Legitimate Apps as Traitorware for Persistent Microsoft 365 Compromise

By Sharon Martin

Published: 2023-08-03 · Archived: 2026-04-05 23:16:36 UTC

The idea of "persistence" in a cloud environment is not a well-studied topic. At most, you hear instances of the attacker creating backup logins to maintain their long-term presence in a cloud environment.

To continue our series exposing the tradecraft around business email compromise (BEC), this blog will dive into how Huntress identified a threat actor using a novel form of persistence (M365 applications) in order to try to stay under the radar and avoid detection. We discovered a compromised user account with the ability to add apps during the beta phase of our newest product, Huntress Managed Identity Threat Detection and Response.

What Happened

This is another unfortunate case of compromised credentials without additional security controls.

There was a failed login from a US IP, and then shortly thereafter, a successful login via a US IP. However, it was clear quite quickly that this wasn't a normal IP—it was a proxy/VPN IP. Here's an overall screenshot of the timeline of events that will be explained in more detail below:

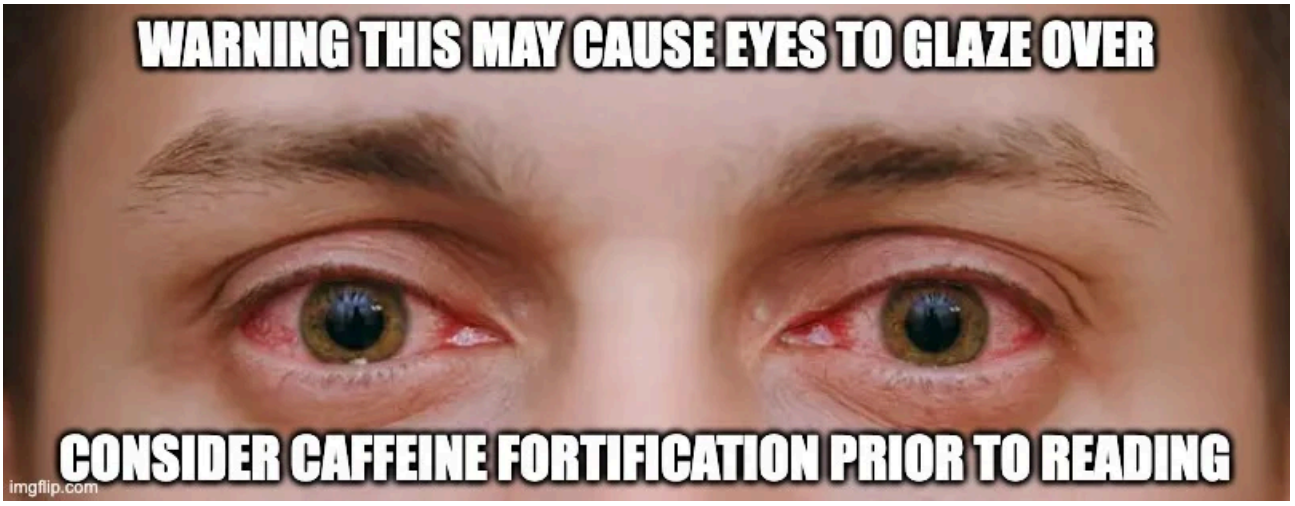
79 hits

Documents | Field statistics BETA

@timestamp	event.code	event.action	o365.audit.Parameters.FromAddressContainsWords	o365.audit.Parameters.MarkAsRead	o365.audit.Parameters.MoveToFolder	o365.audit.Target.Name	o365.audit.ModifiedProperties.DelegatedPermissionGrant.Scope.NewValue
Jun 12, 2023 @ 22:32:54.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:22:31.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:22:30.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:22:30.000	AzureActiveDirectory	Consent to application.	-	-	-	Newsletter Software SuperMailer	-
Jun 12, 2023 @ 22:22:30.000	AzureActiveDirectory	Add app role assignment grant to user.	-	-	-	Newsletter Software SuperMailer	-
Jun 12, 2023 @ 22:22:29.000	AzureActiveDirectory	Add delegated permission grant.	-	-	-	Microsoft Graph	offline_access Mail.Read Mail.Send Contacts.Read
Jun 12, 2023 @ 22:22:29.000	AzureActiveDirectory	Add service principal.	-	-	-	Newsletter Software SuperMailer	-
Jun 12, 2023 @ 22:22:21.000	AzureActiveDirectoryStsLogon	UserLoginFailed	-	-	-	-	-
Jun 12, 2023 @ 22:19:16.000	ExchangeAdmin	New-InboxRule	@	True	Deleted Items	-	-
Jun 12, 2023 @ 22:17:47.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:44.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:44.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:44.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:36.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:34.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:34.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:33.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:30.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:17:24.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 22:14:37.000	AzureActiveDirectoryStsLogon	UserLoginFailed	-	-	-	-	-
Jun 12, 2023 @ 20:08:38.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 18:43:40.000	AzureActiveDirectory	Consent to application.	-	-	-	eM Client	-
Jun 12, 2023 @ 18:43:40.000	AzureActiveDirectory	Add app role assignment grant to user.	-	-	-	eM Client	-
Jun 12, 2023 @ 18:43:40.000	AzureActiveDirectory	Add delegated permission grant.	-	-	-	Microsoft Graph	EWS.AccessAsUser.All offline_access email openid
Jun 12, 2023 @ 18:43:40.000	AzureActiveDirectory	Add service principal.	-	-	-	eM Client	-
Jun 12, 2023 @ 18:43:40.000	AzureActiveDirectoryStsLogon	UserLoggedIn	-	-	-	-	-
Jun 12, 2023 @ 18:43:32.000	AzureActiveDirectoryStsLogon	UserLoginFailed	-	-	-	-	-

Click to enlarge

Detailed Event Breakdown



The events you saw above are where it started to get more interesting. We saw an application added with several events in Azure around it:

“Add service principal.”

