OAUTH ABUSE: THINK SOLARWINDS/SOLORIGATE CAMPAIGN WITH FOCUS ON CLOUD APPLICATIONS

proofpoint.com/us/blog/cloud-security/oauth-abuse-think-solarwindssolorigate-campaign-focus-cloud-applications

March 12, 2021

Blog
Cloud Security
OAUTH ABUSE: THINK SOLARWINDS/SOLORIGATE CAMPAIGN WITH FOCUS ON CLOUD APPLICATIONS

March 23, 2021 Itir Clarke and Assaf Friedman

Enterprise app stores such as Microsoft AppSource and Google Workspace Marketplace offer millions of useful OAuth apps and add-ons: analytics, security, CRM, document management, project management and more. An OAuth app is an application that integrates with a cloud service and may be provided by a vendor other than the cloud service provider. These apps add business features and user-interface enhancements to cloud services such as Microsoft 365 and Google Workspace. Most OAuth apps request permission to access and manage user information and data and sign into other cloud apps on the user's behalf. For example, they can access users' files, read their calendars, send emails on their behalf and more. Given the broad permissions they can have to your core cloud applications, OAuth apps have become a growing attack surface and vector. Attackers use various methods to abuse OAuth apps, including compromising app certificates, which was also used in the SolarWinds/Solorigate campaign.

Risk of Cloud-to-Cloud Integration with OAuth

These add-on apps use OAuth authentication to obtain limited access to cloud services. OAuth enables a user's account information or data to be used by apps without exposing the user's password. OAuth works over HTTPS, using access tokens (rather than login credentials) to authorize devices, APIs, servers, and applications. OAuth apps can be added to an entire domain or to an individual user account. OAuth apps can easily be exploited. Attackers can use OAuth access to compromise and takeover cloud accounts. Until the OAuth token is explicitly revoked, the attacker has persistent access to the user's account and data. In simple terms, we can describe these risky apps as malicious, vulnerable and abused.

Malicious apps phish users for permissions to access cloud resources

A malicious application is cloud malware that uses various tricks such as OAuth token phishing and app impersonation to manipulate account owners into consent.

These apps usually require sensitive user delegated permissions so they can avoid being detected by an IT administrator and still get access to the sensitive resources. Such resources may include:

- Email, which allows reading, sending on users' behalf and setting up mail-forwarding rules
- Files to exfiltrate data or plant malware to share with contacts
- Other resources that allow access to data or enable lateral movement within the organization

In 2020 alone, we discovered more than 180 malicious applications, most of them were attacking multiple tenants. The ones that were the most widespread attacked more than 200 different tenants with one app instance. An example of such a widespread and ongoing attack is "MexOAuth\TA2552." It targets Spanish-speaking users (specifically Mexican citizens). This attack starts with phishing emails sent to enterprise users. The phishing URL redirects to the enterprise login and app consent page. The page impersonates the Mexican tax authorities, Facebook or Amazon. It even sometimes has a COVID-19 theme. Many families of malicious applications attack specifically VPs, Managers, HR leaders and other senior accounts with high visibility to sensitive assets.

Bad coding or design can make OAuth apps vulnerable to attackers

Vulnerable applications are applications that are susceptible to a hostile takeover by a bad actor due to bad coding or design. Such applications will not require interaction between the attacker and the target accounts to turn rogue. Instead, an attacker will compromise the applications' assets or mechanisms. Such examples were observed in the Microsoft Teams application in March 2020, where sharing a GIF may result in an account takeover.

OAuth abuse often rides on the coattails of account compromise

Abused applications are legitimate applications that are authorized and/or used by an attacker to perform a non-legitimate activity such as exfiltrate data, maintain persistence on specific resources while compromising an account or any other goal.

The main reason for choosing that method would be the low footprint and the durability:

- With abused applications, the attacker can either add a legitimate application to the cloud tenant or use an existing one previously authorized by the account owner. The actual consent of a new app would raise no questions and the logs would be legitimate. In contrast, malicious applications usually require abnormal consent from the account owner.
- Such attacks are resistant to password changes and multifactor authentication (MFA) setups, which are the two main tools used against account compromise. This also is true for malicious OAuth apps.

