



WHITE PAPER

Remediation and Hardening Strategies for Microsoft 365 to Defend Against UNC2452

Version 1.1

Table of Contents

- Overview** 3
 - Background..... 3
 - Goals and Objectives..... 3
- Attacker Tactics, Techniques and Procedures (TTPs)** 4
- Remediation Strategy** 4
 - Remediation Sequencing..... 5
- AD FS Attack Mitigation** 6
- Attack Technique: Golden SAML Attack**7
 - Active Directory Federation Service Overview7
 - Technique..... 9
 - Remediation.....10
 - Hardening 11
 - Detection.....18
- Microsoft 365 Attack Mitigation**19
- Attack Technique: Modify Trusted Domains**.....20
 - Azure AD Federated Domains Overview.....20
 - Technique.....20
 - Detection.....20
 - Remediation..... 23
- Attack Technique: Abuse Azure AD Privileged Roles** 24
 - Azure Active Directory Privileged Roles Overview 24
 - Technique.....24
 - Remediation..... 24
- Attack Technique: Hijack Azure AD Applications**..... 26
 - Azure Applications and Service Principals Overview 26
 - Technique..... 27
 - Detection..... 27
 - Remediation..... 29
- Attack Technique: Modify Mailbox Folder Permissions**..... 33
 - Mailbox Folder Permission Overview 33
 - Technique..... 34
 - Remediation..... 34
 - Detection..... 36
- Microsoft 365 Hardening** 38
- Conclusion**.....46

Overview

Background

In December 2020, FireEye uncovered and publicly disclosed a widespread campaign conducted by the threat group we track as [UNC2452](#). In some, but not all, of the intrusions associated with this campaign where Mandiant has visibility, the attacker used their access to on-premises networks to gain unauthorized access to the victim's Microsoft 365 environment.

Goals and Objectives

[UNC2452](#) and other threat actors have used several methodologies to move laterally from on-premises networks to the cloud, specifically Microsoft 365. This paper will help organizations understand these techniques used by UNC2452, how to proactively harden their environments, and how to remediate environments where similar techniques have been observed.

It is important to note that there is no formal security boundary between on-premises networks and cloud services provided by Microsoft 365. If an organization discovers evidence of targeted threat actor activity in their on-premises network, a thorough review of the cloud environment is often necessary as well.

Organizations can use the [Azure AD Investigator auditing script, available from the FireEye GitHub repository](#), to check their Microsoft 365 tenants for indicators relative to the techniques detailed throughout this paper. The script will alert administrators and security practitioners to artifacts that may require further review to determine if they are truly malicious or part of legitimate activity.

Attacker Tactics, Techniques and Procedures (TTPs)

Mandiant has observed UNC2452 and other threat actors moving laterally to the Microsoft 365 cloud using a combination of five primary techniques:

1. Steal the Active Directory Federation Services (AD FS) token-signing certificate and use it to forge tokens for arbitrary users (sometimes described as [Golden SAML](#)). This would allow the attacker to authenticate into a federated resource provider (such as Microsoft 365) as any user, without the need for that user's password or their corresponding multi-factor authentication (MFA) mechanism.
2. Modify or add trusted domains in Azure AD to add a new federated Identity provider (IdP) that the attacker controls. This would allow the attacker to forge tokens for arbitrary users and has been described as an [Azure AD backdoor](#).
3. Compromise the credentials of on-premises user accounts that are synchronized to Microsoft 365 and are assigned high privileged directory roles, such as Global Administrator or Application Administrator.
4. Hijack an existing Microsoft 365 application by adding a rogue credential to it in order to use the legitimate permissions assigned to the application, such as the ability to read email, send email as an arbitrary user, access user calendars, etc., while bypassing MFA.
5. Modify the permissions of folders in a victim mailbox (such as the inbox) to make its contents readable by any other user in the victim's Microsoft 365 environment.

Remediation Strategy

Developing a successful strategy to eradicate UNC2452 access to a victim’s environment is contingent upon many factors, so each organization must develop their own unique plan. Two of the most important factors are the actions taken by the attacker and the specifics of the victim’s environment.

To help victims plan their own strategy, Table 1 provides a high-level overview of techniques related to Microsoft 365 with links to corresponding remediation and hardening steps.

Table 1. Overview of covered TTPs.			
Technique	MITRE ATT&CK	Remediation	Hardening
Attack Technique: Golden SAML Attack	TA0006 T1552 T1552.004 T1199	Step 1: Issue new AD FS Certificates Step 2: Revoke Microsoft 365 Refresh Tokens	Step 1: AD FS Service Account - Best Practices Step 2: AD FS Logging and Auditing Step 3: AD FS Servers - Account Access Restrictions and Inbound Connectivity Hardening
Attack Technique: Modify Trusted Domains	T1550	Step 1: Review Configured Domains and Trusts Within Microsoft 365 and Remove Untrusted Domains	Step 1: Filter accounts synced to Azure Active Directory
Attack Technique: Abuse Azure AD Privileged Roles	TA0006 T1552	Step 1: Review and Configure Cloud-Only Accounts for Privileged Role Holders Step 2: Rotate All Passwords for Cloud-Only Accounts Step 3: Invalidate Refresh Tokens for Each Cloud-Only Account	Step 2: Limit Privileged Users to Trusted IPs Step 3: Enhance Mailbox Auditing Step 4: Review Azure Application and Service Principal Permissions Step 5: Enforce multi-factor authentication (MFA) for Accounts
Attack Technique: Hijack Azure AD Applications	T1114 T1114.002 T1098.001	Step 1: Review and Remediate Keys Associated with Service Principals and Applications Step 2: Invalidate All Existing Refresh Tokens	Step 6: Review all registered MFA devices Step 7: Review Partner (Cloud Service Provider) Relationships
Attack Technique: Modify Mailbox Folder Permissions	T1114 T1114.002 T1098	Step 1: Review and remediate Mailbox Folder Permissions where read privileges have been granted to Anonymous or Default User	Step 1: Review overly permissive mailbox Folder Permissions Step 2: Enable mailbox audit logging action: Update Folder Permissions

Remediation Sequencing

Timing and sequencing are critical for the successful execution of a remediation plan, especially if an attacker remains active during the containment and eradication timeframes. To maximize the probability of fully eradicating this threat actor from hybrid Microsoft 365 environments, Mandiant recommends sequencing the remediation plan to:

- First, regain control of the on-premises environment. This environment houses and protects the various secrets that attackers often abuse to gain initial access and attack cloud environments.
- Next, rotate secrets and credentials to regain control of the Microsoft 365 environment and fully eradicate all attacker access.

To accomplish these two goals, consider executing a plan similar to the following, in order of operation. For actions directly related to the attacker techniques discussed in this white paper, we included internal jump links to more detailed instructions. For remaining actions, we have linked to public sources where appropriate:

1. Implement short term hardening measures to mitigate risk of similar attacks
2. Eliminate any attacker remote access to the on-premises environment, which could include:
 - a. Implementing network blocks and DNS sinkholes
 - b. Removing remote access backdoors, including impacted SolarWinds software
 - c. Un-enrolling attacker multi-factor authentication (MFA) tokens
 - d. Rotating secrets associated with remote access MFA token generation
3. Rotate impacted on-premises credentials, which could include:
 - a. [Rotating the Kerberos ticket granting ticket account \(krbtgt\) credentials \(twice\)](#)
 - b. Rotating credentials for “Tier-0” accounts in Active Directory
 - c. Rotating credentials and/or disabling compromised accounts
 - d. Rotating credentials for other Active Directory accounts
 - e. Rotating credentials for local administrator accounts
 - f. Rotating credentials for cloud connector accounts
 - i. [On-premises Azure AD DS connector account](#)
 - ii. [Azure AD connector account](#)
 - iii. [On-premises ADSync Service Account \(that runs AD Connect services on servers\)](#)
 - iv. If Azure seamless single sign-on is utilized, rotate the decryption key for the on-premises [AZUREADSSOACC computer account](#)
 - g. Rotating other internal credentials or secrets
4. [Rotate the AD FS token-signing and token-decrypting certificates used with SAML tokens \(twice\)](#)
5. [Rotate credentials for impacted cloud accounts](#)
6. [Remove attacker created identity providers and custom domains](#)
7. [Remove attacker certificates and passwords from applications and service principals](#)
8. Revoke all existing refresh tokens for Microsoft 365
Note: this will force all users to re-enter their credentials to utilize Microsoft 365 services through clients such as Outlook and Teams.
9. [Review and Remediate Mailbox Folder Permissions](#)

AD FS Attack Mitigation

ATTACK TECHNIQUE:

Golden SAML Attack

Active Directory Federation Service Overview

Active Directory Federation Services (AD FS) provides an on-premises authentication workflow for cloud-based resources. While other authentication workflows also exist, AD FS is a popular option for enterprises that want to maintain control of their own authentication. In this workflow, AD FS is an Identity Provider (IdP) because it facilitates user identity information to cloud-based resources and applications.

Microsoft AD FS uses a “claim-based” identity model to identify roles, permissions, or groups associated with a user authentication. A claim can be any piece of information about the requesting user’s identity, such as a group they belong to, or whether an MFA authentication was performed during login. For example, if MFA occurred during the AD FS login, then the claim shown in Figure 1 would be provided by AD FS.

Multiple claims associated with the authentication are packaged into an XML document, known as a SAML token, and digitally signed to ensure verifiability and to prevent unauthorized modification. When Microsoft 365 receives a SAML token issued by an AD FS service, it performs the following steps to ensure its validity before providing access to the resource (listed in no particular order):

- Verify that the digital signature of the SAML token is valid and uses a known keypair
- Verify the SAML token is constructed properly and complies with the standard
- Verify that the SAML token has not expired
- Verify that the token was issued by a source that Microsoft 365 expects (the issuer)
- Verify that the unique identifier for the user (the ImmutableID) matches a user in Azure AD
- Evaluate conditional access policies, if any

Figure 1.

Example SAML claim for Microsoft 365.

```
<saml:AttributeValue>http://schemas.microsoft.com/claims/multipleauthn</saml:AttributeValue>
```

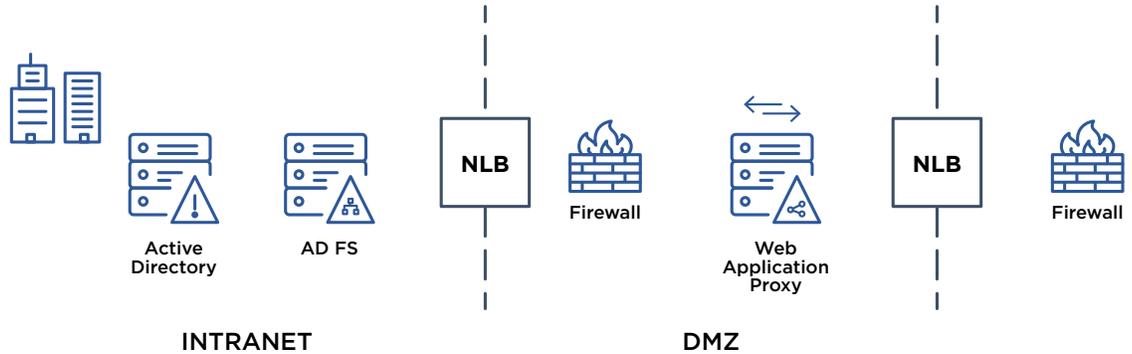
In this authentication scheme, two vital pieces of data are the digital signature and the ImmutableID. The ImmutableID is a globally unique identifier (GUID) associated with a user and stored in Azure Active Directory (Azure AD). A copy of this GUID is also stored in the on-premises Active Directory as the [ms-DS-ConsistencyGuid](#) attribute of the User object.

This attribute is viewable by any authenticated user in both Azure AD and on premises AD.

The foundation of the security of AD FS is the confidentiality of the token-signing certificate, which is used to digitally sign SAML tokens issued by the AD FS service. Exposure of the token-signing certificate and private key, paired with any compromised AD account (to query users’ ImmutableIDs), would allow an attacker to issue their own forged but verifiable SAML tokens. Taking it a step further, a one-time extraction of all ImmutableIDs and the token-signing certificate with private key would allow the attacker to issue SAML tokens for any user in the organization regardless of their password and any password resets that occur afterwards. The attacker can insert any claim they want into the SAML token. This could include the claim detailed in Figure 1, which states that the user performed MFA at the AD FS service. **This forged SAML token will bypass MFA requirements imposed by the cloud application.**

The locations of the token-signing certificate and private key depend on how an organization has chosen to implement AD FS. On-premises AD FS infrastructure can consist of either a single or multiple AD FS servers, a database instance, and either single or multiple AD FS web application proxy server(s) housed in a DMZ (Figure 2). Most organizations will likely leverage an AD FS server farm, using a local Windows Internal Database (WID).

