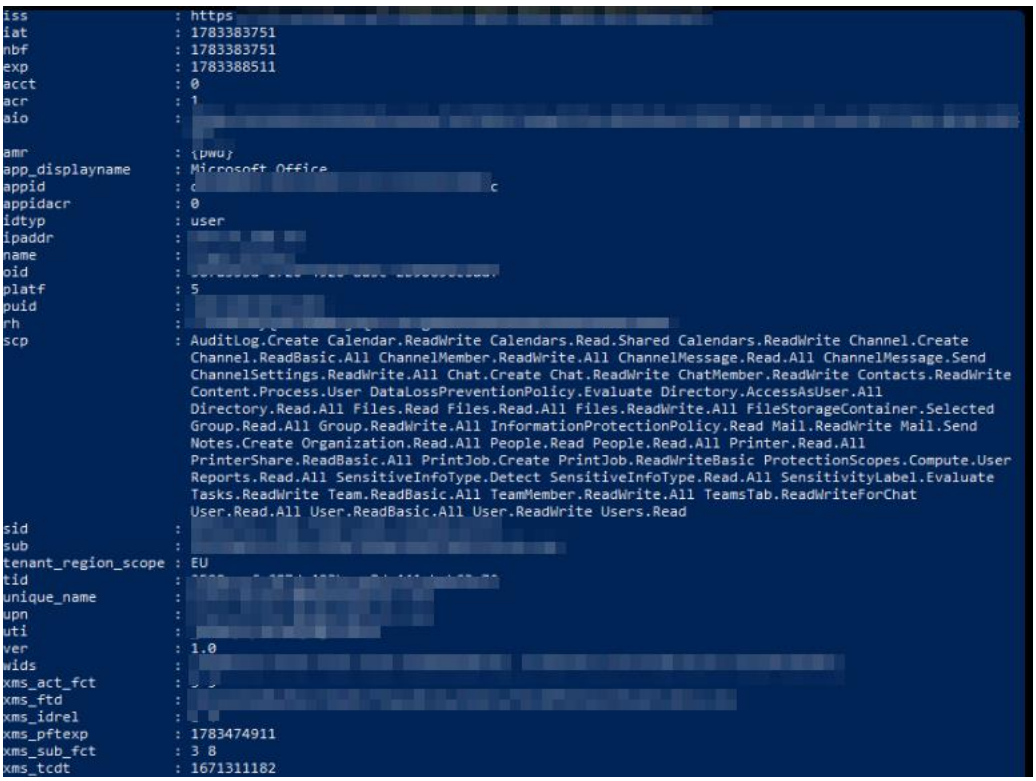


Figure 5.1 — Microsoft Graph token obtained for an authenticated lab identity.

GraphRunner was used to search SharePoint and OneDrive content available through the token for files containing the term password. The search returned a spreadsheet containing site usernames and passwords and a finance document containing login information for internal services. Plaintext credential material was visible directly within the returned file previews.

```
Invoke-SearchSharePointAndOneDrive -Tokens $tokens -SearchTerm "password"
```

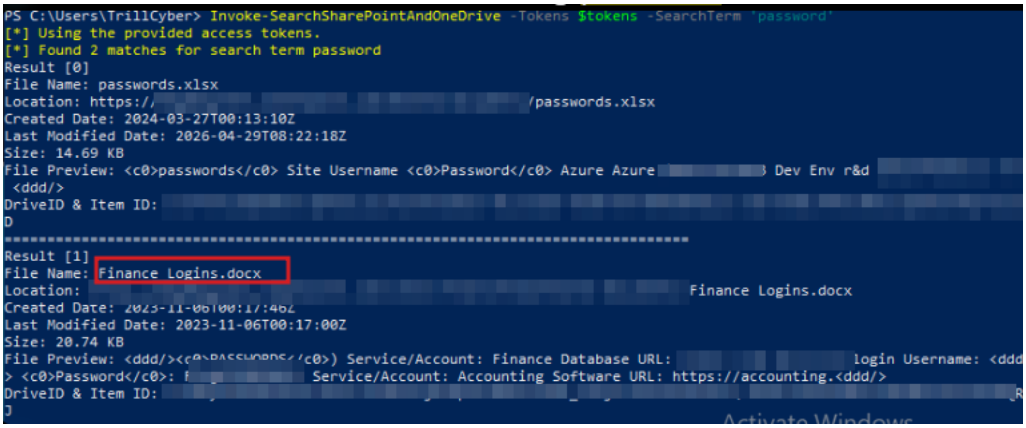


Figure 5.2 — SharePoint and OneDrive results exposing credential-related files.

A separate search for the term bonus identified a confidential, password-protected spreadsheet containing employee financial information. Although the document itself required a password, Microsoft Teams messages accessible to the authenticated lab identity were found to contain that password in plaintext.

```
Invoke-SearchSharePointAndOneDrive -Tokens $tokens -SearchTerm "bonus"
Invoke-SearchTeams -Tokens $tokens -SearchTerm "password"
```



Figure 5.3 — Confidential financial document identified through SharePoint and OneDrive search.

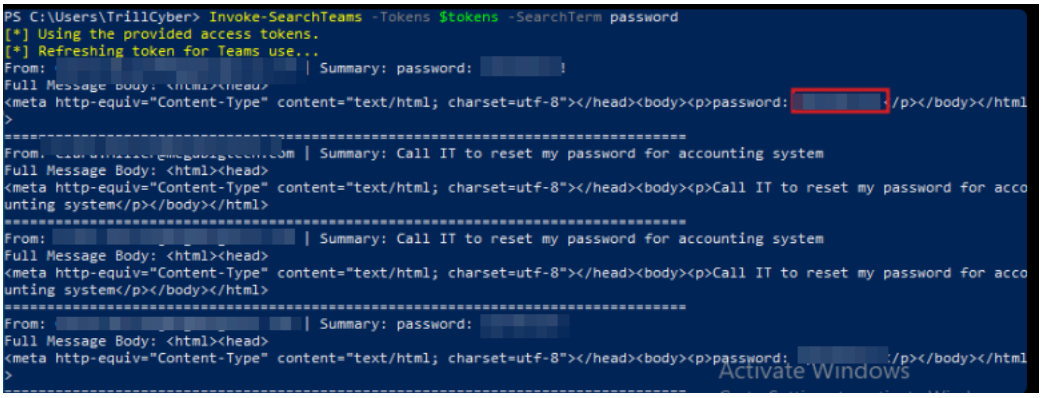


Figure 5.4 — Plaintext document password exposed in Microsoft Teams.

The recovered value corresponded to the password required by the protected document. This demonstrated that credentials and document passwords stored in Microsoft 365 collaboration content could be discovered through ordinary search functionality by an authorized user account, creating opportunities for unauthorized access and lateral movement if that account were compromised.

End of Appendix 005.