OAuth abuse is widespread

OAuth application abuse follows on the coattails of account compromise. Proofpoint monitors thousands of cloud tenants and over 20 million active cloud users. In a study of 2020 data, Proofpoint observed the following:

- 95% of organizations were targeted
- 52% of organizations had at least one compromised account
- 32% of compromised organizations had post-access activity such as file manipulation, email forwarding, and OAuth app activity
- 10% of organizations had authorized malicious OAuth apps

Without visibility to OAuth app abuse, your cloud users and data are not safe from threat actors. So, let’s dig deeper into the topic of OAuth abuse.

Application abuse: When and Why?

An application abuse is not a usual "exploit" or a vulnerability, but simply the misuse of a legitimate application for the following reasons:

- Maintain persistence:
  - Authorization of an application in most cases is a one-time effort. The application will maintain access to the resources owned by the account owner until the OAuth token is revoked (refresh token) or the app is deleted.
  - Setting MFA or changing credentials will not affect the application token generation capabilities.
- Leave a low footprint:
  - Abusing a legitimate application will raise no questions upon consent, application sign-in, or activity.
  - Other than consent to an application in a compromised session, there will be little correlation between the abuse setup and the malicious activity.
- From an attacker's standpoint, setting up an application for abuse can be performed easily or automatically, using CLI \ PowerShell scripts.
- In most cases, an attacker will have to compromise the target account first to abuse an application. The attacker can achieve this by malicious login to the target account, man-in-the-middle attacks (session hijack), malicious applications, etc. Compromising a target account refers to an in-session attack. An attack that will not require the user session such as endpoint malware refers to an out-of-session attack.

How attackers abuse OAuth apps and how to spot different methods:

Application login:

An attacker can abuse an application by directly accessing it. This can be done in several methods:

- Logging into the target’s existing application from the attacker's device: This method requires the target’s credentials and can be achieved by password spraying or credentials stuffing attacks, phishing and others. The app can be accessed from any device. However, credentials brute force is likely to be noisy (with many unsuccessful logins) and is not simple to conduct.
- Authorizing a new application while compromising an account and setting it up for external use:

This method requires the attacker to authorize an application with wide and sensitive permissions including one that allows external API usage. Most likely, the account owner will not even notice the abused application added to his library. However, this method has a slightly higher footprint - an unknown consent to an application can be monitored. Having said that, in most organizations consenting to an application is not an abnormal event and may go unnoticed.

In November 2020, we observed an application consent during a compromised session. An attacker who managed to compromise an account authorized an application named "Mailbird". This application is an email client, which allows integration between many services, such as mail service, WhatsApp, Facebook, Dropbox and others. As such, it requires native mail permissions (use IMAP, POP and SMTP on user’s behalf). So, how can this application be abused? As this application allows a "unified account", the attackers can add an account created for the sake of the attack to the active mail account used by the account owner. This would allow full control over the attacked account mailbox remotely. As this application is being used by several other users in the tenant, such authorization and usage is unlikely to raise flags or create security alerts of any kind. Again, this is not a vulnerability or a flaw in the abused application.

Certificate update (used in the SolarWinds/Solorigate campaign):

In the recent SolarWinds attack, X.509 certificates played a crucial role. These certificates are used in many Internet protocols to transfer data privately and securely between a server and a client. When signed by a trusted certificate authority, this digital certificate verifies that you are who you say you are - the domain, organization, or the individual contained within the certificate. Attackers can steal certificates or create them by compromising signing systems.

This method was key in maintaining persistence on accounts affected by the attack on SolarWinds application update. The attacker was able to generate signed certificates from an admin’s compromised endpoint. After generating the certificate, the attacker added it to the application service principal visible in the applications “certificates and secrets” section in the Azure portal.

The attacker chose a previously authorized application, one with highly privileged permissions, such as application type mail scopes or even directory roles. Once the attacker chose an application and added the certificate, an OAuth2.0 token was generated by sending a JSON Web Token (JWT) post request to the authorization server. The list of OAuth2.0 authorization and token generation endpoints (as well as SAML and federated sign-on endpoints) is available on the application section of Azure Active Directory (Azure AD). After the token was granted, the attacker used it to access the resources permitted to the abused application, bypassing the application altogether.

This type of attack has an exceptionally low footprint. It is rarely detectable, especially since such an operation can only be tracked in the audit logs for 7 days (in most cases) and is considered to be a completely legitimate action. Authentications are also logged, however, it's likely to be done by the application and the audit logs cannot trace which certificate was used to authenticate.

Note that self-signed certificates can also be used to complete such authentication.