Figure 2.
<https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/best-practices-securing-ad-fs>



In the default installation of AD FS, the token-signing certificate is automatically generated and valid for one year. The AD FS service will handle the certificate rollover process. The following points detail how the token-signing certificate is stored on the AD FS server:

- The token-signing certificate is stored as a base64 encoded encrypted blob within the database used by the AD FS servers.
- The token-signing certificate is protected using a Windows technology called Distributed Key Manager (DKM). AD FS uses a key derivation function to derive an AES key that is used to decrypt the token-signing certificate stored in the AD FS database.

This data is stored within an Active Directory attribute **CN=ADFS,CN=Microsoft,CN=Program Data,DC=<domain>,DC=<suffix>**. In addition to obtaining the token-signing certificate, an attacker would also need to obtain the value from Active Directory for decrypting the token-signing certificate. By default, this value is only readable by the AD FS service account and Domain Administrators (Figure 3).

Note: If multiple AD FS certificate containers are present in Active Directory, from an AD FS server, the PowerShell command **(Get-AdfsProperties).CertificateSharingContainer** can be used to identify the container that will be used by DKM.

Figure 3.
 Active Directory - AD FS Container Attributes.

```
PS C:\Windows\system32> (Get-AdfsProperties).certificatesharingcontainer
CN=F0839c4f-6610-41b3-a4b4-9a0f06c6e559,CN=ADFS,CN=Microsoft,CN=Program Data,
```

Name	Class
CN=CryptoPolicy	contact
CN=d26e78c2-a63e-4745-8e7b-dea1add2ed40	container
CN=f0839c4f-6610-41b3-a4b4-9a0f06c6e559	container

The AD FS service also makes use of other certificates that are out of scope for this whitepaper. For example, each cloud application can define a token-decrypting certificate, which is used to encrypt the SAML token as it is transmitted to the cloud resource. This option is not widely used.

Note: If an AD FS server farm model is leveraged, the same token-signing and decryption certificates are used by all the AD FS servers.

The default configuration for AD FS uses a built-in MSSQL database service called Windows Internal Database (WID). The WID service is only accessible from the local host via a specific named pipe and does not listen on the standard TCP port 1433.

The WID service stores the backing files for the AD FS database in the following locations:

- **C:\Windows\WID\Data\AdfsConfiguration.mdf**
- **C:\Windows\WID\Data\AdfsConfiguration_log.ldf**

A default configuration of AD FS using the WID service will generate a database table named **IdentityServerPolicy.ServiceSettings** to store the token-signing certificate.

Technique

Once UNC2452 had gained on-premises access to an organization's environment, Mandiant observed them targeting on-premises AD FS servers with the goal of obtaining the token-signing certificate to forge SAML tokens. There are two steps an attacker must take to obtain the token-signing certificate and private key.

Figure 4.

ADFind command to query the secret DKM value.

```
C:\windows\system32\cmd.exe /C c:\windows\temp\srv.exe -h -sc fo:* -b "CN=ADFS,CN=Microsoft,CN=Program Data,DC=acme,DC=com"
```

First, the attacker must obtain the encrypted token-signing certificate. The threat actor has used a number of techniques to extract the encrypted blob from the **IdentityServerPolicy.ServiceSettings** table. Mandiant has observed techniques including:

- **Stealing the backup files for the database:** Connecting to the server via SMB and copying the database file **AdfsConfiguration.mdf** off the server
- **Querying the database via PowerShell:** Connecting to the server to execute PowerShell that then connects to and queries the WID service for the table rows.
 - SELECT **ServiceSettingsData** from **IdentityServerPolicy.ServiceSettings**
- **Querying the database via custom executable:** Writing a custom executable to disk and executing it. The executable queried the WID for the table row and wrote the output to disk.

Next, the attacker must obtain the secret DKM value from Active Directory to decrypt the token-signing certificate. Mandiant has observed the threat actor using open-source tools such as ADFind to query Active Directory for the DKM value used to decrypt the encrypted token-signing certificate. Figure 4 details an ADFind query used by UNC2452 to obtain this value.

With the certificate and DKM value, the attacker now has the data they need to forge SAML tokens and access cloud-based resources, while bypassing MFA requirements and the need to know a user's password (GoldenSAML).

Remediation

If an attacker is able to connect to an on-premises AD FS server and has credentials for either the service account that runs the AD FS service or an account that has local administrative permissions on AD FS servers, this provides the threat actor with the access needed to extract the token-signing certificate.

If an organization suspects an attacker has accessed their AD FS token-signing certificate, two critical steps are required to remediate and protect against a GoldenSAML attack.

1. Issue new certificates on the AD FS server(s) and synchronize them to Azure AD (and any other cloud applications that use AD FS for authentication).
 - Rotate the token-signing AD FS certificate in rapid succession **twice**.
 - By default, if the certificate is only rotated once, a copy of the previous (potentially compromised)

certificate will still be resident in Azure AD, and can still be used to forge SAML tokens. To fully ensure that only new and trusted certificates are available, a “double-tap” is required.

2. Immediately revoke all existing refresh tokens for the Microsoft 365 tenant.

Step 1: Issue new AD FS Certificates

Note: The certificate rollover will temporarily impact single sign-on communications with applications that use AD FS as an IdP.

To view the properties related to the AD FS certificates, reference the PowerShell commands noted in Figure 5.

To rotate the AD FS token-signing and token-decrypting certificates using PowerShell, the commands noted in Figure 6 can be utilized. The commands must be run from the primary AD FS server (if an AD FS server farm is utilized).

Figure 5.
Command to review the properties of AD FS certificates.

```
Get-ADFSCertificate -CertificateType Token-Signing
Get-ADFSCertificate -CertificateType Token-Decrypting
```

Figure 6.
Commands to rotate AD FS token-signing and token-decrypting certificates.

```
** 1st Iteration **

Set-ADFSProperties -AutoCertificateRollover $true

Update-AdfsCertificate -CertificateType Token-Decrypting -Urgent
Update-AdfsCertificate -CertificateType Token-Signing -Urgent

Set-ADFSProperties -AutoCertificateRollover $false

Connect-MsolService
Get-MsolFederationProperty -DomainName <domain>
Update-MsolFederatedDomain -DomainName <domain>
Get-MsolFederationProperty -DomainName <domain>

** 2nd Iteration **

Set-ADFSProperties -AutoCertificateRollover $true

Update-AdfsCertificate -CertificateType token-signing
Update-AdfsCertificate -CertificateType token-decrypting

Set-ADFSProperties -AutoCertificateRollover $false

Connect-MsolService
Get-MsolFederationProperty -DomainName <domain>
Update-MsolFederatedDomain -DomainName <domain>
Get-MsolFederationProperty -DomainName <domain>
```

Once completed, ensure that the output from the **Get -MsolFederationProperty -DomainName** command only includes the certificate thumbprints and validity dates that match the recently rotated certificates, and that the certificate thumbprint from the previously (potentially compromised) certificates are not present in Azure AD.

If an organization maintains backup copies of the token-signing and token-decrypting certificates, the certificate backups should not be stored on the local AD FS servers and should be secured and password-protected using a separate enterprise vaulting / backup-storage technology (preferably offline storage).

Step 2: Revoke Microsoft 365 Refresh Tokens

After rotating the AD FS certificates, organizations should immediately revoke all existing refresh tokens for their Microsoft 365 tenant. While this will result in users being required to reauthenticate, this is a prudent step to ensure that all logged in users authenticate to Microsoft 365 using the new and trusted SAML tokens. Using an account that has Microsoft 365 **Global Administrator** permissions, the commands noted in Figure 7 can be used to invalidate existing refresh tokens.

Figure 7.

PowerShell commands to connect to Azure AD and revoke existing refresh tokens.

```
Connect-AzureAD  
Get-AzureADuser -all $true | Revoke-AzureADUserAllRefreshToken
```

Hardening

Proactive hardening steps can be leveraged to secure on-premises AD FS infrastructure. While the scope of generalized on-premises hardening workstreams can be cumbersome and often represent multiple layers of defenses, the most prudent steps related to AD FS hardening and mitigating the previously noted attacker techniques are noted below.

- **Step 1:** Configure a Group Managed Service Account (gMSA) for AD FS services
 - Alternatively, configure a dedicated on-premises account that is only leveraged as an AD FS service account and has restricted access rights
- **Step 2:** Review AD FS Logging and Auditing Settings
- **Step 3:** Review Account and Network Access Restrictions for AD FS Servers

Step 1: AD FS Service Account – Best Practices

- Since AD FS v3.0 (Windows Server 2012 R2+), a group managed service account (gMSA) can be configured for running AD FS services. The benefits of leveraging a gMSA include a complex (120 character) password that is automatically managed and rotated on a pre-defined frequency (30 days by default). For additional information, reference <https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/configure-a-federation-server>.
- If a gMSA cannot be leveraged, a dedicated (non-privileged) service account should be created in Active Directory for running AD FS services. This account should not be used on any additional systems or services. Additionally, the account should be configured with a strong (> 25 character) password and rotated on a periodic basis. For additional information, reference <https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/manually-configure-a-service-account-for-a-federation-server-farm>.

Note: If an organization needs to change the existing AD FS service account to leverage a gMSA (rather than a static service account), this process can be complex and may require assistance from Microsoft support. For specific information related to this process, reference:

- <https://gallery.technet.microsoft.com/scriptcenter/Active-Directory-ddb67df0>
- <https://social.technet.microsoft.com/Forums/windowsserver/en-US/8f558762-f92c-4803-916c-cc36ecc7c988/adfs-2016-change-service-account-to-gmsa?forum=ADFS#75f9b050-ad0b-461a-92a4-796180d6c6d2>
- <https://github.com/Microsoft/adfsToolbox/tree/master/serviceAccountModule>

Step 2: AD FS Logging and Auditing

AD FS utilizes two primary event logs on servers that run AD FS services: Admin log and Tracing/Debug log.

With the default settings, the AD FS service registers its own Windows event log group called “AD FS”. The “Admin” event log in this group records high-level events related to the AD FS service status, service requests and overall service health. If debug logging is enabled, an additional Windows event log group called “AD FS Tracing” will be present—housing the “Debug” event log.

Note: Mandiant does not recommend enabling debug logging unless there is a specific need to perform troubleshooting for AD FS services. Enabling debug logging will generate a large volume of events in a short timeframe, and can greatly impact operational performance of the AD FS server(s).

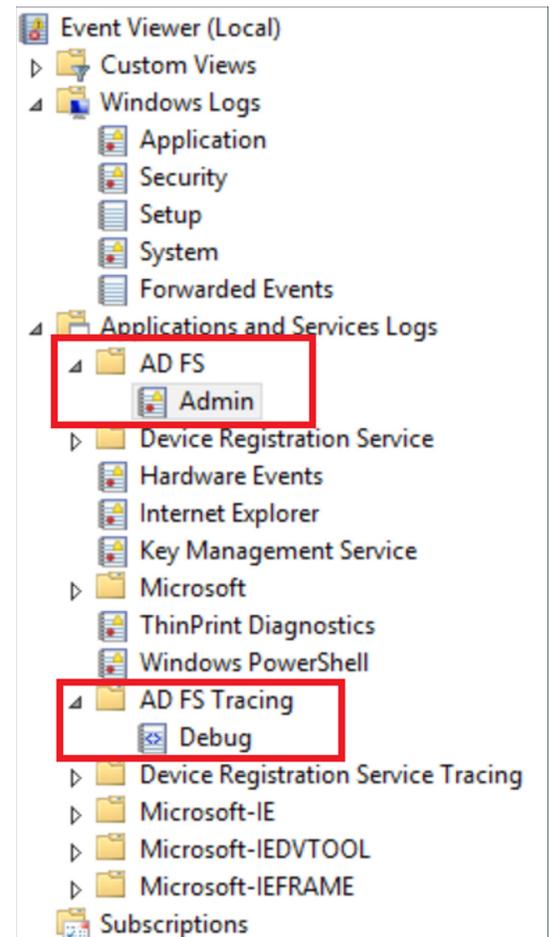
Organizations can verify their current AD FS audit level by using the **Get-AdfsProperties** PowerShell cmdlet. AD FS logging levels can be configured and adjusted using the **Set-AdfsProperties** PowerShell cmdlet.

Note: For an AD FS server farm, any changes to AD FS audit levels will need to be configured on each server within the farm.

For AD FS services running on Windows Server 2012R2, there are six configurable audit levels:

- Errors
- FailureAudits
- SuccessAudits
- Information
- Verbose
- Warnings

Figure 8. AD FS audit logs.



Mandiant's recommended settings for configuring the audit level for AD FS services running on Windows Server 2012R2 are shown in Figure 9.

For AD FS services running on Windows Server 2016+, there are three configurable audit levels:

- **Basic (default):** no more than 5 events will be logged for a single request
- **Verbose:** all events will be logged
- **None:** auditing is disabled (not recommended)

Mandiant's recommended settings for configuring the audit level for AD FS services running on Windows Server 2016+ is shown in Figure 10.

