


# Threat actors misuse OAuth applications to automate financially driven attacks

 [microsoft.com/en-us/security/blog/2023/12/12/threat-actors-misuse-oauth-applications-to-automate-financially-driven-attacks/](https://microsoft.com/en-us/security/blog/2023/12/12/threat-actors-misuse-oauth-applications-to-automate-financially-driven-attacks/)

December 12, 2023



By

Threat actors are misusing OAuth applications as an automation tool in financially motivated attacks. OAuth is an open standard for token-based authentication and authorization that enables applications to get access to data and resources based on permissions set by a user. Threat actors compromise user accounts to create, modify, and grant high privileges to OAuth applications that they can misuse to hide malicious activity. The misuse of OAuth also enables threat actors to maintain access to applications even if they lose access to the initially compromised account.

In attacks observed by Microsoft Threat Intelligence, threat actors launched phishing or password spraying attacks to compromise user accounts that did not have strong authentication mechanisms and had permissions to create or modify OAuth applications. The threat actors misused the OAuth applications with high privilege permissions to deploy virtual

machines (VMs) for cryptocurrency mining, establish persistence following business email compromise (BEC), and launch spamming activity using the targeted organization's resources and domain name.

Microsoft continuously tracks attacks that misuse of OAuth applications for a wide range of malicious activity. This visibility enhances the detection of malicious OAuth applications via Microsoft Defender for Cloud Apps and prevents compromised user accounts from accessing resources via Microsoft Defender XDR and Microsoft Entra Identity Protection. In this blog post, we present cases where threat actors compromised user accounts and misused OAuth applications for their financially driven attacks, outline recommendations for organizations to mitigate such attacks, and provide detailed information on how Microsoft detects related activity:

## **OAuth applications to deploy VMs for cryptomining**

---

Microsoft observed the threat actor tracked as Storm-1283 using a compromised user account to create an OAuth application and deploy VMs for cryptomining. The compromised account allowed Storm-1283 to sign in via virtual private network (VPN), create a new single-tenant OAuth application in Microsoft Entra ID named similarly as the Microsoft Entra ID tenant domain name, and add a set of secrets to the application. As the compromised account had an ownership role on an Azure subscription, the actor also granted 'Contributor' role permission for the application to one of the active subscriptions using the compromised account.

The actor also leveraged existing line-of-business (LOB) OAuth applications that the compromised user account had access to in the tenant by adding an additional set of credentials to those applications. The actor initially deployed a small set of VMs in the same compromised subscriptions using one of the existing applications and initiated the cryptomining activity. The actor then later returned to deploy more VMs using the new application. Targeted organizations incurred compute fees ranging from 10,000 to 1.5 million USD from the attacks, depending on the actor's activity and duration of the attack.

Storm-1283 looked to maintain the setup as long as possible to increase the chance of successful cryptomining activity. We assess that, for this reason, the actor used the naming convention `[DOMAINNAME]_[ZONENAME]_[1-9]` (the tenant name followed by the region name) for the VMs to avoid suspicion.

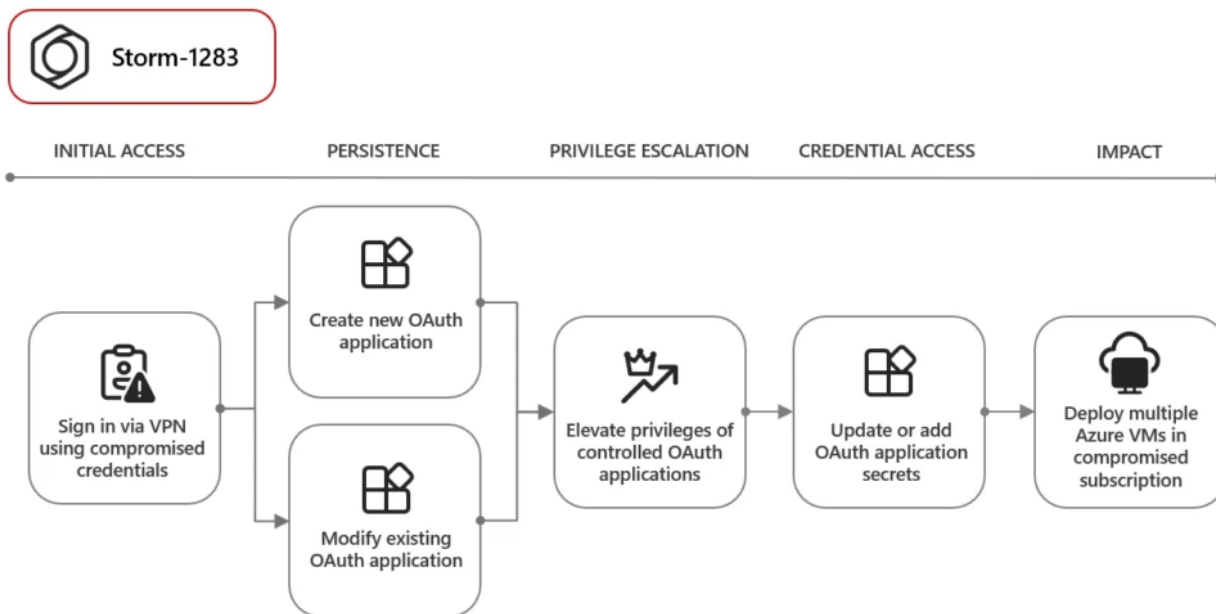


Figure 1. OAuth application for cryptocurrency mining attack chain

One of the ways to recognize the behavior of this actor is to monitor VM creation in Azure Resource Manager audit logs and look for the activity

“*Microsoft.Compute/virtualMachines/write*” performed by an OAuth application. While the naming convention used by the actor may change in time, it may still include the domain name or region names like

“*east|west|south|north|central|japan|france|australia|canada|korea|uk|poland|brazil*”

Microsoft Threat Intelligence analysts were able to detect the threat actor’s actions and worked with the Microsoft Entra team to block the OAuth applications that were part of this attack. Affected organizations were also informed of the activity and recommended further actions.