- When you register a new application in Azure AD, a service principal is automatically created for the app registration. For more details, see this Microsoft write-up.
- The important detail to note here is that the “Target.Name” has the name of the application which was added, “eM Client”. eM Client is an app that can integrate with email, calendar, etc. The “InterSystemsId” can also be used to help correlate this eM Client target with other event logs—it’s the GUID used to track actions across components in the Office 365 service.

“Add delegated permission grant.”

- When an application that’s added requires access, a delegated permission grant is created for the permissions needed by the application on behalf of the user. For more details, this article provides some great background.
- The “InterSystemsId” was the same as the previous event, showing that it’s related to the eM Client application being added.
- Actual permissions granted are shown in “ModifiedProperties.DelegatedPermissionGrant_Scope.NewValue”. In this case, the application was granted the following permissions:
- **“EWS.AccessAsUser.All”** - This configures the app for delegated authentication. The app would have the same access to mailboxes as the signed-in user via Exchange web services.
- **“Offline_access”** - This gives the app access to resources on behalf of the user for an extended period of time. On the permission page itself that would pop up, it’s described as “Maintain access to data you have given it access to.” When a user approves this access scope, the app can receive long-lived refresh tokens from the Microsoft Identity platform token endpoint and then reach out to get an updated access token when the older one expires—all without user intervention.
- **“Email”** - This can be used along with the “Openid” scope and gives the app access to the user’s primary email address.

- **“Openid”** - This indicated the app signed in by using OpenID Connect. It allows the app to get a unique identifier for the user (a sub-claim), which can then be used to acquire identity tokens that are used for authentication.

“Add app role assignment grant to user.”

- We saw the same “InterSystemsId” as the above events correlating it to the same email app.
- This means the app has been assigned to a user via Azure AD so that the user can access the app. The “ModifiedProperties.User_UPN.NewValue” indicates which user it’s been assigned to. In this case, it’s the same user the threat actor was logged into.

“Consent to application.”

- In a well-configured Azure environment, admin approval and consent should be needed to add any new apps. These configurations are called risk-based or step-up consent.
- Alas, the “Actor.UPN” of our hacked user account along with the success of the log for “InterSystemsId” show that consent was able to be granted by this user. So either they were an admin or one of the consent models was not configured, allowing any user to add any application.

Wait, There’s More?

Adding just one app was apparently not enough for this threat actor—or perhaps, the app didn’t allow them to do everything they wanted to do, which seems to be sending and receiving emails on behalf of the user. But before adding another app, the threat actor again showed some more sophistication in their attack.

When there’s a risk that something you’re doing as a threat actor can generate emails to the user, the obvious solution is to prevent the user from seeing said emails. How? Well, of course, with our favorite Microsoft 365 threat actor tradecraft of using email inbox rules. 🦹

The rules added were pretty much as expected. They set up a rule that matched “@”. Yes, it would have matched **any** email. Then, messages were marked as read and moved to Deleted Items. 🪣

Once that was in place, the threat actor went through the step of adding another app to manage email. This time it was Newsletter Software SuperMailer, another legitimate app that’s great for sending mass amounts of emails in a short period of time. This app had some slightly different permissions in addition to “offline_access”:

- **Mail.Read** - Allows the app to read email in user mailboxes.
- **Mail.Send** - Allows the app to send mail as users in the organization.
- **Contacts.Read** - Allows the app to read user contacts.

The permissions paired with the app name seem to indicate that the intent is to send emails to all the contacts of the user that look like they are coming from the user. Perhaps follow-on phishing emails so the threat actor can gain access to more valuable user accounts?

Setting the probability of the app sending a welcome email aside, another reason the threat actor would not want the user to see any emails arriving in their inbox is simple: the legitimate user would be alerted faster to the

compromise if any contacts reply asking “what in the world is this email you just sent me?”

Why Is This a Big Deal?

Let’s go back to the meaning of the “offline_access” permission. **Any** app with this access permission can continue to get new authentication tokens from Microsoft, even **after** the threat actor no longer controls the compromised account. So, the threat actor would have continued happily reading and sending emails all on behalf of this user account until the application access was revoked; thus maintaining persistent access to the compromised account.

Imagine someone stole your car keys including your key fob, then cloned the key fob. Even if you got back your original set of keys, they can use that cloned key fob to keep unlocking your car because that code is authorized to control the car alarm. That’s essentially what the threat actor was doing.

Closing Thoughts

So what’s the best way to prevent this kind of attack?

- MFA, MFA, MFA. If this account had been protected via two-factor or multi-factor authentication, this would have made our threat actor’s job much more difficult.
- Normal users should not be allowed to add new apps. This is like allowing any user to install any application on their PC—you never know what they will install. In Microsoft 365, this is as simple as turning the user consent to apps feature off.

As always, we hope this helps those of you hunting sneaky threat actors in the Microsoft Cloud. If ever you decide you need someone to provide some Managed Identity Threat Detection and Response, so you don’t have to make your eyes bleed reviewing arcane logging events, you know who to call. 😉

Catch up on the other BEC tradecraft we exposed in part one, part two, part three, and part four.