Although the SolarWinds attack was not a cloud-based attack, it did utilize this method in order to maintain persistence and have mail visibility that would not have been available when using the Sunburst malware alone.

Client Secret abuse:

A client secret is an application password. It is used in a way similar to that of application certificates. A client secret is created in the same Azure AD dashboard as certificates. Secrets can be used with OAuth2.0 flows other than “Authorization Code Flow” (i.e. mobile\native apps). The secrets are usually 34 bytes long and immutable. The secret can be stored and transferred as plain text. The secret can be copied upon secret creation before it is obfuscated. In many cases, this compels the account owner to store the secret as a plain text in a file, mail or note.

While an attacker can create a secret and use it in a similar way to a certificate, the actual operation of secret addition is logged and can be tracked by security analysts. Using an existing secret will eliminate this.

First, the attacker will look for a target application with sensitive permissions and a client secret. Since the “hint” (usually the first 3 characters) for the secret is stored in the application manifest, the attacker will seek a 34 bytes long passphrase starting with the “hint” in the files and mails of the compromised user. If found, this will allow the attacker to generate tokens for the abused application untracked.

Configuration changes:

Once an account has been compromised, the attacker has clear visibility to the account owner applications list. Every application has its own settings. Many allow several types of collaborations or interfaces with other accounts or applications. The attacker can configure an application of choice to interact with an account or an application set on the attacker's end. Although changing settings is simple, detecting the change is not straightforward. There is a huge list of available applications and internal application activity is not logged.

Let’s take the popular Outlook application as an example. While compromising an account and logging in to the Outlook application, an attacker can change settings:

- Add an email delegation with full email access (read, write and send mails, tasks, calendar events, contacts and notes)
- Add a forwarding rule - a straightforward way of getting notifications on important mails (or a full mail replication) to an external account
- Add a certificate authentication, as described above

Of course, an attacker can do the same using malicious applications.

Abuse reply \ redirect URL

When an application is authenticated, the token is shared with a reply URL, which can designate the application host or the user client. After compromising an account, an attacker can add a reply URL (web\SSO) to an existing application in the app registration section to abuse it. The "owned applications" section in Azure AD lists all applications owned by the compromised user. By simply registering a new URL, the attacker can obtain the application secret or token, but only after the account owner logs in to the application. An admin account, if compromised, can edit every application in Azure AD and use them to access resources belonging to any user account. This attack method leaves a small footprint – the application will have more than one redirection with a different domain.

Side by side comparison of OAuth abuse:

	Evasiveness	Attacker value	Persistence	Implementation	In \ Out of Session
Certificate abuse	Medium	High	Medium	Medium	Out
Secret abuse	High	High	Medium	Hard	Out
Application login	Medium	High	High	Easy	In

Configuration changes	Medium	Medium	High	Easy	In
Reply URL	Low	Medium	High	Medium	Out

App governance is key to mitigating OAuth abuse

- Application abuse, an attacker using a legitimate application, is not an application vulnerability
- Organizations should actively govern OAuth applications, minimizing the granted permissions to applications and managing the user list to reduce risk
- Applications with admin roles and app-delegated permissions pose a greater risk because when abused, they provide wider access to an attacker
- Never store plain text secrets and code signing keys
- Manage roles – an application should have owners who understand the application's resource access mechanism and track its changes. Do not allow every user to sign certificates.
- Note anomalies – odd consent to an uncommon application; addition of secrets, certificates and reply URLs of unfamiliar domains.

How Proofpoint CASB can help:

Proofpoint Cloud App Security Broker (Proofpoint CASB) detects, assesses and revokes OAuth permissions for third-party apps and scripts that access your IT-approved core cloud services. Our in-depth analysis helps identify risky apps, including malicious ones, and reduce your attack surface. Based on risk score and context, you can define or automate actions. For example, you can manually or automatically revoke OAuth tokens for Microsoft 365 and Google apps that can pose risk if abused.

Protect users and data with Proofpoint's people-centric security for cloud apps. Proofpoint Cloud App Security Broker combines compromised account detection, data loss prevention (DLP), cloud and third-party apps governance with adaptive access controls to help you secure Microsoft 365, Google Workspace, Box, Salesforce, AWS, Azure, Slack and more.

For more information on OAuth apps and how to govern them, download our whitepaper, What Every Security Professional Should Know About Third-party OAuth Apps.

Subscribe to the Proofpoint Blog