## OAuth applications for BEC and phishing

In another attack observed by Microsoft, a threat actor compromised user accounts and created OAuth applications to maintain persistence and to launch email phishing activity. The threat actor used an adversary-in-the-middle (AiTM) phishing kit to send a significant number of emails with varying subject lines and URLs to target user accounts in multiple organizations. In [AiTM attacks](#), threat actors attempt to steal session tokens from their targets by sending phishing emails with a malicious URL that leads to a proxy server that facilitates a genuine authentication process.

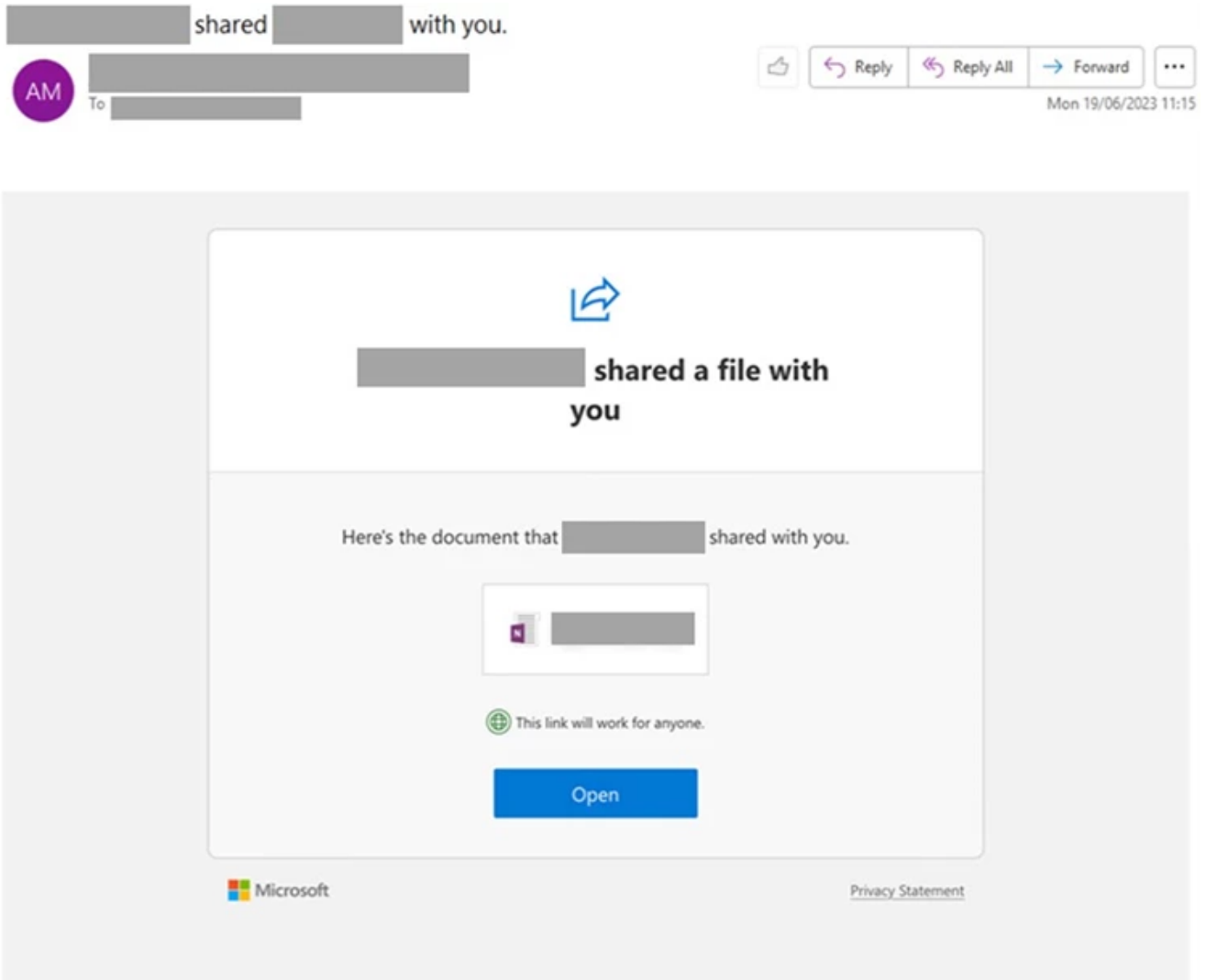


Figure 2. Snippet of sample phishing email sent by the threat actor

We observed the following email subjects used in the phishing emails:

- *<Username> shared "<Username> contracts" with you.*
- *<Username> shared "<User domain>" with you.*
- *OneDrive: You have received a new document today*
- *<Username> Mailbox password expiry*
- *Mailbox password expiry*
- *<Username> You have Encrypted message*
- *Encrypted message received*

After the targets clicked the malicious URL in the email, they were redirected to the Microsoft sign-in page that was proxied by the threat actor's proxy server. The proxy server set up by the threat actor allowed them to steal the token from the user's session cookie. Later, the stolen token was leveraged to perform session cookie replay activity. Microsoft was able to confirm during further investigation that the compromised user account was flagged for risky sign-ins when the account was used to sign in from an unfamiliar location and from an uncommon user agent.

