Midnight Blizzard attack on Microsoft corporate environment: a detailed analysis, detections and recommendations

By Lior Sonntag

Published: 2024-02-08 · Archived: 2026-04-05 21:16:37 UTC

On January 25th, Microsoft disclosed a security breach by a Russian cyber group identified as Midnight Blizzard (among other names), that targeted email accounts from November 2023 to January 2024. The actors initially gained access by compromising a legacy, non-production test tenant account that did not have MFA (Multi Factor Authentication) enabled, and subsequently moved laterally to the main Microsoft corporate production tenant. They secured elevated privileges within Microsoft's own Exchange Online tenant, resulting in unrestricted access to their corporate mailboxes.

In this blog, we'll provide a detailed analysis of the whole attack chain. We'll discuss what we think might have happened while focusing mainly on the OAuth attack techniques used in this campaign, as there are still significant pieces missing from the report (which obscures a full understanding of all the details). In addition, we'll also provide some detections/threat-hunting queries related to potential risks posed by OAuth applications and provide some general recommendations and best-practices that any organization might implement to reduce the risk of being compromised by malicious OAuth applications and brute-force/password-spray attacks.

Attack analysis

Microsoft's report states that after the adversary gained initial access to the legacy test tenant account by compromising an Entra ID user through a password-spray technique, it compromised a legacy test OAuth application that had elevated access to the Microsoft corporate environment.

Based on this, we can confidently claim that the initial compromised Entra ID account had the privilege to create new secrets/certificates for OAuth applications in the test tenant, subsequently allowing the adversary to authenticate as the test app and execute actions on behalf of it.

The statement "legacy test OAuth application that had elevated access to the Microsoft corporate environment" indicates that the test application has been previously granted consent by a highly privileged Entra ID user in the corporate tenant — allowing it high privileges in the corporate environment. To understand what privileges were previously granted by a legitimate admin, we'll take note of the following statement: "They created a new user account to grant consent in the Microsoft corporate environment." In other words, either the MS Graph app permission of `Directory.ReadWrite.All` or `User.ReadWrite.All` was previously granted consent to the corporate tenant, since these are the only MS Graph permissions that allow creation of new Entra ID users. For the demonstration's purpose, let's assume that the `Directory.ReadWrite.All` permission was granted consent.

The report later states that the adversary used the Office 365 Exchange Online app permission of `full_access_as_app` to access corporate mailboxes. Since it requires admin consent, we can conclude that the newly created user in the corporate environment was assigned admin privileges. Thus, we can infer that the MS

Graph app permission of `RoleManagement.ReadWrite.Directory` was also previously granted access to the corporate environment (prior to the incident).

From the Microsoft corporate tenant, the service principal representing the legacy test OAuth application (in the test tenant) would look like this, indicating that the MS Graph permissions of `Directory.ReadWrite.All` and `RoleManagement.ReadWrite.Directory` have been granted consent on behalf of the entire tenant:

The attacker then abused the previously granted MS Graph elevated permissions in the corporate environment, using the credentials of the OAuth test application to create a new Entra ID user with admin privileges. This was done in three steps.

Step 1: First, the attacker requests an access token that provides access to the corporate tenant with the previously granted privileges:

Step 2: With this token, the attacker uses the `Directory.ReadWrite.All` permission to create a new user in the corporate tenant:

Step 3: Then the attacker utilizes the `RoleManagement.ReadWrite.Directory` permission to assign the user the Global Administrator role:

Furthermore, the report states that the adversary also created additional OAuth applications, although it doesn't indicate whether they were created by the initial compromised user or the test app (we'll assume the test app created them). It also doesn't specify whether they were created in the test tenant or the prod tenant. Considering that this nation-state APT group is known for its sophisticated operations and the ability to stay under the radar and evade detection, we'll assume that they would not choose to create apps in the prod tenant, since Microsoft almost certainly rigidly monitors their production environments.

This means that the test app also had the elevated MS Graph app permission of `Application.ReadWrite.All`. The report also states that the threat actor used the legacy test OAuth app to grant their new malicious apps the elevated Office 365 Exchange Online permission of `full_access_as_app`, meaning that the test app also had the highly privileged MS Graph permission of `AppRoleAssignment.ReadWrite.All`.

Overall, the legacy test OAuth application in the test tenant might have had the following MS Graph permissions:

- `Directory.ReadWrite.All`: MS Graph permission that was previously granted in the Microsoft corporate tenant – allowing the adversaries to create a new user in the corporate tenant.
- `RoleManagement.ReadWrite.Directory`: MS Graph permission that was previously granted in the Microsoft corporate tenant – allowing the adversaries to assign any directory roles to their new user.
- `Application.ReadWrite.All`: MS Graph permission that allows the OAuth app to create new OAuth applications. This permission allowed the attacker to create the new malicious OAuth application.
- `AppRoleAssignment.ReadWrite.All`: MS Graph permission that allows the OAuth app to assign new permissions for existing applications. This permission allowed the attacker to assign the `full_access_as_app` permission to their malicious application.