Note: While Mandiant's recommendation of including Verbose auditing will provide the most information related to AD FS events, this setting will likely cause event logs to rotate faster and should be utilized in conjunction with event log forwarding to a SIEM.

Figure 9.

Recommended AD FS audit level for AD FS services on Windows Server 2012R2.

```
Set-AdfsProperties -LogLevel Errors,FailureAudits,Information,Verbose,SuccessAudits,Warnings
```

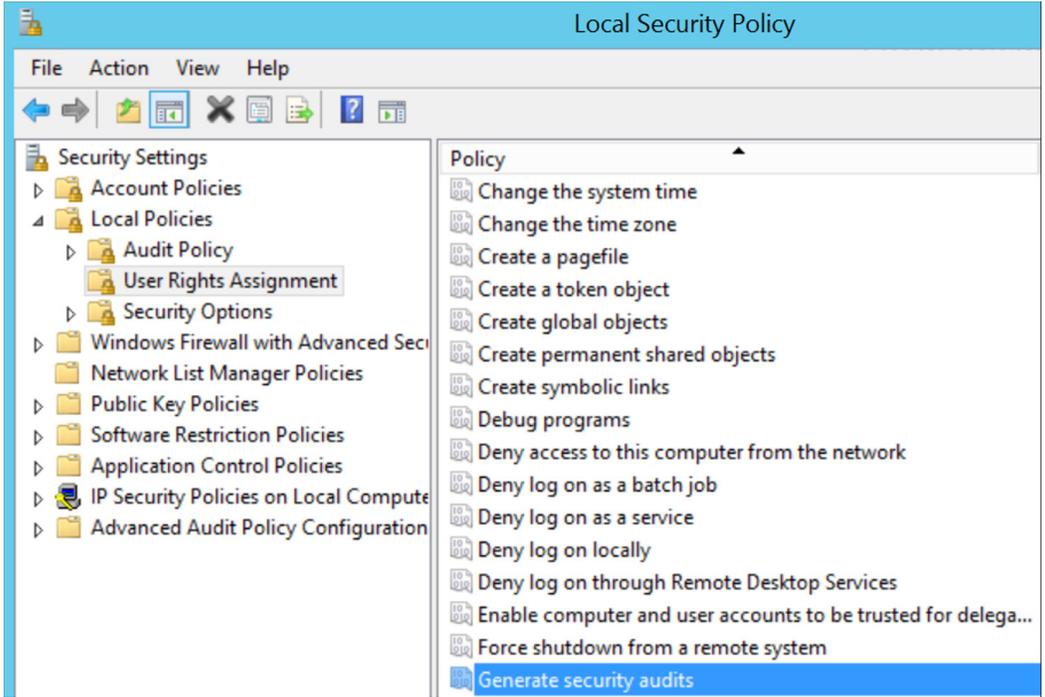
Figure 10.

Recommended AD FS audit level for AD FS services on Windows Server 2016+.

```
Set-AdfsProperties -AuditLevel Verbose
```

Figure 11.

Local Security Policy - Generate Security Audits configuration.



Enable AD FS Security and Service Account Auditing

In addition to audit levels, Mandiant recommends enabling security auditing in AD FS. Security auditing provides visibility to authentication events performed through the AD FS service, including issued claims, security token validation failures, IP addresses for requests and user agent information. By default, both "Success audits" and "Failure audits" are **not** enabled, which can severely affect monitoring and investigative efforts.

To enable security auditing for AD FS events, implement the following steps:

1. On each AD FS server - Navigate to Local Security Policy > Security Settings > Local Policies > User Rights Assignment > Generate security audits (Figure 11).
 - Local Security Setting tab > verify that the AD FS service account is listed. If not listed, add the account.

- To record audits generated by the AD FS service within the Security event log, on each AD FS server - Navigate to Local Security Policy > Security Settings > Advanced Audit Policy Configuration > System Audit Policies > Object Access
 - Audit Application Generated tab > Select both "Success" and "Failure" (Figure 12).
 - This can also be accomplished using an elevated command prompt, and running the command noted in Figure 13.

Figure 12.

Local Security Policy – Audit Application Generated configuration.

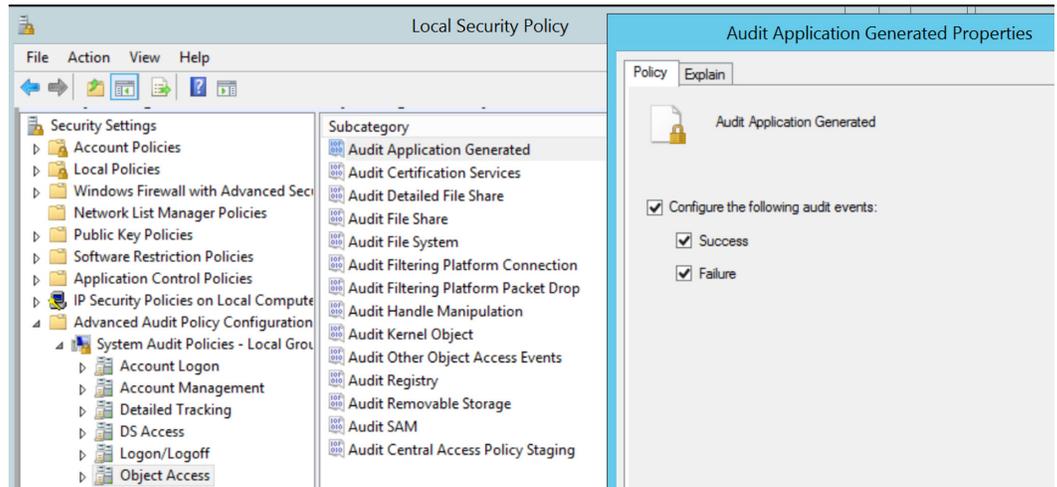


Figure 13.

Command to enable AD FS service account auditing.

```
auditpol.exe /set /subcategory:"Application Generated" /failure:enable /success:enable
```

- For the AD FS Server Farm - navigate to Start > Programs > Administrative Tools, > AD FS Management
 - Actions pane > Edit Federation Service Properties
 - Federation Service Properties > Events tab
 - Select **Success Audits and Failure Audits** (Figure 14)

Note: Enabling AD FS service auditing can increase the volume of events logged on AD FS servers. It is recommended that events captured within event logs on AD FS servers be forwarded to a SIEM, for enhanced correlation and offline storage for historical review purposes.

Once AD FS auditing is configured, any successful changes to the AD FS service configuration will result in Event ID 307 (**ConfigurationChangeSuccessAudit**) or Event ID 309 (**WmiConfigurationChangeSuccessAudit**) being recorded within the Security event log on the AD FS server(s). For correlation, Event ID 510 will contain additional information related to the configuration changes that were enforced (Figure 15).

Figure 14. Federation Service Properties - Recommended Auditing Actions (per AD FS Farm).

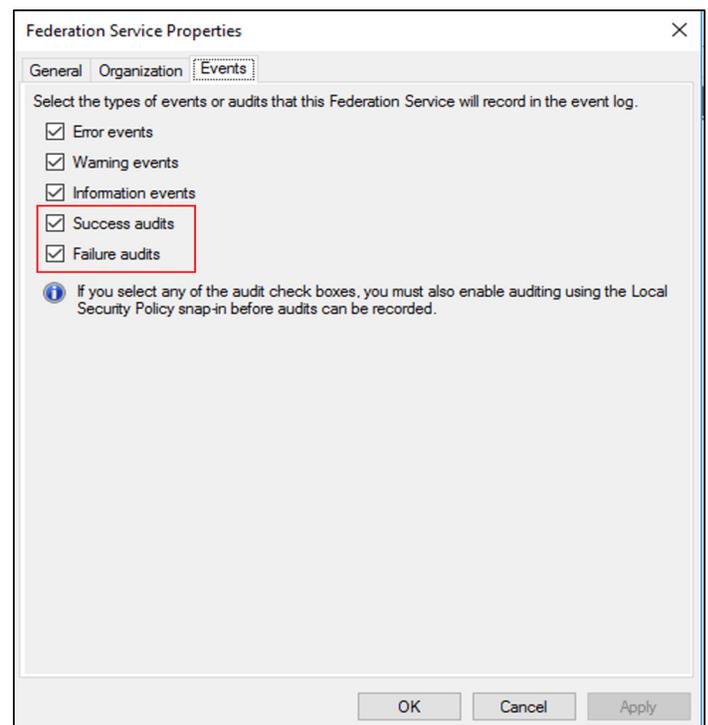
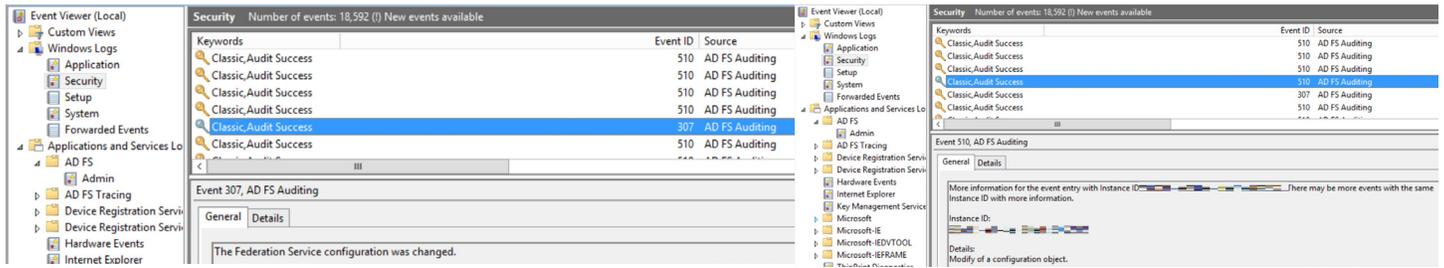


Figure 15. Event IDs related to AD FS configuration setting modifications.



For additional information related to AD FS logging and auditing—in addition to specific Event IDs (per Operating System platform) which can be leveraged for troubleshooting and investigative purposes, reference:

<https://adfs-help.microsoft.com/AdfsEventViewer/GetAdfsEventList>

Enable Domain Auditing for AD FS Distributed Key Manager Access Requests

The DKM secret value is required to decrypt the AD FS token-signing certificate and is stored in Active Directory. The Active Directory container should only be accessed periodically by the AD FS service account and AD FS service process. To detect anomalous access requests targeting the AD FS Distributed Key Manager (DKM) container in Active Directory, the following monitoring should be configured.

1. On Domain Controllers, ensure that the **“Audit Directory Service Access”** subcategory (within the **“DS Access”** category) is configured for auditing. This configuration can be enforced using settings defined within a Group Policy Object (GPO) linked to Domain Controllers (Figure 16).
2. Ensure that auditing for **“Read All Properties”** is configured for the **CN=ADFS,CN=Microsoft,CN=Program Data,DC=<domain>,DC=<suffix>** parent (and child) containers in Active Directory. This can be accomplished using ADSIEdit (Figure 17).

Figure 16.

GPO configuration example for enforcing “Directory Service Access” auditing for Domain Controllers.

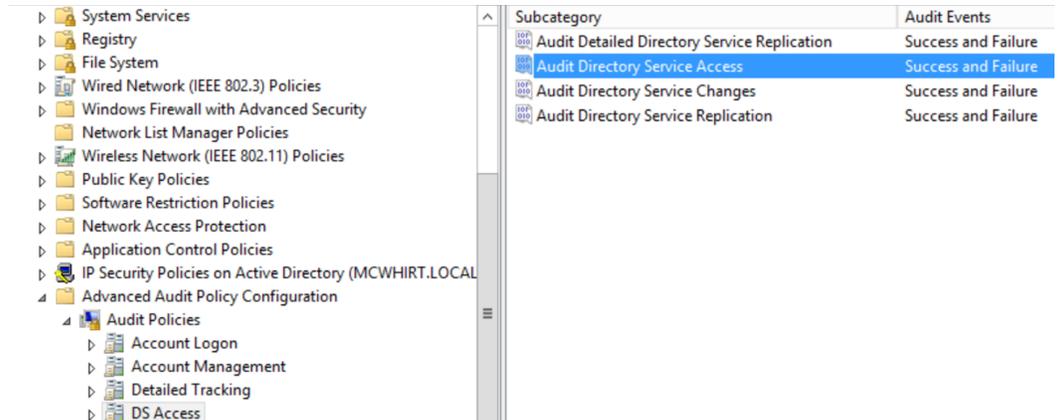
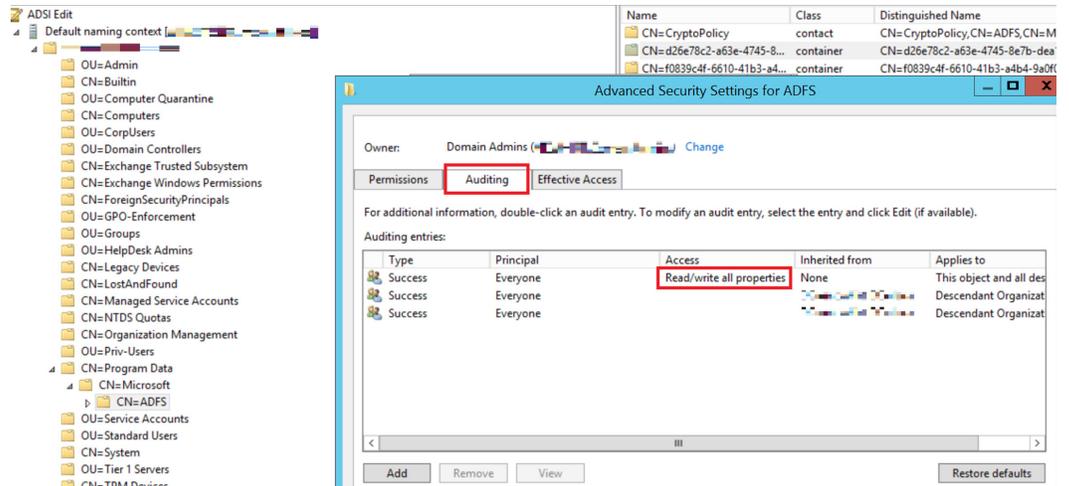


Figure 17.