## For persistence following business email compromise

In some cases, following the stolen session cookie replay activity, the actor leveraged the compromised user account to perform BEC financial fraud reconnaissance by opening email attachments in Microsoft Outlook Web Application (OWA) that contain specific keywords such as “payment” and “invoice”. This action typically precedes financial fraud attacks where the threat actor seeks out financial conversations and attempts to socially engineer one party to modify payment information to an account under attacker control.

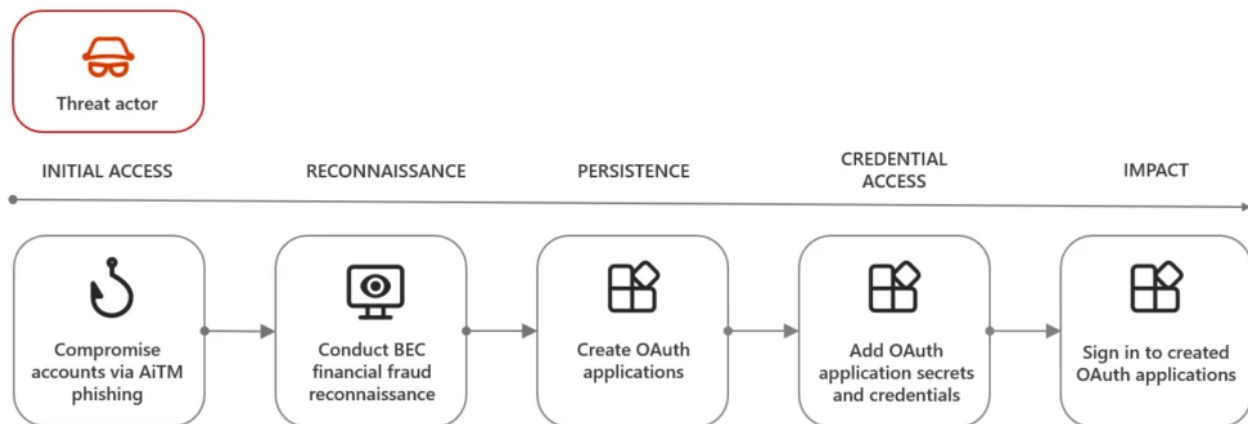


Figure 3. Attack chain for OAuth application misuse following BEC

Later, to maintain persistence and carry out malicious actions, the threat actor created an OAuth application using the compromised user account. The actor then operated under the compromised user account session to add new credentials to the OAuth application.

## For email phishing activity

In other cases, instead of performing BEC reconnaissance, the threat actor created multitenant OAuth applications following the stolen session cookie replay activity. The threat actor used the OAuth applications to maintain persistence, add new credentials, and then access Microsoft Graph API resource to read emails or send phishing emails.