The procedure for creating a new, potentially malicious OAuth application and assigning it the Office 365 Exchange Online permission `full_access_as_app` via the legacy test OAuth app might proceed as follows:

- Creating a new multi-tenant OAuth application:
- Assigning the new application the Office 365 Exchange Online permission of `full_access_as_app`:

Afterward, the new **malicious-mail-access** OAuth application is created in the test tenant and might look like this:

The adversaries would then need to create their new malicious OAuth application credentials (probably using the same user account that was initially compromised via the password-spray attack), and then initiate an illicit-consent attack where they would consent to the Office 365 Exchange Online permission of `full_access_as_app` using their new admin user in the Microsoft corporate tenant, subsequently allowing unrestricted access to any mailbox associated with users in the corporate tenant. This procedure for this operation could be as follows:

The adversaries sign in to their malicious multi-tenant app using their new admin user in the Microsoft corporate tenant.

Then, they utilize the user's administrative privileges to grant consent to their malicious app, which aims to obtain full access to all corporate mailboxes.

Finally, upon obtaining consent, the attackers would only need to request an access token with the credentials of their **malicious-mail-access** OAuth application (a name we've chosen for the purpose of this demonstration) to gain access to any mailbox within the corporate tenant:

Detecting similar activity in your environment

Following are some detection/hunting queries related to anomalous activity associated with Entra ID OAuth applications, using Entra ID (AAD) Audit Logs in Azure Log analytics.

1. Detect assignments of high privileged permissions to OAuth applications:

This query will detect the assignment of application permissions of the type noted in this blog. If you're interested in covering additional MS Graph permissions, refer to the official Microsoft documentation.

2. Detect assignments of high privileged directory roles to Entra ID users, specifically the Global Administrator, Privileged Role Administrator, Application Administrator, and Cloud Application Administrator built-in roles that can grant tenant-wide admin consent for OAuth applications:

3. Detect secrets/certificates generated for multiple OAuth applications by a specific principal in a short period of time:

Be sure to define the time frame and the threshold according to your requirements.

4. Detect whenever a new third-party application has been consented in your tenant:

Be sure to monitor new service principals being added to your tenant as a result of consenting to a third-party OAuth application. If you find any new suspicious service principal, investigate their consented permissions and determine whether this is a legitimate application accessing your organization's data.

Recommendations and best practices

Enforce MFA in your Entra ID tenant

This kind of attack could have been prevented if strict security controls were in place, such as having MFA on each Entra ID user in the tenant.

Be sure to enforce MFA authentication registration policy in your tenant by following these steps.

Enable Smart Lockout

Azure AD Smart Lockout defends against brute force or password-spray attacks by locking out attackers while ensuring that legitimate users can access their accounts. It differentiates between likely legitimate sign-ins and attacker attempts, setting thresholds and durations for account lockouts to balance security and accessibility. To enforce it in your tenant, follow these steps.

Prevent illicit application consent attacks

By default, all users can grant consent to applications that don't require administrative review. To reduce the risk of malicious applications trying to trick users into granting them access to your organization's data, we highly recommend restricting users from consenting to unverified third-party applications. To configure user consent settings, follow these steps and be sure to select the second option (**Allow user consent for apps from verified publishers**):

How can Wiz help detect and prevent this kind of attack chain?

Wiz offers a comprehensive solution to detect and prevent similar attacks.

Detection

Wiz customers can use Wiz CDR (Cloud Detection and Response) to detect emerging cloud threats in real-time, such as the following scenarios:

1. Detect a successful brute-force attack on a specific Entra ID user account.
2. Detect multiple valid Entra ID user accounts failing to authenticate from the same IP address (usually associated with password-spray attacks).
3. Detect creation of new credentials to multiple OAuth applications by a specific Entra ID principal in a short period of time.
4. Detect potential illicit admin consent to a third-party application that requests high privileged permissions.

Prevention

Wiz customers can use Wiz to assess the risk in their environment and detect relevant toxic and risky combinations, similar to the ones exploited in the above-described attack:

1. Discover whether you have highly privileged third-party service principals in your tenant that were previously granted consent.
2. Keep track of all the highly privileged Entra ID users that don’t have MFA enabled.

Stay informed

The Midnight Blizzard attack involved a sophisticated, multi-layered approach. It serves as a stark reminder of the importance of robust security measures for organizations doing business in the cloud. Wiz customers can use the Wiz CDR for relevant threat detection information and built-in Controls to identify related risks in their environments. For any questions or help with patching or mitigating vulnerabilities, please don't hesitate to contact us at threat.hunters@wiz.io.

We would like to thank Matt Graeber and Justin Schoenfeld from Red Canary for reaching out and bringing to our attention that the final attack step requires the Office365 Outlook *scope specification to retrieve the relevant app role token.*