Configuring auditing for the ADFS container using ADSIEdit.



3. Within Security event logs on Domain Controllers, monitor for Event ID 4662 containing {8d3bca50-1d7e-11d0-a081-00aa006c33ed} in the Properties field of the event:

- **ObjectName** contains the GUIDs of the AD FS DKM Containers for the token-signing and token-decrypting certificates (can be found using ADSIEdit and navigating to **CN=ADFS,CN=Microsoft,CN=Program Data,DC=<domain>,DC=<suffix>**)
- **ObjectName** contains the path of **CN=CryptoPolicy,CN=ADFS,CN=Microsoft,CN=Program Data,DC=<domain>,DC=<suffix>**
- **Properties** contains the **thumbnailPhoto** Attribute GUID ({8d3bca50-1d7e-11d0-a081-00aa006c33ed}) which stores the DKM Master Key

Using a privileged account that is granted access to the certificate sharing container, the PowerShell command in Figure 18 can be used to test the detection.

Figure 18.

Command to generate a read event of token-signing certificate attribute.

```
(Get-ADObject -filter 'ObjectClass -eq "Contact" -and name -ne "CryptoPolicy" -searchBase "CN=ADFS,CN=Microsoft,CN=Program Data,DC=<domain>,DC=<suffix>" -Properties thumbnailPhoto). thumbnailPhoto
```

Figure 19. Example Event ID 4662 - recorded when accessing the DKM container in AD using PowerShell.



Note: When an AD FS server is rebooted or the AD FS service is restarted, the service account legitimately running AD FS services will access the containers in Active Directory for certificate decryption using DKM. To further enhance monitoring and reduce false positives,

additional correlation on AD FS servers related to the detections noted above can be leveraged, including:

- System Event Log - Event ID 7036
- System Event Log - Event ID 1074

Figure 20.

System Event Log - Event ID 7036.

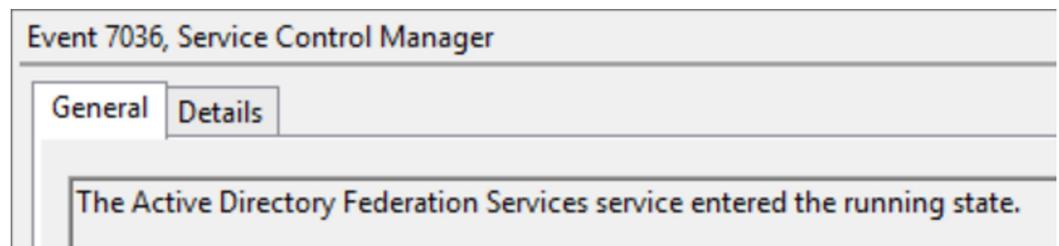
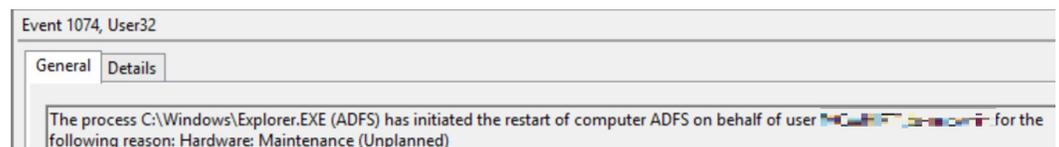


Figure 21.

System Event Log - Event ID 1074.



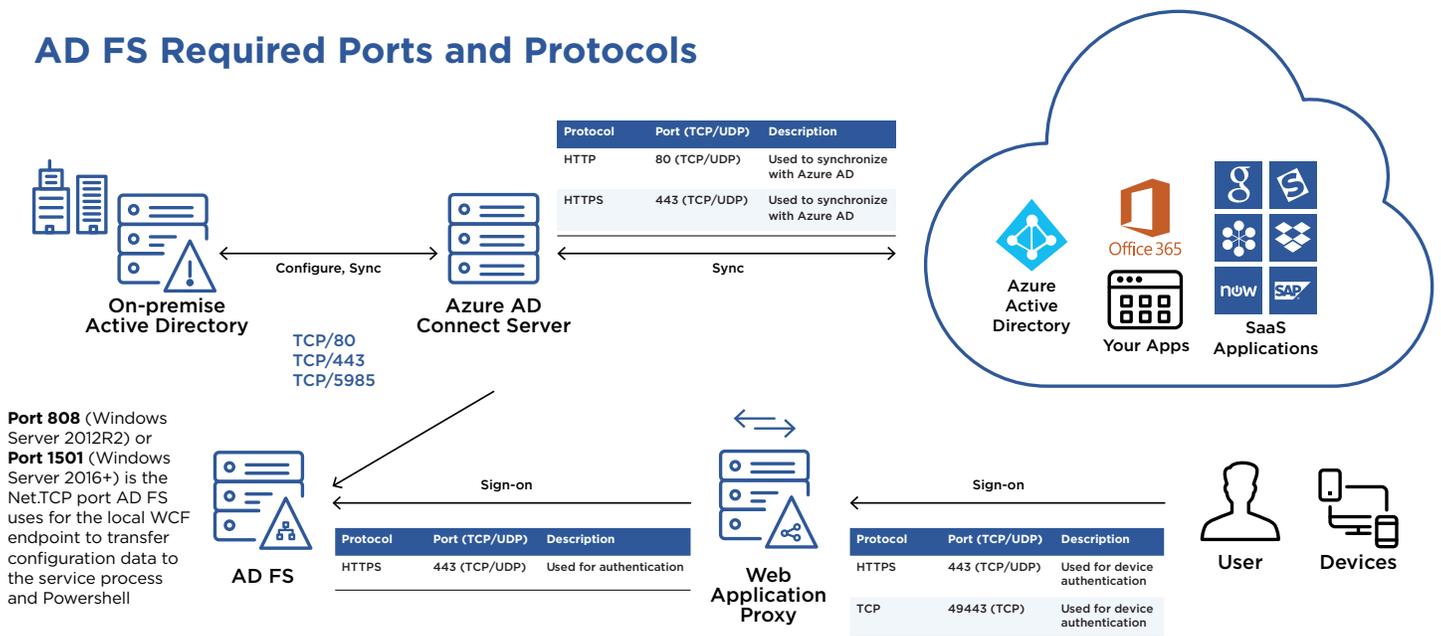
Step 3: AD FS Servers - Account Access Restrictions and Inbound Connectivity Hardening

On-premises AD FS servers should be considered Tier 0 assets. Mandiant recommends restricting access to the AD FS servers to an even smaller subset of unique accounts than the typical “Domain Administrators” group. Furthermore, Mandiant recommends that the accounts in this subset are explicitly prevented (via Group Policy) from authenticating to Tier 1 and Tier 2 assets, to prevent their credentials from being potentially recoverable on less secure systems. Particular attention should also be paid to the local administrator account on the AD FS system, ensuring that its password is strong and unique.

In addition to authentication considerations, inbound access to AD FS servers should be restricted to the following small subset of ports / protocols required for inbound connectivity (Figure 22).

- TCP/80 (HTTP)
- TCP/443 (HTTPs)
- TCP/5985 (WinRM)
- TCP/808 (Windows Server 2012R2) or TCP/1501 (Windows Server 2016+)

Figure 22. <https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/best-practices-securing-ad-fs>



Common ports leveraged for lateral movement can be restricted (inbound) for AD FS servers using a Windows Firewall configuration. Administrative connectivity and access to AD FS and AD Connect servers should only be initiated from dedicated Tier 0 Privileged Access Workstations (PAWs) or hardened jump hosts and can be defined as exceptions within Windows Firewall rulesets. Ports where inbound connectivity on AD FS servers should be restricted using Windows Firewall rules include:

- Predefined Windows Firewall Rule: File and Print Sharing
 - TCP/445, TCP/135 - TCP/139 (SMB)
- Predefined Windows Firewall Rule: Remote Desktop
 - TCP/3389 (RDP)
- Predefined Windows Firewall Rule: Windows Management Instrumentation (WMI)
 - Multiple ports

For specific Windows Firewall rule configurations that can be leveraged to restrict inbound access to these ports, reference:

<https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/wp-ransomware-protection-and-containment-strategies.pdf>.

Detection

Detection of forged SAML tokens actively being used against an organization has proven to be difficult. One possibility is to compare entries in the Azure AD Sign-Ins log against the security event logs of the on-premises AD FS servers to ensure that all authentications originated from AD FS. Technically, every sign-in recorded in Azure

AD will have a corresponding event in the on-premises security event logs. However, in real-world environments, this exercise is impractical for most organizations due to a variety of reasons:

- Sign-ins in Azure AD may not be recorded for up to 24 hours after the event occurs
- One sign-in may generate multiple sign-in records in Azure AD
- One sign-in may generate multiple security event log events on the AD FS server
- Time skew between multiple systems
- Event log retention

Other resources have also suggested looking for anomalous attributes in the SAML tokens themselves. Mandiant has observed UNC2452 using non-standard claims in the forged SAML tokens, as well as extending the lifetime of the forged SAML tokens beyond the standard one hour. These attributes are not visible to customers in the Azure AD sign-in logs to be used for detection.

To detect the use of forged SAML tokens, organization should instead focus on traditional logon analysis techniques. [Microsoft's Cloud Application Security](#) and Azure AD portal's "Azure AD risky sign-in" reports can be useful to identify sign-in events from anonymous VPN providers and infrequently used devices or locations. Impossible travel events can also help identify suspicious logons.

Organizations with Sysmon or other process and command auditing tools installed on AD FS servers may be able to identify suspicious connections to the AD FS WID from unexpected processes.

Microsoft 365 Attack Mitigation



ATTACK TECHNIQUE:

Modify Trusted Domains

Azure AD Federated Domains Overview

Similar to on-premises Active Directory, Azure AD uses the concept of “domains” to manage directories of identities. By default, an Azure AD instance will represent a single domain that is commonly called the “cloud-only” or “onmicrosoft” domain. The domain’s name will be in the form of **<domain name>.onmicrosoft.com**.

For a user’s email address to match the organization’s domain name (such as fireeye.com), an organization must add their registered domain to Azure AD and prove ownership. If the organization intends to use a third-party identity provider such as AD FS for authentication, the domain must then be configured as federated and a **federated realm object** will be created. The federated realm object tracks information such as the URL for the federated login page, the URL for the federation metadata, and the public portion of the token-signing certificate that Azure AD will use to validate digitally signed SAML tokens it receives. Any verified domain can be configured as a federated domain. **Importantly, a SAML token issued by a federated domain can be used to authenticate users that belong to any of the configured domains in the Microsoft 365 tenant.**

Technique

The threat actor must first compromise an account with permission to modify or create federated realm objects. Accounts with the “Global Administrator” or “Hybrid Identity Administrator” role can perform this action. Global Administrators, in particular, can also add new domains and configure them for federation. In an earlier investigation of suspected UNC2452 activity, Mandiant observed connections to a Microsoft 365 tenant with [MSOnline PowerShell](#) followed by the configuration of a new, attacker-controlled domain as federated.

A threat actor could also modify the federation settings for an existing domain by configuring a new, secondary, token-signing certificate. This would allow for an attack (similar to Golden SAML) where the threat actor controls a private key that is able to digitally sign SAML tokens and is trusted by Azure AD.

Both of these techniques would allow the threat actor to authenticate to Microsoft 365 as any user—bypassing the requirement to leverage a valid password for the account and perform MFA.

A more detailed discussion of this attacker technique can be found in the FireEye blog post, “[Detecting Microsoft 365 and Azure Active Directory Backdoors](#).”

Detection

The Azure AD Audit log and Unified Audit log records when a domain is configured for federated authentication and the modification of federated realm objects. In most organizations, domain federation settings will be updated infrequently. Organizations should create rules to alert on the log events generated by these activities and audit them to ensure they are legitimate.