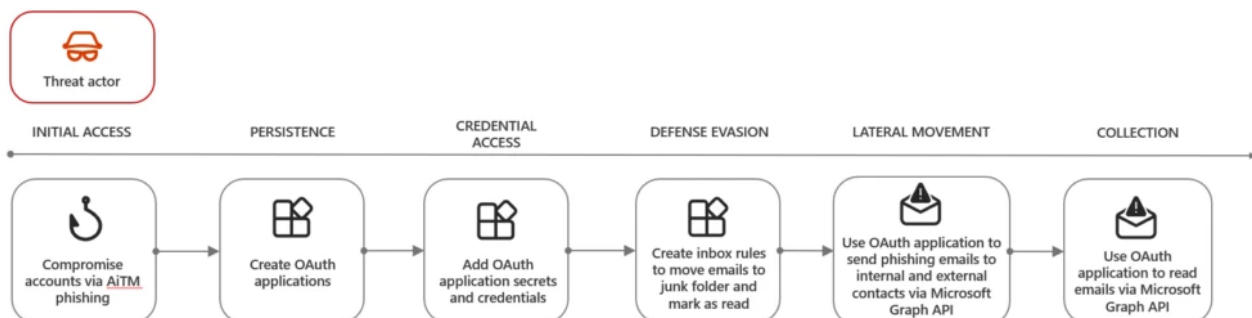
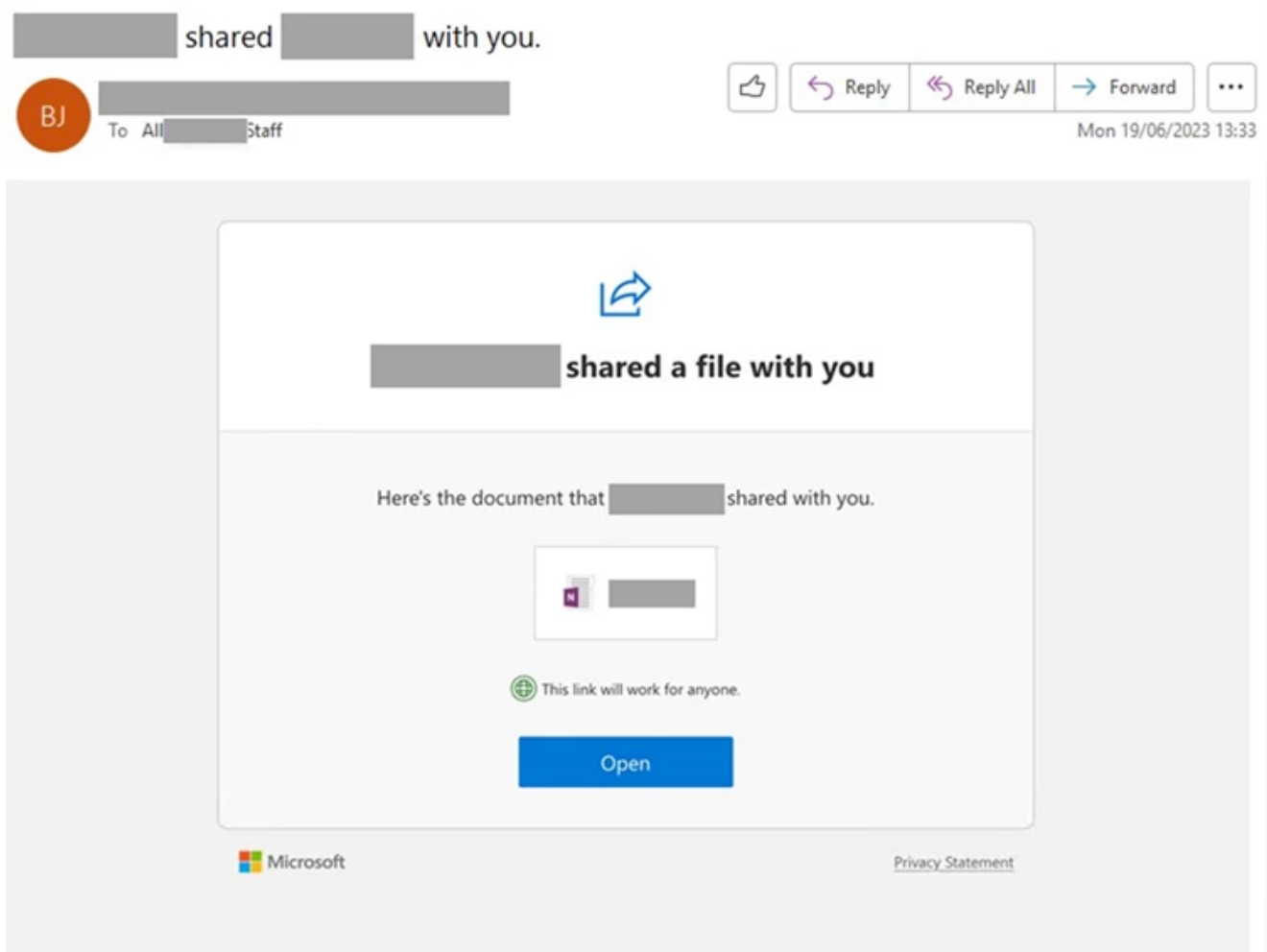


Figure 4. Attack chain for OAuth application misuse for phishing

At the time of analysis, we observed that threat actor created around 17,000 multitenant OAuth applications across different tenants using multiple compromised user accounts. The created applications mostly had two different sets of application metadata properties, such as display name and scope:

- Malicious multitenant OAuth applications with the display name set as “oauth” were granted permissions “user.read; mail.readwrite; email; profile; openid; mail.read; people.read” and access to Microsoft Graph API and read emails.
- Malicious multitenant OAuth applications with the display name set as “App” were granted permissions “user.read; mail.readwrite; email; profile; openid; mail.send” and access to Microsoft Graph API to send high volumes of phishing emails to both intra-organizational and external organizations.



*Figure 5. Sample phishing email sent by the malicious OAuth application*

In addition, we observed that the threat actor, before using the OAuth applications to send phishing emails, leveraged the compromised user accounts to create inbox rules with suspicious rule names like “...” to move emails to the junk folder and mark them as read. This is to evade detection by the compromised user that the account was used to send phishing emails.

```
"Operation": "New-InboxRule",
"Parameters": [
  {
    {
      "Name": "MoveToFolder",
      "Value": "Junk Email"
    },
    {
      "Name": "Name",
      "Value": "..."
    },
    {
      "Name": "MarkAsRead",
      "Value": "True"
    }
  }
]
```

Figure 6. Inbox rule created by the threat

actor using the compromised user account

Based on the email telemetry, we observed that the malicious OAuth applications created by the threat actor sent more than 927,000 phishing emails. Microsoft has taken down all the malicious OAuth applications found related to this campaign, which ran from July to November 2023.

## OAuth applications for spamming activity

---

Microsoft also observed large-scale spamming activity through OAuth applications by a threat actor tracked as Storm-1286. The actor launched password spraying attacks to compromise user accounts, the majority of which did not have multifactor authentication (MFA) enabled. We also observed the user agent *BAV2ROPC* in the sign-in activities related to the compromised accounts, which indicated the use of legacy authentication protocols such as IMAP and SMTP that do not support MFA.

We observed the actor using the compromised user accounts to create anywhere from one to three new OAuth applications in the targeted organization using [Azure PowerShell](#) or a Swagger Codegen-based client. The threat actor then granted consent to the applications using the compromised accounts. These applications were set with permissions like *email*, *profile*, *openid*, *Mail.Send*, *User.Read* and *Mail.Read*, which allowed the actor to control the mailbox and send thousands of emails a day using the compromised user account and the



organization domain. In some cases, the actor waited for months after the initial access and setting up of OAuth applications before starting the spam activity using the applications. The actor also used legitimate domains to avoid phishing and spamming detectors.

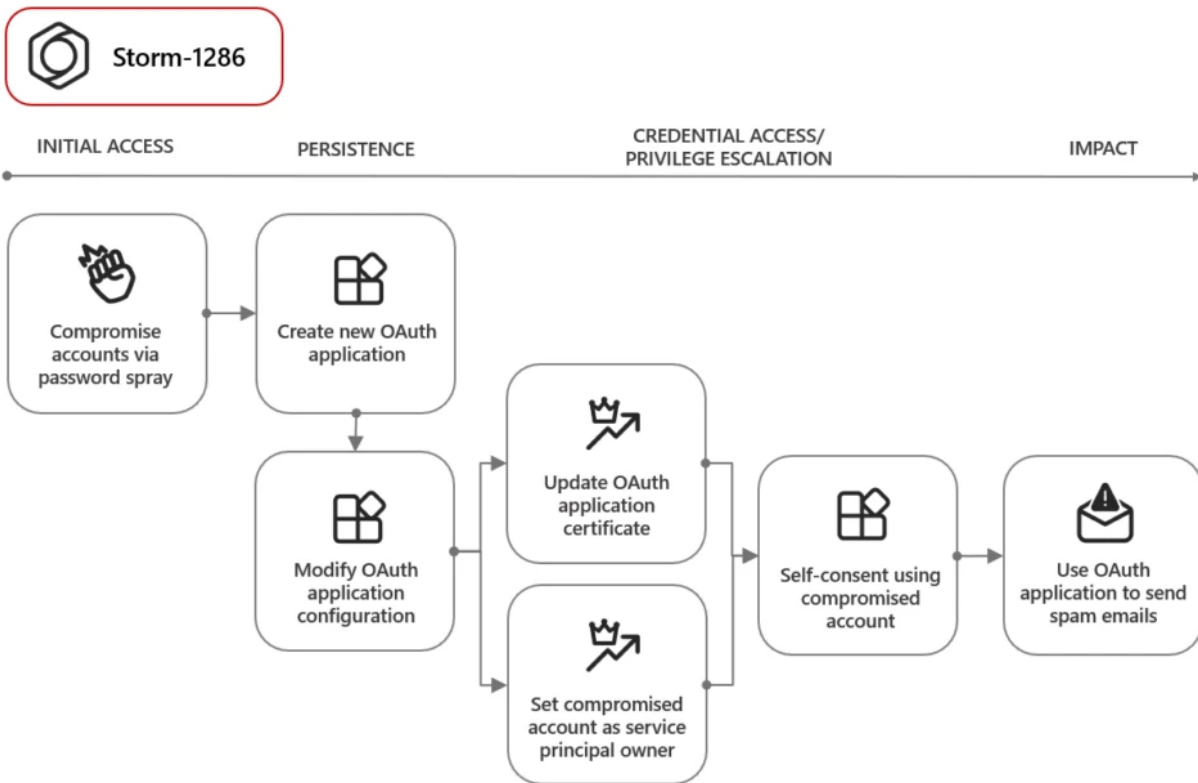


Figure 7. Attack chain for large-scale spam using OAuth applications

In previous large-scale spam activities, we observed threat actors attempting to compromise admin accounts without MFA and create new LOB applications with high administrative permissions to abuse Microsoft Exchange Online and spread spam. While the activity of the actor then was limited due to actions taken by Microsoft Threat Intelligence such as blocking clusters of the OAuth applications in the past, Storm-1286 continues to try new ways to set a similar high-scale spamming platform in victim organizations by using non-privileged users.

## Mitigation steps

Microsoft recommends the following mitigations to reduce the impact of these types of threats.

### Mitigate credential guessing attacks risks

A key step in reducing the attack surface is securing the identity infrastructure. The most common initial access vector observed in this attack was account compromise through credential stuffing, phishing, and reverse proxy (AiTM) phishing. In most cases the



compromised accounts did not have MFA enabled. Implementing [security practices that strengthen account credentials](#) such as enabling MFA reduced the chance of attack dramatically.

## **Enable conditional access policies**

---

[Conditional access](#) policies are evaluated and enforced every time the user attempts to sign in. Organizations can protect themselves from attacks that leverage stolen credentials by enabling policies for [User and Sign-in Risk](#), device compliance and trusted IP address requirements. If your organization has a [Microsoft-Managed Conditional Access](#) policy, make sure it is enforced.

## **Ensure continuous access evaluation is enabled**

---

[Continuous access evaluation](#) (CAE) revokes access in real time when changes in user conditions trigger risks, such as when a user is terminated or moves to an untrusted location.

## **Enable security defaults**

---

While some of the features mentioned above require paid subscriptions, the [security defaults in Azure AD](#), which is mainly for organizations using the free tier of Azure Active Directory licensing, are sufficient to better protect the organizational identity platform, as they provide preconfigured security settings such as MFA, protection for privileged activities, and others.

## **Enable Microsoft Defender automatic attack disruption**

---

[Microsoft Defender automatic attack disruption](#) capabilities minimize lateral movement and curbs the overall impact of an attack in its initial stages.

## **Audit apps and consented permissions**

---

[Audit apps and consented permissions](#) in your organization ensure applications are only accessing necessary data and adhering to the principles of least privilege. Use [Microsoft Defender for Cloud Apps](#) and its [app governance add-on](#) for expanded visibility into cloud activity in your organization and control over applications that access your Microsoft 365 data.

Educate your organization on application permissions and data accessible by applications with respective permissions to identify malicious apps.

Enhance suspicious OAuth application investigation with the recommended approach to [investigate and remediate risky OAuth apps](#).

Enable "[Review admin consent requests](#)" for forcing new applications review in the tenant.