Figure 23 shows a sample event from the Unified Audit Log that is recorded when a new domain is federated resulting in the creation of a federated realm object. This event records the user that performed the action and the source IP address, in addition to important details about the federated realm object itself. Outside of traditional analysis techniques such as user activity and IP address analysis, organizations should pay attention to the **IssuerUri** value. This value records the address of the IdP that is configured to issue SAML tokens. The address should match an expected value, such as the organization’s AD FS server.

Figure 23.

Event recorded when a new domain is federated, and a federated realm object is created.

```
{
  "OrganizationId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "CreationTime": "2020-06-23T23:03:32.0000000Z",
  "RecordType": 8,
  "Operation": "Set domain authentication.",
  "UserType": 0,
  "Workload": "AzureActiveDirectory",
  "ClientIP": "1.1.1.1",
  "Version": 1,
  "UserId": "doug@victim.onmicrosoft.com",
  "ObjectId": "Not Available",
  "ResultStatus": "Success",
  "ModifiedProperties": [
    {
      "Name": "IssuerUri",
      "NewValue": "[\r\n \"]http://any.sts/16B45E3B\"[\r\n]",
      "OldValue": "[]"
    },
    {
      "Name": "Included Updated Properties",
      "NewValue": "IssuerUri,LiveType",
      "OldValue": ""
    },
    {
      "Name": "LiveType",
      "NewValue": "[\r\n \"]Federated\"[\r\n]",
      "OldValue": "[\r\n \"]Managed\"[\r\n]"
    }
  ],
}
```

Figure 24 shows a sample event recorded in the Unified Audit Log when an existing federated realm object for a domain is modified. In most organizations, domain federation settings will be updated infrequently. Organizations should create rules to alert on the log events that get generated by these activities and audit them to ensure they are legitimate. The event records

the user that performed the action, the IP address the action was performed from, as well as the domain that was modified. It does not record what elements of the federated realm object were modified. Additional analysis using MSOnline PowerShell (detailed in the hardening and remediation sections) will be needed to identify whether the signing-certificate was modified.

Figure 24.

Event recorded when an existing federated realm object is modified.

```
{
  "OrganizationId": " xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx ",
  "CreationTime": "2020-12-04T18:41:45.0000000Z",
  "RecordType": 8,
  "Operation": "Set federation settings on domain.",
  "UserType": 0,
  "Workload": "AzureActiveDirectory",
  "ClientIP": "1.1.1.1",
  "Version": 1,
  "UserId": "victim@test.onmicrosoft.com",
  "ResultStatus": "Success",
  "Id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx",
  "ObjectId": "Not Available",
  "ModifiedProperties": [],
  "AzureActiveDirectoryEventType": 1,
  "ExtendedProperties": [
    {
      "Value": "{}",
      "Name": "additionalDetails"
    },
    {
      "Value": "Domain",
      "Name": "extendedAuditEventCategory"
    }
  ],
  "SupportTicketId": "",
  "ActorIpAddress": "1.1.1.1",
  "Target": [
    {
      "Type": 1,
      "ID": "test.com"
    }
  ],
  "
}
```

Remediation

The [Azure AD Investigator auditing script, available from the FireEye GitHub repository](#), contains a module that will enumerate all domains in a Microsoft 365 tenant. The script will output all unverified domains, domains that are configured for federation, as well as domains with a federated realm object that is evidence of the [AADBackdoor](#) tool. Administrators should review this output to ensure that only expected domains are configured as federated and that the federation login pages are expected as well.

Step 1: Review Configured Domains and Trusts Within Microsoft 365 and Remove Untrusted Domains

1. Enumerate all Registered Domains

Organizations should proactively review the scope of configured domains within a Microsoft 365 tenant, including the configured authentication methods (Managed or Federated). The PowerShell cmdlets noted in Figure 25 will provide a listing of configured domains and the associated authentication methods.

2. Remove unrecognized or suspicious domains

Organizations should assess each domain's verification status and authentication mode. These settings should be reviewed for legitimacy and functional operation. If a domain is no longer in use, or the domain's existence cannot be readily identified, it should be investigated and removed from the tenant (Figure 26).

Figure 25.

PowerShell cmdlets to review domain configurations.

```
Connect-MsolService
Get-MsolDomain

Connect-ExchangeOnline
Get-FederatedOrganizationIdentifier -IncludeExtendedDomainInfo
Get-FederationTrust
```

Figure 26.

PowerShell commands to remove an Azure AD or federated domain.

```
Connect-MsolService

** will delete a domain from Azure Active Directory. **
Remove-MsolDomain

** will remove a domain and the associated relying party trust
settings in AD FS. **
Remove-MsolFederatedDomain
```

ATTACK TECHNIQUE:

Abuse Azure AD Privileged Roles

Azure Active Directory Privileged Roles Overview

Azure Active Directory uses the concept of roles to confer administrative privileges. Roles can be assigned to users, groups, applications, and service principals (covered later in this white paper). There are over 50 roles available in Azure AD and some are more sensitive than others.

Table 2 details some of the highest privileged directory roles that threat actors often target – and should be closely monitored by organizations. This should not be treated as an exhaustive list of privileged roles. For a full listing see <https://docs.microsoft.com/en-us/azure/active-directory/roles/permissions-reference>.

Table 2. Azure AD - Highly Privileged Roles.

Role	Description	Impact
Application administrator	Can create and manage app registrations and enterprise apps	Backdoor existing applications
Cloud application administrator	Can create and manage app registrations and enterprise apps	Backdoor existing applications
Exchange administrator	Manage all aspects of exchange	Grant access to user mailboxes Change mail settings
Global administrator	Manage all aspects of Azure AD and Microsoft services that use it	Complete control over the Microsoft 365 environment
Privileged role administrator	Manage role assignments	Privilege escalation via privileged roles
User access administrator	Assign roles in all subscriptions and management groups	Privilege escalation via privileged roles
SharePoint administrator	Can manage all aspects of SharePoint	Complete control over SharePoint including access to sites and documents
Hybrid identity administrator	Can update federation settings for domains	Ability to install an azure AD backdoor

Technique

Organizations will often grant privileged roles to user identities that are synced from on-premises Active Directory. Most commonly, this includes a privileged on-premises account being used to manage Azure AD and Microsoft 365 resources. This can present a security risk as it facilitates lateral movement from the on-premises environment to the cloud environment. Mandiant has observed threat actors performing reconnaissance to identify on-premises AD accounts that hold privileged roles and target them for credential theft. Once in possession of credentials for an account that holds a privileged directory role, the threat actor uses it to connect to Microsoft 365 and establish a foothold for directly accessing cloud resources without the continued need for on-premises access.

Remediation**Step 1: Review and Configure Cloud-Only Accounts for Privileged Role Holders**

Organizations should review the scope of accounts assigned privileged permissions and roles in Microsoft 365. Additionally, the roles and accounts with privileged access should be cloud-only accounts (not accounts synced from on-premises identity stores such as Active Directory). All privileged cloud-only accounts should leverage MFA through either conditional access policies or security defaults.

1. Review Accounts assigned Privileged Roles

Using the PowerShell commands referenced in Figure 27, review each account that is assigned to a privileged role in Azure AD. Any identified on-premises synced accounts that are configured with a privileged role in Azure AD should be replaced with a cloud-only account. Cloud-only accounts are denoted by the **@<tenant>.onmicrosoft.com** domain suffix.

Figure 27.

PowerShell commands to review privileged role assignments in Azure AD.

```
Connect-AzureAD
$Role = Get-AzureADDirectoryRole
$Role | ForEach-Object {
    $ObjectId = $_.ObjectId
    $displayName = $_.displayName
    $a = Get-AzureADDirectoryRoleMember -objectid $ObjectId | select-Object
    ObjectID, UserPrincipalName
    $a | Add-Member -NotePropertyName GroupID -NotePropertyValue $ObjectId
    $a | Add-Member -NotePropertyName GroupName -NotePropertyValue
    $DisplayName
    $a | Export-csv Get-AzureADDirectoryRoleMember.csv -Append
    -NoTypeInfoInformation
}
```

2. List and review ImmutableID values for cloud-only accounts

ImmutableIDs act as an anchor attribute between synced on-premises identities and their cloud account counterpart. If a cloud-only account has an ImmutableID, then a threat actor could forge a SAML token for the account, even though it is considered cloud-only. For this reason, all cloud-only accounts should not be configured with an ImmutableID.

The PowerShell command noted in Figure 28 can be leveraged to review cloud-only accounts with an ImmutableID.

3. For accounts that contain an ImmutableID value, delete and recreate the account

Figure 28.

PowerShell command to identify cloud-only accounts with an ImmutableID.

```
Connect-MsolService
Get-MsolUser -All | where-Object {$_.
UserPrincipalName -like "*.onmicrosoft.com"} | select
DisplayName,UserPrincipalName,ImmutableID
```

Step 2: Rotate All Passwords for Cloud-Only Accounts

Once the final set of privileged accounts have been determined, force password changes for all cloud-only accounts that have not had their passwords recently rotated. The PowerShell command noted in Figure 29 will list all enabled cloud-only accounts and the corresponding last password change value.

Figure 29.

PowerShell command to determine the last password change date for accounts.

```
Connect-MsolService
Get-MsolUser -All | where-Object {($_.UserPrincipalName -like
"*.onmicrosoft.com") -and ($_.BlockCredential -eq $false)} | select
DisplayName,UserPrincipalName,LastPasswordChangeTimeStamp
```

Step 3: Invalidate Refresh Tokens for Each Cloud-Only Account

After rotating credentials for existing accounts, organizations should immediately revoke existing refresh tokens for each account. This will result in users being required to reauthenticate. Using an account that has Microsoft 365 Global Administrator permissions, invalidate each existing account's refresh token using the PowerShell commands noted in Figure 30.

Figure 30.

PowerShell commands to connect to Azure AD and revoke existing refresh tokens.

```
Connect-AzureAD
Get-AzureADUser -Identity <UPN> | Revoke-AzureADUserAllRefreshToken
```

ATTACK TECHNIQUE:

Hijack Azure AD Applications

Azure Applications and Service Principals Overview

Applications are created in Azure AD to provide API access to Microsoft 365 data by third parties. Different terms are used to refer to applications depending on how they are created and how they are interacted with. This section defines these terms before a discussion of the threat actor techniques and how to protect against them.

- **App Registrations**, also known as **Applications or Application Objects**, define the initial instance of an application, including API permissions, application secrets, and branding elements. The app registration resides in the tenant where it was created. For example, the Exchange Online application resides in the Microsoft Corporation tenant and the custom application created by the IT team resides in the organization's tenant. Application objects serve as a "blueprint" to create a service principal in any tenant that uses the application.
- **Enterprise Applications**, also known as **Service Principals or Application Instances**, represent applications that are being used within a Microsoft 365 tenant. These can be third-party applications such as a Salesforce integration, or custom applications built by the organization. There are also "first-party" service principals used for components of Microsoft 365 itself. Examples include Exchange Online, Microsoft Graph, and the Azure Portal. Every tenant using an application will automatically create a service principal for it. There is a one-to-many relationship between applications and service principals.

The term "applications" is often used to refer to both App Registrations and Enterprise Applications as many concepts apply to both. This white paper will follow the same practice going forward and refer to specific terms when needed.

There are two different models when assigning permissions for applications to access an API:

- **Delegated Permissions**, also known as **OAuth2PermissionGrants**, enable an application to perform API operations that access or manipulate data on behalf of a signed-in user. A user must sign-in and consent to, or allow, the application to access data on their behalf. An administrator can also consent on the behalf of all users. Delegated permissions are out of scope for this white paper.
- **Application Permissions**, also known as **AppRoles and AppRoleAssignments**, enable an application to perform API operations that access or manipulate data without a signed-in user. Application Permissions allows the application to interact with an API as itself, without a session that is acting on behalf of a specific user.

Both Applications and Service Principals can have delegated or application permissions associated with them.

In order to authenticate to Microsoft 365 and access the APIs, Applications and service principals can also have credentials in the form of **secrets** or **certificates** associated with them. These are roughly analogous to API keys in other cloud services. Administrators can add a credential that is then used to connect to Microsoft 365 with the identity of the application or service principal. Once the credential has been added to Microsoft 365, its value cannot be extracted. An attacker that is able to add a new secret or certificate can connect to Azure AD as the application and perform API operations using whatever Application Permissions that are assigned to it. For example, an organization may configure a backup application with the mail.read Application Permission to allow it to read all mail in the tenant and create backups. **Hijacking an application (by adding a rogue secret or certificate) with granted permissions will allow the attacker to access data that is normally protected by MFA requirements.**

Technique

Mandiant has observed UNC2452 adding additional certificates or secrets to existing applications and service principals. To accomplish this, the threat actor first needed to compromise an account with a privileged directory role that was granted the ability to modify Enterprise Applications or Application Registrations. A frequently targeted role was the Application Administrator, because it allowed complete control over applications in Azure AD. Once in possession of those credentials, UNC2452 modified the application by using one of the following techniques:

- **Modification of an application via the browser:**
The attacker connected to the Azure Portal with a web browser and added a new secret to an App Registration that had permissions to read all mail. There is no supported way to add a credential to an Enterprise Application (service principal) via the browser.
- **Modification of an application via PowerShell:**
The attacker connected to the Microsoft 365 tenant using Azure AD PowerShell. Once connected they added certificates to App Registrations and Enterprise Applications that had the **mail.read** and **files.read** permission.