In addition to the recommendations above, Microsoft has published incident response playbooks for [App consent grant investigation](#) and [compromised and malicious applications investigation](#) that defenders can use to respond quickly to related threats.

## Secure Azure Cloud resources

---

Deploy MFA to all users, especially for tenant administrators and accounts with Azure VM Contributor privileges. Limit unused quota and monitor for unusual quota increases in your Azure subscriptions, with an emphasis on the resource's originating creation or modification. Monitor for unexpected sign-in activity from IP addresses associated with free VPN services on high privilege accounts. [Connect Microsoft Defender for Cloud Apps connector to ARM](#) or use Microsoft Defender for ARM

With the rise of hybrid work, employees might use their personal or unmanaged devices to access corporate resources, leading to an increased possibility of token theft. To mitigate this risk, organizations can enhance their security measures by obtaining complete visibility into their users' authentication methods and locations. Refer to the comprehensive blog post [Token tactics: How to prevent, detect, and respond to cloud token theft](#).

Check your Office 365 email filtering settings to ensure you block spoofed emails, spam, and emails with malware. Use for enhanced phishing protection and coverage against new threats and polymorphic variants. Configure Defender for Office 365 to [recheck links upon time of click](#) and [delete sent mail](#) in response to newly acquired threat intelligence. Turn on [Safe Attachments policies](#) to check attachments in inbound emails.

## Detections for related techniques

---

Leveraging its cross-signal capabilities, Microsoft Defender XDR alerts customers using Microsoft Defender for Office 365, Microsoft Defender for Cloud Apps, Application governance add-on, Microsoft Defender for Cloud, and Microsoft Entra ID Protection to detect the techniques covered in the attack through the attack chain. Each product can provide a different aspect for protection to cover the techniques observed in this attack:

### Microsoft Defender XDR

---

Microsoft Defender XDR detects threat components associated with the following activities:

- User compromised in AiTM phishing attack
- User compromised via a known AiTM phishing kit
- BEC financial fraud-related reconnaissance
- BEC financial fraud

## Microsoft Defender for Cloud Apps

---

Using Microsoft Defender for Cloud Apps [connectors](#) for [Microsoft 365](#) and [Azure](#), Microsoft Defender XDR raises the following alerts:

- Stolen session cookie was used
- Activity from anonymous IP address
- Activity from a password-spray associated IP address
- User added or updated a suspicious OAuth app
- Risky user created or updated an app that was observed creating a bulk of Azure virtual machines in a short interval
- Risky user updated an app that accessed email and performed email activity through Graph API
- Suspicious creation of OAuth app by compromised user
- Suspicious secret addition to OAuth app followed by creation of Azure virtual machines
- Suspicious OAuth app creation
- Suspicious OAuth app email activity through Graph API
- Suspicious OAuth app-related activity by compromised user
- Suspicious user signed into a newly created OAuth app
- Suspicious addition of OAuth app permissions
- Suspicious inbox manipulation rule
- Impossible travel activity
- Multiple failed login attempts

## **App governance**

---

[App governance](#) is an add-on to Microsoft Defender for Cloud Apps, which can detect malicious OAuth applications that make sensitive Exchange Online administrative activities along with [other threat detection alerts](#). Activity related to this campaign triggers the following alerts:

- Entra Line-of-Business app initiating an anomalous spike in virtual machine creation
- OAuth app with high scope privileges in Microsoft Graph was observed initiating virtual machine creation
- Suspicious OAuth app used to send numerous emails

To receive this alert, [turn on app governance for Microsoft Defender for Cloud Apps](#).

## **Microsoft Defender for Office 365**

---

Microsoft Defender for Office 365 detects threat activity associated with this spamming campaign through the following email security alerts. Note, however, that these alerts may also be triggered by unrelated threat activity. We're listing them here because we recommend that these alerts be investigated and remediated immediately.

- A potentially malicious URL click was detected

- A user clicked through to a potentially malicious URL
- Suspicious email sending patterns detected
- User restricted from sending email
- Email sending limit exceeded

## **Microsoft Defender for Cloud**

---

Microsoft Defender for Cloud detects threat components associated with the activities outlined in this article with the following alerts:

- Azure Resource Manager operation from suspicious proxy IP address
- Crypto-mining activity
- Digital currency mining activity
- Suspicious Azure role assignment detected
- Suspicious creation of compute resources detected
- Suspicious invocation of a high-risk 'Execution' operation by a service principal detected
- Suspicious invocation of a high-risk 'Execution' operation detected
- Suspicious invocation of a high-risk 'Impact' operation by a service principal detected

## **Microsoft Entra Identity Protection**

---

Microsoft Entra Identity Protection detects the threats described with the following alerts:

- Anomalous Token
- Unfamiliar sign-in properties
- Anonymous IP address
- Verified threat actor IP
- Atypical travel

## **Hunting guidance**

---

### **Microsoft 365 Defender**

---

Microsoft 365 Defender customers can run the following query to find related activity in their networks:

#### **OAuth application interacting with Azure workloads**

```

let OAuthAppId = <OAuth app ID in question>;
CloudAppEvents
| where Timestamp > ago (7d)
| where AccountId == OAuthAppId
| where AccountType == "Application"
| extend Azure_Workloads = RawEventData["operationName"]
| distinct Azure_Workloads by AccountId

```

## Password spray attempts

This query identifies failed sign-in attempts to Microsoft Exchange Online from multiple IP addresses and locations.

```

IdentityLogonEvents
| where Timestamp > ago(3d)
| where ActionType == "LogonFailed" and LogonType == "OAuth2:Token" and Application
== "Microsoft Exchange Online"
| summarize count(), dcount(IPAddress), dcount(CountryCode) by AccountObjectId,
AccountDisplayName, bin(Timestamp, 1h)

```

## Suspicious application creation

This query finds new applications added in your tenant.

```

CloudAppEvents
| where ActionType in ("Add application.", "Add service principal.")
| mvexpand modifiedProperties = RawEventData.ModifiedProperties
| where modifiedProperties.Name == "AppAddress"
| extend AppAddress = tolower(extract('\\"Address\\": \\"
(.*)\\",',1,tostring(modifiedProperties.NewValue)))
| mvexpand ExtendedProperties = RawEventData.ExtendedProperties
| where ExtendedProperties.Name == "additionalDetails"
| extend OAuthApplicationId = tolower(extract('\\"AppId\\": \\"
(.*)\\",',1,tostring(ExtendedProperties.Value)))
| project Timestamp, ReportId, AccountObjectId, Application, ApplicationId,
OAuthApplicationId, AppAddress

```

## Suspicious email events

*NOTE: These queries need to be updated with timestamps related to application creation time before running.*

```

//Identify High Outbound Email Sender
EmailEvents
| where Timestamp between (<start> .. <end>) //Timestamp from the app creation time
to few hours upto 24 hours or more
| where EmailDirection in ("Outbound")
| project
    RecipientEmailAddress,
    SenderFromAddress,
    SenderMailFromAddress,
    SenderObjectId,
    NetworkMessageId
| summarize
    RecipientCount = dcount(RecipientEmailAddress),
    UniqueEmailSentCount = dcount(NetworkMessageId)
    by SenderFromAddress, SenderMailFromAddress, SenderObjectId
| sort by UniqueEmailSentCount desc
//| where UniqueEmailSentCount > <threshold> //Optional, return only if the sender
sent more than the threshold
//| take 100 //Optional, return only top 100

//Identify Suspicious Outbound Email Sender
EmailEvents
//| where Timestamp between (<start> .. <end>) //Timestamp from the app creation time
to few hours upto 24 hours or more
| where EmailDirection in ("Outbound")
| project
    RecipientEmailAddress,
    SenderFromAddress,
    SenderMailFromAddress,
    SenderObjectId,
    DetectionMethods,
    NetworkMessageId
| summarize
    RecipientCount = dcount(RecipientEmailAddress),
    UniqueEmailSentCount = dcount(NetworkMessageId),
    SuspiciousEmailCount = dcountif(NetworkMessageId, isnotempty(DetectionMethods))
    by SenderFromAddress, SenderMailFromAddress, SenderObjectId
| extend SuspiciousEmailPercentage = SuspiciousEmailCount/UniqueEmailSentCount * 100
//Calculate the percentage of suspicious email compared to all email sent
| sort by SuspiciousEmailPercentage desc
//| where UniqueEmailSentCount > <threshold> //Optional, return only if the sender
suspicious email percentage is more than the threshold
//| take 100 //Optional, return only top 100

//Identify Recent Emails Sent by Restricted Email Sender
AlertEvidence
| where Title has "User restricted from sending email"
| project AccountObjectId //Identify the user who are restricted to send email
| join EmailEvents on $left.AccountObjectId == $right.SenderObjectId //Join
information from Alert Evidence and Email Events
| project
    Timestamp,

```

```
RecipientEmailAddress,  
SenderFromAddress,  
SenderMailFromAddress,  
SenderObjectId,  
SenderIPv4,  
Subject,  
UrlCount,  
AttachmentCount,  
DetectionMethods,  
AuthenticationDetails,  
NetworkMessageId  
| sort by Timestamp desc  
//| take 100 //Optional, return only first 100
```