Mandiant has observed UNC2452 first performing reconnaissance of an organization's Azure AD to identify existing applications that already have sensitive permissions set, such as **mail.read** and **files.read**. Once a viable application was identified, UNC2452 added a rogue credential to it using one of the methods detailed above.

At a minimum this allowed UNC2452 to access any user's data in the victim tenant without MFA.

An important note about hijacking an App Registration:

App Registrations are the original copy of an application that resides in the creator's tenant. Any other tenant that uses this application creates a copy of it called a service principal. **If a threat actor were to hijack the App Registration of an application that is used by other organizations, the threat actor would be able to access the data of those organizations as well.**

Detection

The Azure AD Audit log and the Unified Audit Log record when credentials are added to both service principals and App Registrations. In most organizations, credentials will be added to service principals and App Registrations infrequently. Organizations should create rules to alert on the log events that get generated by these activities and audit them to ensure they are legitimate. For example, [Microsoft has published suggested hunting rules](#) using Sentinel to identify these activities.

Figure 31 shows a sample event from the Unified Audit Log that is recorded when a credential is added to an App Registration. The event includes the user that performed the action, the ID of the application that was modified, details on the credential that was added, and more. Of note are the **ClientIP** and **ActorIPAddress** fields. When the credential is added via the Azure Portal, the IP address that gets recorded is a Microsoft IP address. This hides the true IP address of the action and can make searching for activities by IP address prone to missing data.

Figure 31.

Sample event recorded when a credential is added to an App Registration.

```
{
  "OrganizationId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "CreationTime": "2021-01-09T18:40:16.0000000Z",
  "RecordType": 8,
  "Operation": "Update application - Certificates and secrets management ",
  "Workload": "AzureActiveDirectory",
  "UserType": 0,
  "ClientIP": "20.190.130.49",
  "Version": 1,
  "UserId": "doug@testing.onmicrosoft.com",
  "ObjectId": "Application_xxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "Id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "ResultStatus": "Success",
  "ModifiedProperties": [
    {
      "Name": "KeyDescription",
      "NewValue": "[\r\n \"][KeyIdentifier= xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx
xxx,KeyType=Password,KeyUsage=Verify,DisplayName=test]\r\n]",
      "OldValue": "[]"
    },
    {
      "Name": "Included Updated Properties",
      "NewValue": "KeyDescription",
      "OldValue": ""
    }
  ],
  "AzureActiveDirectoryEventType": 1,
  "ExtendedProperties": [
    {
      "Value": "{\\"User-Agent\\":\\"Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88
Safari/537.36\\"}",
      "Name": "additionalDetails"
    }
  ]
}
```

A similar event is recorded when credentials are added to a service principal. Figure 32 details a sample log event from the Unified Audit Log.

When a credential that has been added to an application is used to login to Microsoft 365, it is recorded differently than an interactive user sign-in. In the Azure Portal these logins can be viewed by navigating to Sign-Ins under the Azure Active Directory blade and then clicking the service principal Sign-ins tab. The sign-in log records the IP address that performed the sign-in, but not the credential that was used. **Note that currently these sign-ins are not recorded in the Unified Audit Log.**

Figure 32.

Sample event recorded when a credential is added to a service principal.

```
{
  "OrganizationId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "CreationTime": "2021-01-07T21:32:13.0000000Z",
  "RecordType": 8,
  "Operation": "Add service principal credentials.",
  "UserType": 0,
  "Workload": "AzureActiveDirectory",
  "ClientIP": "1.1.1.1",
  "Version": 1,
  "UserId": "doug@test.onmicrosoft.com",
  "ObjectId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "Id": " xxxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ",
  "ResultStatus": "Success",
  "ModifiedProperties": [
    {
      "Name": "KeyDescription",
      "NewValue": "[\r\n \"][KeyIdentifier=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
xx,KeyType=Symmetric,KeyUsage=Verify,DisplayName=]\r\n]",
      "OldValue": "[]"
    },
  ],
}
```

Remediation

The [Azure AD Investigator auditing script, available from the FireEye GitHub repository](#), contains a module that will enumerate all applications and service principals in a Microsoft 365 tenant. The script will output all applications and service principals that have both certain high-risk application permissions assigned to them and an assigned credential. These are applications that a threat actor may have used to connect to Microsoft 365 and access data. Administrators will need to work with their security and IT teams to validate that the credentials added to the applications and service principals are legitimate.

Step 1: Review and Remediate Keys Associated with Service Principals and Applications

Organizations can leverage Azure PowerShell to review and remediate service principals or applications that may be configured with credentials for remote authentication using either a static secret or certificate.

1. Search for suspicious service principal credentials
To search for evidence of credentials that have been configured or added to a specific service principal, the PowerShell command noted in Figure 33 can be leveraged. The command output will include the KeyId, StartDate, EndDate, and key type (password / certificate).
2. Remove suspicious service principal credentials
To remove a suspicious credential associated with a specific service principal, the PowerShell commands noted in Figure 34 can be leveraged.
3. Search for suspicious Application credentials
To search for evidence of credentials that have been configured or added to a specific Azure application, the PowerShell command noted in Figure 35 can be leveraged.
4. Remove suspicious Application credentials
To remove a suspicious credential associated with a specific application, the PowerShell commands noted in Figure 36 can be leveraged.

Figure 33.

PowerShell command to review a service principal – and any associated credentials that may be configured.

```
Connect-AzAccount
Get-AzADServicePrincipal -ApplicationID <> | Get-AzADSPCredential
```

Figure 34.

PowerShell commands to remove credentials associated with a service principal – or remove a specific service principal.

```
Connect-AzAccount
** Remove all credentials associated with a service principal **
Get-AzADServicePrincipal -ObjectId <> | Remove-AzADSpCredential
** Remove a specific credential (KeyID) associated with a service
principal **
Remove-AzADSpCredential -ObjectId <> -KeyId <>
** Remove a specific service principal **
Remove-AzADServicePrincipal -ObjectId <>
```

Figure 35.

PowerShell command to review a service principal – and any associated credentials that may be configured.

```
Connect-AzAccount
Get-AzADApplication -ApplicationID * | Get-AzADAppCredential
```

Figure 36.

PowerShell commands to remove credentials associated with an application – or remove a specific application.

```
Connect-AzAccount
** Remove all credentials associated with a service principal **
Get-AzADApplication -ObjectId <> | Remove-AzADAppCredential
** Remove a specific credential (KeyID) associated with a service
principal **
Remove-AzADAppCredential -ObjectId <> -KeyId <>
** Remove a specific application **
Remove-AzADApplication -ObjectId <>
```

Step 2: Invalidate All Existing Refresh Tokens

After removing malicious credentials, organizations should immediately revoke all existing refresh tokens for their Microsoft 365 tenant. This will result in users being required to reauthenticate. Using an account that has Microsoft 365 Global Administrator permissions, invalidate all existing refresh tokens using the PowerShell commands noted in Figure 37.

Figure 37.

PowerShell commands to connect to Azure AD and revoke existing refresh tokens.

```
Connect-AzureAD
Get-AzureADuser -all $true | Revoke-AzureADUserAllRefreshToken
```

ATTACK TECHNIQUE: Modify Mailbox Folder Permissions

Mailbox Folder Permission Overview

Within a Microsoft Exchange mailbox, you can grant granular access permissions to individual folders. These permissions can be granted directly by the mailbox owner, or on behalf of a mailbox owner by either an account that has been delegated access to the mailbox or an Exchange administrator. Mailbox folder permissions are frequently used to share a specific folder within a mailbox with a co-worker instead of delegating rights for the entire mailbox. Administrators will often make a calendar public by granting read access to it for all authenticated users.

Mailbox folder permissions can be assigned through either individual folder permissions or roles, which are a collection of folder permissions. The **ReadItems** permission grants read access to the items in a mailbox folder. The following roles include permissions that grant read access to folder items:

- Author
- NonEditingAuthor
- PublishingEditor
- Reviewer
- Editor
- Owner
- PublishingAuthor

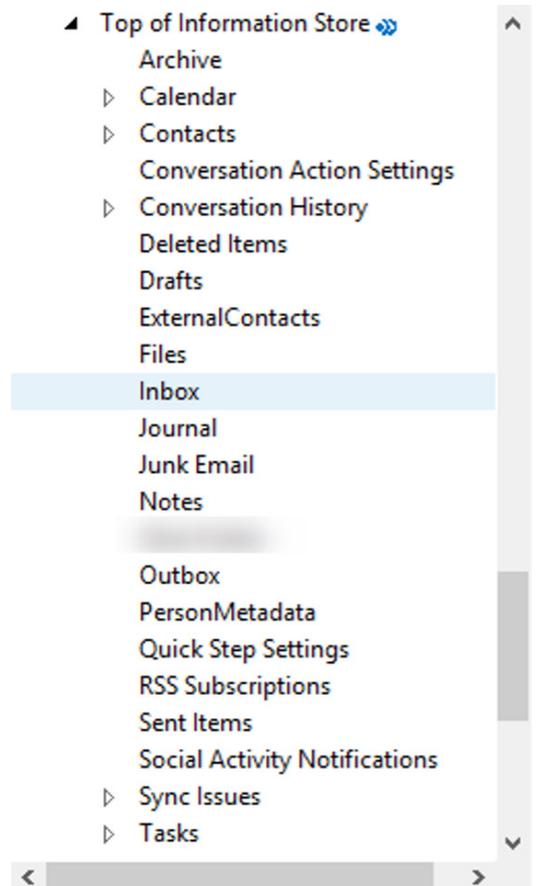
For a full listing of mailbox folder permissions and roles see <https://docs.microsoft.com/en-us/powershell/module/exchange/set-mailboxfolderpermission?view=exchange-ps#parameters>. Folder permissions can be assigned to users and mail-enabled security groups. In addition to the individual users in an Microsoft 365 tenant, there are two special users:

- Anonymous: External, unauthenticated users
- Default: Internal, authenticated users

By default, the access right of None is assigned to both the Default and Anonymous user on each mailbox in the tenant. For each mailbox, the user-accessible folders (such as Inbox, Sent Items, Deleted Items) reside within a hidden root folder called the “Top of Information Store”. Just like user-visible folders, folder permissions can be granted to it. By default, the Default and Anonymous user are assigned the **None** permission (Figure 38).

Figure 38.

Top of Information Store Hierarchy.



Technique

Mandiant has observed threat actors assigning the **Default** user permissions or roles that grant read privileges to items stored in the Inbox and other mailbox folders. By assigning read privileges to the **Default** user, an attacker could then login as any user in the Microsoft 365 tenant and read the items stored in the folder. First, the threat actor used an administrator account to modify mailbox folder permissions for mailboxes of interest in the tenant. Then, the threat actor used a low-privileged user to login on a daily basis and download messages from the mailboxes folders that they previously modified. These logins were only recorded in Azure AD sign-ins because they were coded as non-interactive sign-ins.

Mailbox folder permissions can be added or modified using either the Outlook client or PowerShell using the **Add-MailboxFolderPermission** or **Set-MailboxFolderPermission** cmdlets.

Remediation

If an organization has evidence that a sophisticated threat actor has accessed their Microsoft 365 tenant, they should consider auditing mailbox folder permissions for any unauthorized modifications. The Mandiant Azure AD Investigator will enumerate permissions for the Top of Information Store and Inbox folders for all mailboxes within a tenant and output mailboxes where access rights for the Default or Anonymous users are modified from the default setting of None. Based on the output data from the script, organizations will need to determine whether or not the modified mailbox folder permissions are legitimate. Given the overly permissive nature of the Default and Anonymous users, access rights should be revoked and the principle of least privilege should be applied. Organizations should only grant permissions to an explicit list of users based on business need.

Review and Remove Folder Access Rights Granted for Default or Anonymous Permissions

1. Enumerate Mailbox Folder Permissions (quick)

The Mandiant Azure AD Investigator tool includes a cmdlet to perform this task.

2. Enumerate all Folder Permissions for a Specific Mailbox (thorough)

Given the number of folders that exist in each mailbox, enumerating every folder permission for every mailbox is inefficient. Mandiant recommends a triage approach to first identify mailboxes of interest by scanning the permissions of specific well-known folders such as Inbox and then performing a full review on any of the identified mailboxes. The script in Figure 39 will enumerate all assigned permissions for all folders for a given mailbox.

3. Remove untrusted or suspicious permissions

To remove any excessive permissions for a folder, the PowerShell command noted in Figure 40 can be used.