## **BEC recon and OAuth application activity**



```
//High and Medium risk SignIn activity
```

```
AADSignInEventsBeta
```

```
| where Timestamp >ago (7d)
| where ErrorCode==0
| where RiskLevelDuringSignIn >= 50
| project
    AccountUpn,
    AccountObjectId,
    SessionId,
    RiskLevelDuringSignIn,
    ApplicationId,
    Application
```

```
//OAuth Application creation or modification by user who has suspicious sign in activities
```

```
AADSignInEventsBeta
```

```
| where Timestamp >ago (7d)
| where ErrorCode == 0
| where RiskLevelDuringSignIn >= 50
| project SignInTime=AccountUpn, AccountObjectId, SessionId, RiskLevelDuringSignIn,
ApplicationId, Application
| join kind=leftouter (CloudAppEvents | where Timestamp > ago(7d)
| where ActionType in ("Add application.", "Update application.", "Update application
- Certificates and secrets management "))
| extend appId = tostring(parse_json(RawEventData.Target[4].ID))
| project
    Timestamp,
    ActionType,
    Application,
    ApplicationId,
    UserAgent,
    ISP,
    AccountObjectId,
    AppName=ObjectName,
    OAuthApplicationId=appId,
    RawEventData ) on AccountObjectId
| where isnotempty(ActionType)
```

```
//Suspicious BEC reconnaissance activity
```

```
let bec_keywords = pack_array("payment", "receipt", "invoice", "inventory");
```

```
let reconEvents =
```

```
    CloudAppEvents
```

```
    | where Timestamp >ago (7d)
    | where ActionType in ("MailItemsAccessed", "Update")
    | where AccountObjectId in ("<Impacted AccountObjectId>")
    | extend SessionId = tostring(parse_json(RawEventData.SessionId))
    | project
        Timestamp,
        ActionType,
        AccountObjectId,
        UserAgent,
```

```

        ISP,
        IPAddress,
        SessionId,
        RawEventData;
reconEvents;
let updateActions = reconEvents
    | where ActionType == "Update"
    | extend Subject=toString(RawEventData["Item"].Subject)
    | where isnotempty(Subject)
    | where Subject has_any (bec_keywords)
    | summarize UpdateCount=count() by bin (Timestamp, 15m), Subject,
AccountObjectId, SessionId, IPAddress;
updateActions;
let mailItemsAccessedActions = reconEvents
    | where ActionType == "MailItemsAccessed"
    | extend OperationCount = toint(RawEventData["OperationCount"])
    | summarize TotalCount = sum(OperationCount) by bin (Timestamp, 15m),
AccountObjectId, SessionId, IPAddress;
mailItemsAccessedActions;

//SignIn to newly created app within Risky Session
AADSignInEventsBeta
| where Timestamp >ago (7d)
| where AccountObjectId in ("<Impacted AccountObjectId>") and
SessionId in ("<Risky Session Id>")
| where ApplicationId in ("<Oauth appId>") // Recently added or modified App Id
| project
    AccountUpn,
    AccountObjectId,
    ApplicationId,
    Application,
    SessionId,
    RiskLevelDuringSignIn,
    RiskLevelAggregated,
    Country

// To check suspicious Mailbox rules
CloudAppEvents
| where Timestamp between (start .. end) //Timestamp from the app creation time to
few hours, usually before spam emails sent
| where AccountObjectId in ("<Impacted AccountObjectId>")
| where Application == "Microsoft Exchange Online"
| where ActionType in ("New-InboxRule", "Set-InboxRule", "Set-Mailbox", "Set-
TransportRule", "New-TransportRule", "Enable-InboxRule", "UpdateInboxRules")
| where isnotempty(IPAddress)
| mvexpand ActivityObjects
| extend name = parse_json(ActivityObjects).Name
| extend value = parse_json(ActivityObjects).Value
| where name == "Name"
| extend RuleName = value
| project Timestamp, ReportId, ActionType, AccountObjectId, IPAddress, ISP, RuleName

```

```
// To check any suspicious Url clicks from emails before risky signin by the user
UrlClickEvents
| where Timestamp between (start .. end) //Timestamp around time proximity of Risky
signin by user
| where AccountUpn has "<Impacted User's UPN or Email address>" and ActionType has
"ClickAllowed"
| project Timestamp,Url,NetworkMessageId
```

```
// To fetch the suspicious email details
EmailEvents
| where Timestamp between (start .. end) //Timestamp lookback to be increased
gradually to find the email received
| where EmailDirection has "Inbound"
| where RecipientEmailAddress has "<Impacted User's UPN or Email address>" and
NetworkMessageId == "<NetworkMessageId from UrlClickEvents>"
| project SenderFromAddress,SenderMailFromAddress,SenderIPv4,SenderFromDomain,
Subject,UrlCount,AttachmentCount
```

```
// To check if suspicious emails sent for spamming (with similar email subjects, urls
etc.)
```

```
EmailEvents
| where Timestamp between (start .. end) //Timestamp from the app creation time to
few hours upto 24 hours or more
| where EmailDirection in ("Outbound","Intra-org")
| where SenderFromAddress has "<Impacted User's UPN or Email address>" or
SenderMailFromAddress has "<Impacted User's UPN or Email address>"
| project RecipientEmailAddress,RecipientObjectId,SenderIPv4,SenderFromDomain,
Subject,UrlCount,AttachmentCount,NetworkMessageId
```

## Microsoft Sentinel

---

Microsoft Sentinel customers can use the TI Mapping analytics (a series of analytics all prefixed with 'TI map') to automatically match the malicious domain indicators mentioned in this blog post with data in their workspace. If the TI Map analytics are not currently deployed, customers can install the Threat Intelligence solution from the [Microsoft Sentinel Content Hub](#) to have the analytics rule deployed in their Sentinel workspace.

Analytic rules:

Hunting queries:

## Learn more

---

For the latest security research from the Microsoft Threat Intelligence community, check out the Microsoft Threat Intelligence Blog: <https://aka.ms/threatintelblog>.

To get notified about new publications and to join discussions on social media, follow us on X (formerly Twitter) at <https://twitter.com/MsftSecIntel>.

To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape, listen to the Microsoft Threat Intelligence podcast: <https://thecyberwire.com/podcasts/microsoft-threat-intelligence>.

## Related Posts

---



### **Microsoft Incident Response lessons on preventing cloud identity compromise**

In real-world customer engagements, Microsoft Incident Response (Microsoft IR) sees combinations of issues and misconfigurations that could lead to attacker access to customers' Microsoft Entra ID tenants. Effective protection of a customer's Entra ID tenant is less challenging than protecting an Active Directory deployment but does require governance and monitoring. Reducing risk and exposure of your most privileged accounts plays a critical role in preventing or detecting attempts at tenant-wide compromise.



## **Defending new vectors: Threat actors attempt SQL Server to cloud lateral movement**

Microsoft security researchers recently identified an attack where attackers attempted to move laterally to a cloud environment through a SQL Server instance. The attackers initially exploited a SQL injection vulnerability in an application within the target's environment to gain access and elevated permissions to a Microsoft SQL Server instance deployed in an Azure Virtual Machine (VM). The attackers then used the acquired elevated permission to attempt to move laterally to additional cloud resources by abusing the server's cloud identity.