Figure 39.

Permission enumeration script.

```
#Check if current session is connected to EXO, else prompt for login
$EXOCheck = Get-PSSession | where-Object ComputerName -eq 'outlook.office365.com'
If ($EXOCheck){
    Write-Host "Already connected to: $($Get-AcceptedDomain | where-Object Default
-eq $true).Name)" -ForegroundColor Yellow
    $EXOContinue = Read-Host -Prompt "Continue scanning this tenant (Y/n)"
    If ($EXOContinue -match '^y$|^yes$|^$'){
        Write-Host 'Continuing against connected tenant.' -ForegroundColor Green
    } Else {
        Write-Host 'Script aborted due to user input, please re-run.'
    }
} Else {
    Connect-ExchangeOnline
}
#Receive user input for UPN
$User = Read-Host -Prompt "Enter UPN of Mailbox to Review Folder Permissions"

#Store output file to variable
$OutputFileName = "Get-UserMailboxFolderPermission.csv"

#Store all mailboxes to variable
Try{
    Write-Host -Object "Retrieving Mailboxes, Please wait..."
    $mbox = Get-EXOMailbox -Identity $User
}Catch{
    Write-Warning -Message "Problem obtaining a list of mailboxes, check
connection or permissions to Connect-ExchangeOnline"
    Write-Warning -Message $_
    break
}

Write-Host -Object "Enumerating Mailbox Folder Permissions. Please wait...."
#Retrieve all permissions for all folders in a specific mailbox
Try{
    $Folders = (Get-MailboxFolderStatistics $mbox.UserPrincipalName).FolderPath
    foreach($folder in $Folders){
        $identity = "$($mbox.UserPrincipalName):$folder"
        Get-MailboxFolderPermission -Identity $identity.Replace("/", "\")
    }
}Catch{
    Write-Warning -Message "Problem accessing Mailbox Folders permissions"
    Write-Warning -Message $_
    break
}

#output results. Display file location
Write-Host -Object "Completed the export of folder permissions. Results have been
saved to $(Join-Path -Path (Get-Location).Path -ChildPath $OutputFileName)"
```

Figure 40.

PowerShell command to remove mailbox folder permissions for a specific account.

```
Connect-ExchangeOnline
Set-MailboxFolderPermission -Identity <UPN>:\<Top of Information Store, Inbox, or
Desired Path> -User <Anonymous or Default> -AccessRights None
```

Detection

To proactively monitor for malicious folder permission modifications, specific mailbox auditing must be enabled. For new Microsoft 365 tenants, mailbox auditing is enabled by default. For pre-existing tenants it must be enabled per mailbox. Each mailbox must have the **UpdateFolderPermissions** action enabled for all logon types (owner, delegate and admin). This auditing configuration is part of the default audit set when an organization enables mailbox auditing. To review mailbox audit settings and enable **UpdateFolderPermissions** auditing, refer to the [Step 3: Enhance Mailbox Auditing](#).

The mailbox audit log will forward folder permission modification events to the Unified Audit Log. Organizations should create rules to alert on **ModifyFolderPermission** operations where the Anonymous or Default user has been assigned permissions other than **None**.

Figure 41 shows a sample event from the Unified Audit Log where the **Default** user has been assigned the **Reviewer** role for an Inbox folder. Note that the terms used in the log record differ from the terms used to assign or view the permissions in Outlook. In the audit log the **Everyone** value for **MemberUpn** is the same as default user, and the collection of **MemberRights** maps to the **Reviewer** role.

Figure 41.

Modified permission of folder for Default user.

```
{
  "CreationTime": "2021-01-13T14:39:18",
  "Id": "xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "Operation": "ModifyFolderPermissions",
  "OrganizationId": "xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "RecordType": 2,
  "ResultStatus": "Succeeded",
  "UserKey": "XXXXXXXXXX",
  "UserType": 0,
  "Version": 1,
  "Workload": "Exchange",
  "ClientIP": "1.1.1.1",
  "UserId": "vic@acme.net",
  "AppId": "00000002-0000-0ff1-ce00-000000000000",
  "ClientAppId": "00000002-0000-0ff1-ce00-000000000000",
  "ClientIPAddress": "1.1.1.1",
  "ClientInfoString": "Client=WebServices;ExchangeWebServicesProxy/CrossSite/EXCH/15.20.3742.012/Microsoft Office/16.0 (Windows NT 10.0; Microsoft Outlook 16.0.12430; Pro)[AppId=00000002-0000-0ff1-ce00-000000000000];",
  "ExternalAccess": false,
  "InternalLogonType": 0,
  "LogonType": 0,
  "LogonUserSid": "S-1-5-21-xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx ",
  "MailboxGuid": "xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx ",
  "MailboxOwnerSid": "S-1-5-21-xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx ",
  "MailboxOwnerUPN": "vic@acme.net",
  "OrganizationName": "acme.onmicrosoft.com",
  "OriginatingServer": "MN2PR07MB7040 (15.20.3742.012)\r\n",
  "Item": {
    "Id": "xxxxxxxxxxxxxxxx",
    "ParentFolder": {
      "Id": "xxxxxxxxxxxxxxxx",
      "MemberRights": "ReadAny, Visible, FreeBusySimple, FreeBusyDetailed",
      "Membersid": "S-1-1-0",
      "MemberUpn": "Everyone",
      "Name": "Inbox",
      "Path": "\\Inbox"
    }
  }
}
```

Microsoft 365 Hardening

This section contains hardening recommendations for Microsoft 365 that are associated with the attacker techniques described in this white paper. A comprehensive discussion on Microsoft 365 hardening is outside the scope of this white paper.

Step 1: Filter accounts synced to Azure Active Directory

On-premises Active Directory users that are synced to Azure AD should follow the concept of least privilege. Only identities that utilize cloud services should be synced using AD Connect, and any accounts that are deemed to have a level of on-premises permissions should not be synced and assigned permissions in Azure AD. This can be accomplished using AD Connect sync filtering, where four options are available:

- Filter by Domain
- Filter by Security Group
- Filter by OU
- Filter by Attribute

1. Review Azure AD Sign-In logs

To identify on-premises Active Directory users that do not utilize Azure AD, Azure AD Sign-in logs can be reviewed (Figure 42 - Figure 44).

Review user Azure AD Sign-ins by user UPN:

Figure 42.

PowerShell command to review Azure AD sign-in logs to identify potentially dormant accounts by UPN.

```
Connect-AzureAD
Get-AzureADAuditSignInLogs -Filter "UserPrincipalName eq <UPN>" | Select-Object UserPrincipalName, CreatedDateTime, AppDisplayName, AppId, IPAddress | Export-csv signins.csv -Append
```

Review user Azure AD Sign-ins by Active Directory OU:

Figure 43.

PowerShell commands to review Azure AD sign-in logs to identify potentially dormant accounts by AD OU.

```
#Export members of AD OU to CSV
Get-ADUser -Filter * -SearchBase "<OU PATH>" | Select-object UserPrincipalName | Export-csv OUMembers.csv

#Import CSV into variable
$Users = Import-csv OUMembers.csv

#Review OU member logins against Azure AD Sign-in logs
Connect-AzureAD
Foreach ($user in $Users) { $u = $user.UserPrincipalName Get-AzureADAuditSignInLogs -Filter "UserPrincipalName eq '$u'" | Select-Object UserPrincipalName, CreatedDateTime, AppDisplayName, AppId, IPAddress | Export-csv signins.csv -Append
}
```

Review user Azure AD Sign-ins by Active Directory Security Group:

Figure 44.

PowerShell commands to review Azure AD sign-in logs to identify potentially dormant accounts by security group association.

```
#Export members of AD Security Group to CSV
Get-ADGroupMember -Identity <Group Name> -Recursive | Select-object
UserPrincipalName | Export-csv GroupMembers.csv

#Import CSV into variable
$Users = Import-csv GroupMembers.csv

#Review Group member logins against Azure AD Sign-in logs
Connect-AzureAD
Foreach ($user in $users) { $u = $user.UserPrincipalName Get-
AzureADAuditSignInLogs -Filter "UserPrincipalName eq '$u'" | Select-Object
UserPrincipalName, CreatedDateTime, AppDisplayName, AppId, IPAddress |
Export-csv signins.csv -Append
}
```

2. Configure filtering with Azure AD Connect

For more information regarding Azure AD Connect filtering, reference:

<https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-sync-configure-filtering>

Step 2: Limit Privileged Users to Trusted IPs

When privileged accounts are leveraged to access and administer Microsoft 365 resources, connectivity from the accounts should only be permissible from trusted locations, usually representing public IP address blocks managed by an organization.

To Configure Trusted Networks:

1. Login to <https://portal.azure.com> > **Azure Active Directory > Security > Named Locations**
2. Click **New Location**
3. Provide a **Name** and **IP Ranges**
 - a. Check **Mark as Trusted location**
4. Select **Save**

Conditional Access Policy Configuration:

1. Under **Security > Conditional Access**
2. Select **New policy**
 - a. **Users and Groups > Include**
 - i. Select **Directory Roles**
 - Check **All Roles**
 - b. **Users and Groups > Exclude**
 - i. Add Break-Glass Global Administrators
 - Break-Glass Global Administrators should exempt from all CAPs in the event CAPs are not being applied as expected.

c. Cloud apps or admins > Include

- i. Select **All cloud apps**

d. Conditions > Locations

- i. Under **Configure**, select **Yes**
- ii. **Include > Any Location**
- iii. **Exclude > select Locations > Choose newly added location**

e. Grant > Block Access

Note: Prior to enabling the policy, to prevent an account from being locked out of Microsoft 365, it is highly recommended to test the policy in **Report-Only** mode to ensure the configuration acts as expected.

For additional information related to conditional access policies, reference

- <https://docs.microsoft.com/en-us/microsoft-365/security/office-365-security/identity-access-policies>
- <https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/plan-conditional-access>

Step 3: Enhance Mailbox Auditing

Enabling mailbox auditing for users will provide greater visibility into potentially suspicious activity. Mailbox auditing provides organizations with visibility related to logon events for mailboxes as well as specific actions that occurred based upon either the mailbox owner, delegate, or an administrator. With optimized audit logging in Microsoft 365, organizations are empowered to enhance detection, monitoring, and investigative activities.

1. Review Mailbox Auditing Settings

To verify mailbox auditing settings for configured mailboxes, the PowerShell command noted in Figure 45 can be leveraged.

Figure 45.

PowerShell command to verify mailbox auditing settings.

```
Connect-ExchangeOnline
Get-Mailbox -Resultsize Unlimited -Filter {(RecipientTypeDetails
-eq "UserMailBox") -and (SKUAssigned -eq "True")} | Select
userprincipalname, Audit* | export-csv EnableMailBoxAudtingSettings.
csv
```

2. Enable Audit Status for all Mailboxes

To enforce that mailbox audit logging is enabled for all mail accounts, the PowerShell command noted in Figure 46 can be leveraged.

Figure 46.

PowerShell command to enforce mailbox auditing.

```
Connect-ExchangeOnline
Get-Mailbox -Resultsize Unlimited -Filter {(RecipientTypeDetails
-eq "UserMailBox") -and (SKUAssigned -eq "True")} | Set-Mailbox
-AuditEnabled $True
```

3. Set Mailbox audit logging retention

At a minimum, Mandiant recommends that auditing for mailbox audit logs be retained for at least 90 days. To enforce this configuration, the PowerShell command noted in Figure 47 can be leveraged.

Figure 47.

PowerShell command to enforce a mailbox audit log retention period.

```
Connect-ExchangeOnline
Get-Mailbox -Resultsize Unlimited -Filter {(RecipientTypeDetails
-eq "UserMailBox") -and (SKUAssigned -eq "True")} | Set-Mailbox
-AuditLogAgeLimit 90
```

4. Enable Verbose Mailbox Auditing Settings

Note: Before applying verbose Mailbox Auditing settings, an organization should verify that their centralized log or SIEM platform can handle the increased logging volume.

Verbose E3 Licensing Auditing Settings

At a minimum, the **MailboxLogin** action for the **Owner** Logon Type should be added to each mailbox's audit settings.

The PowerShell command noted in Figure 48 will **add** the **MailBoxLogin** auditing setting to the **AuditOwner** logon type for each mailbox:

Figure 48.

PowerShell command to add the MailBoxLogin setting for the AuditOwner logon type.

```
Connect-ExchangeOnline
Get-Mailbox -Resultsize Unlimited -Filter {(RecipientTypeDetails
-eq "UserMailBox") -and (SKUAssigned -eq "True")} | Set-Mailbox
-AuditOwner @{Add=MailBoxLogin}
```

To enable the highest level of logging available with E3 licensing, the PowerShell command noted in Figure 49 can be run on each mailbox to **replace** auditing settings for each logon type,

Figure 49.

PowerShell command to replace auditing settings with optimized E3 settings for each logon type.

```
Connect-ExchangeOnline
Set-Mailbox -Identity <UPN> `
-AuditAdmin MoveToDeletedItems,SoftDelete,HardDelete,SendAs,SendOnBehalf,
UpdateFolderPermissions,UpdateInboxRules,UpdateCalendarDelegation `
-AuditDelegate MoveToDeletedItems,SoftDelete,HardDelete,SendAs,
SendOnBehalf,UpdateFolderPermissions,UpdateInboxRules `
-AuditOwner MoveToDeletedItems,SoftDelete,HardDelete,MailboxLogin,
UpdateFolderPermissions,UpdateInboxRules,UpdateCalendarDelegation
```

Verbose E5 Licensing Audit Settings

E5 and E5 Compliance Add-on licensing includes the audit setting **MailItemsAccessed**. **MailItemsAccessed** is a reliable logging mechanism which replaces the **MessageBind** logging mechanism. For incident responders, having access to **MailItemsAccessed** events will provide a better understanding of the scope and the depth of a potential compromise.

Mandiant recommends that **MailItemsAccessed** be enabled for all logon types for highly sensitive mailboxes. The PowerShell command noted in Figure 50 will **add** the **MailItemsAccessed** audit setting for the **AuditOwner**, **AuditDelegate**, and **AuditAdmin** logon types for a specific mailbox.

Figure 50.

PowerShell command to add the MailItemsAccessed setting for the AuditOwner, AuditDelegate, and AuditAdmin logon types.

```
Connect-ExchangeOnline
Set-Mailbox -Identity <UPN> -AuditOwner @{Add=MailItemsAccessed}
-AuditDelegate @{Add=MailItemsAccessed} -AuditAudit @
{Add=MailItemsAccessed}
```

To enable the highest level of logging available for E5 licensing, the PowerShell command noted in Figure 51 can be run on each mailbox to **replace** auditing settings for each logon type.

Figure 51.

PowerShell command to replace auditing settings with optimized E5 settings for each logon type.

```
Connect-ExchangeOnline
Set-Mailbox -Identity <UPN> `
-AuditAdmin MoveToDeletedItems,SoftDelete,HardDelete,SendAs,
SendOnBehalf,UpdateFolderPermissions,UpdateInboxRules,
UpdateCalendarDelegation,MailItemsAccessed `
-AuditDelegate MoveToDeletedItems,SoftDelete,HardDelete,SendAs,
SendOnBehalf,UpdateFolderPermissions,UpdateInboxRules,MailItemsAccessed `
-AuditOwner MoveToDeletedItems,SoftDelete,HardDelete,MailboxLogin,
UpdateFolderPermissions,UpdateInboxRules,UpdateCalendarDelegation,
MailItemsAccessed
```

Note: If Microsoft releases new auditing settings, the new settings will not be automatically applied to mailboxes where custom audit settings have previously been configured. New auditing settings will need to be manually applied.

For more information regarding mailbox auditing, reference:

<https://docs.microsoft.com/en-us/microsoft-365/compliance/enable-mailbox-auditing?view=o365-worldwide>

Step 4: Review Azure Application and Service Principal Permissions

Azure applications and service principals can be granted access to protected resources in Microsoft 365. For example, a token can be granted to allow a third-party application to read a user’s mailbox or download the contents of a user’s OneDrive account.

Once an application has been granted permissions, an attacker can utilize the application to access user accounts with little to no other security controls. This is commonly leveraged by attackers since they can bypass MFA requirements and maintain persistence to targeted employee’s data.

For this reason, it is important to have visibility over which applications have been granted OAuth access to user accounts within the tenant.

1. Run Create-AppConsentGrantReport.ps1 Script

[Create-AppConsentGrantReport.ps1](#) is an open-source PowerShell script that will enumerate all permission grants for all applications in a Microsoft 365 tenant. This output can then be reviewed to identify overly permissive and suspicious applications.

Figure 52.

PowerShell command to execute the Create-AppConsentGraphReport.ps1 script - enumerate all permission grants for all applications.

```
Connect-AzureAD
.\Create-AppConsentGrantReport.ps1 -AdminUPN
globalreader@contoso.onmicrosoft.com -Path .\output.xlsx
```

Many applications have legitimate use cases that warrant permissive access to applications (email applications and cloud-based collaboration platforms) in a Microsoft 365 tenant. The output from this script report should be thoroughly reviewed to determine which applications can be removed from the tenant or have their privileges reduced.

2. Apply Application Access Policies

For applications that are granted the ‘**Application**’ Consent Type for mail permissions (Mail.Read,Mail.Send,Mail.ReadWrite,Mail.ReadBasic,Mail.ReadBasic.All), these applications are granted the rights to **all** mailboxes in the tenant, without needing specific account credentials to access a mailbox. Applications assigned these permissions can be reviewed by filtering the **PermissionType** column by **Application** on the **ConsentGrantData** tab generated within the Create-AppConsentGrantReport report (Figure 53).

Figure 53. Output to review assigned application permissions.

A	B	D	H	I	M	N
PermissionType	ConsentTypeFilter	ClientDisplayName	ResourceDisplayNameFilter	Permission	MicrosoftRegisteredClientA	Risk
Application	Application		Microsoft Graph	Mail.ReadWrite	FALSE	High
Application	Application		Microsoft Graph	Mail.ReadBasic.All	FALSE	High
Application	Application		Microsoft Graph	Mail.Read	FALSE	High
Application	Application		Microsoft Graph	Mail.Send	FALSE	High
Application	Application		Microsoft Graph	Mail.ReadBasic	FALSE	Medium

While some legitimate applications require this privilege grant, they commonly do not need access to every mailbox. To limit which mailboxes the application can access, it is recommended to apply an Application Access Policy to the application.

For more information on configuration Application Access Policies, reference:

<https://docs.microsoft.com/en-us/powershell/module/exchange/new-applicationaccesspolicy?view=exchange-ps>

3. Require Admin Consent to Apps

- Login to <https://portal.azure.com>
- Select **Enterprise Applications**
- Select **User Settings**
- Set Users can consent to apps accessing company on their behalf to **No**
- Users can consent to apps accessing company data for the groups they own to **No**
- Set Users can request admin consent to apps they are unable to consent to **No**
- Select **Consent and permissions**
- Under User Consent for Applications select **Do not allow user consent**
- Under Group owner consent for apps accessing data select **Do not allow group owner consent**

For more information regarding illicit application consent grants, reference:

<https://docs.microsoft.com/en-us/microsoft-365/security/office-365-security/detect-and-remediate-illicit-consent-grants?view=o365-worldwide>

Step 5: Enforce multi-factor authentication (MFA) for Accounts

To protect against attacks that take advantage of single factor authentication for access to Microsoft 365 services, MFA should be enforced for all accounts with access to a Microsoft 365 tenant. At a minimum, MFA should be used for administrative access to a Microsoft 365 tenant, including when Exchange Online and Azure AD PowerShell management functionality is required.

For smaller organizations that need a quick baseline of enabled security settings (including MFA enforcement and blocking legacy authentication protocols) to create a hardened security posture for Microsoft 365, the grouping of settings known as “**Security Defaults**” should be considered. Security Defaults are free (available at any tier) and replace the older and deprecated baseline policies.

Some notes and caveats when using Security Defaults:

- If Conditional Access policies are currently being leveraged (Azure AD Premium P1 license), Security Defaults should not be used.
- If an organization has complex security requirements, Conditional Access policies should be used over Security Defaults.
- If an organization has Azure Active Directory Premium licenses, Security Defaults should not be used.
- If an organization uses Security Defaults and then decides to migrate to using Conditional Access policies, Security Defaults will first need to be disabled.

Step 6: Review all registered MFA devices

As a proactive best practice, organizations should develop a process to perform periodic reviews of all registered MFA devices to identify any potentially malicious registrations. For any devices that do not align to an expected registration, this process may potentially require re-enrollment for privileged and/or user-based accounts.

Note: This process should also include reviewing any accounts that can bypass MFA requirements, to include evaluating the operational impact and associated risk of leveraging such a configuration.

The PowerShell script syntax noted in Figure 54 can be used to generate a report of MFA devices registered with Azure AD.

Figure 54.

PowerShell script to identify all MFA devices registered with Azure AD.

```
Connect-MsolService
$Result=@()
$users = Get-MsolUser -All
$users | ForEach-Object {
    $user = $_
    $mfaStatus = $_.StrongAuthenticationRequirements.State
    $phoneApp = $_.StrongAuthenticationPhoneAppDetails
    $methodTypes = $_.StrongAuthenticationMethods

    if ($mfaStatus -ne $null -or $methodTypes -ne $null)
    {
        if($mfaStatus -eq $null)
        {
            $mfaStatus='Enabled (Conditional Access)'
        }
        $authMethods = $methodTypes.MethodType
        $defaultAuthMethod = ($methodTypes | where{$_.IsDefault -eq "True"}).
        MethodType
        $verifyEmail = $user.StrongAuthenticationUserDetails.Email
        $phoneNumber = $user.StrongAuthenticationUserDetails.PhoneNumber
        $alternativePhoneNumber = $user.StrongAuthenticationUserDetails.
        AlternativePhoneNumber
    }
    Else
    {
        $mfaStatus = "Disabled"
        $defaultAuthMethod = $null
        $verifyEmail = $null
        $phoneNumber = $null
        $alternativePhoneNumber = $null
    }

    $Result += New-Object PSObject -property @{
        UserName = $user.DisplayName
        UserPrincipalName = $user.UserPrincipalName
        MFAStatus = $mfaStatus
        AuthenticationMethods = $authMethods
        DefaultAuthMethod = $defaultAuthMethod
        MFAEmail = $verifyEmail
        PhoneNumber = $phoneNumber
        AlternativePhoneNumber = $alternativePhoneNumber
        DeviceName = $phoneApp.DeviceName
    }
}
$Result | Select
UserName,UserPrincipalName,MFAStatus,DefaultAuthMethod,MFAEmail,
PhoneNumber,AlternativePhoneNumber,DeviceName | export-csv MFAReport.CSV
```

Step 7: Review Partner (Cloud Service Provider) Relationships

Organizations should review any configured partner relationships that exist within a Microsoft 365 tenant. Partner relationships can exist as various partner types, including resellers, advisors and delegated administrators.

The scope of permissions assigned to partners should be reviewed and verified – as partners can exist as Global Admin or Helpdesk Admin roles. Delegated admin roles (Admin Agents and Helpdesk Agents) are also available when partner relationships are present. Based on the roles assigned, members of either groups can access the customer's Azure AD tenant and Microsoft 365 services using their partner credentials and perform administrative actions on behalf of the customer.

[Per Microsoft](#), when a customer grants delegated administration privileges to a partner:

- The Admin Agent group is assigned to the Global Administrator role in the customer's Azure AD tenant.
- The Helpdesk Agent group is assigned to the Helpdesk Administrator role in the customer's Azure AD tenant.

1. Review Microsoft 365 Partner Relationships that have Delegated Admin Privileges

- **Option 1:** Navigate to <https://admin.microsoft.com/AdminPortal/Home#/partners> within the in Microsoft 365 Admin Center.
- **Option 2:** Using PowerShell (Figure 55), review any present **DapEnabled** values using the **Get-MsolPartnerInformation** command

2. Remove DapEnabled Cloud Service Provider Delegated Admin Privileges

Any cloud service provider delegated administrative privileges should be reviewed and potentially removed. Proactively, an organization would want assurance from the cloud service provider that the provider's network and Microsoft 365 tenant are secured and hardened appropriately.

Figure 55.

PowerShell command to review partner relationships.

```
Connect-MsolService  
  
Get-MsolPartnerInformation
```

Conclusion

While [UNC2452](#) has demonstrated a level of sophistication and evasiveness, the observed techniques are both detectable and defensible. This white paper outlines steps that represent actionable measures that can be leveraged to not only remediate observed techniques, but also to empower organizations to detect and harden against similar tactics.

As new information is derived from Mandiant's frontline visibility, this white paper will be updated with additional techniques and the associated detection, remediation and hardening measures.

To learn more about FireEye, visit: www.FireEye.com/mandiant

FireEye, Inc.

601 McCarthy Blvd. Milpitas, CA 95035
408.321.6300/877.FIREEYE (347.3393)
info@FireEye.com

©2020 FireEye, Inc. All rights reserved.
FireEye and Mandiant are registered trademarks of FireEye, Inc. All other brands, products, or service names are or may be trademarks or service marks of their respective owners.
I-EXT-WP-US-EN-000343-03

About Mandiant Solutions

Mandiant Solutions brings together the world's leading threat intelligence and frontline expertise with continuous security validation to arm organizations with the tools needed to increase security effectiveness and reduce business risk.



Publish Date	Document Version	Affected Page(s)	Description
01/19/2021	v1.0	n/a	Original publication
3/18/2021	v1.1	5,6,17,33-37	<ul style="list-style-type: none">• Improved directions to detect read attempts of ADFS token-signing attribute• Added new attack technique: Modify Mailbox Folder Permissions