

# Azure Reconnaissance and Scanning for Ethical Hackers and Special Ops Team



"Everything and anything  
is hackable and vulnerable  
in some way."

# Table of Contents

Introduction.....	3
Footprinting and Scanning .....	3
Footprinting.....	4
Scanning.....	5
Gain Access.....	6
Azure Account .....	6
Service Principal Account .....	8
RBAC Authorization.....	10
The Azure Portal .....	11
The Azure Explorer .....	12
Examine the Naming Standards .....	12
Advanced Scanning Technique .....	12
Azure REST API.....	13
PowerShell Scanning Techniques .....	14
Advanced Coding Techniques.....	17
Covering the Tracks.....	22
Countermeasures and Best Practices.....	22
Summary.....	23
About the Author.....	24
About the Reviewers .....	25

## Introduction

This whitepaper is meant to provide a quick and practical guide to Ethical Hackers and Special Ops Team. There are many available tools and techniques for **scanning** and **reconnaissance**, and this whitepaper provides the most practical for Microsoft Azure.

The final objective of a Penetration Testing project is to provide useful information to resolve the errors identified before they can be used for malicious purposes. A Pentest activity must be considered as a project and requires the use of a methodology that helps the pentester team to perform the various steps in a correct and controlled way. A pentest can be an automated activity or performed in manual mode; in both cases, there are a series of steps that must be performed in sequence.

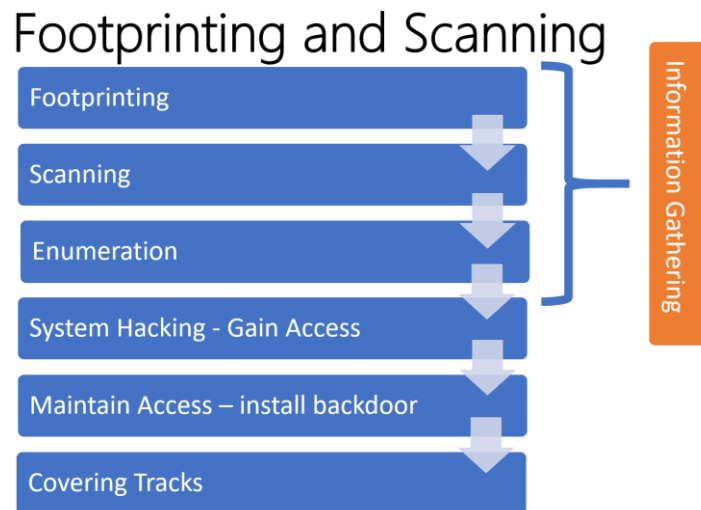
The first phase and perhaps the most important, **Reconnaissance/Footprinting**, consists of the recognition of the target and the acquisition of as much information as possible through actions that are often passive and do not involve direct interaction with the target's systems.

The reconnaissance and scanning are the first procedures to start with in order to identify our attack surface. Azure contains a large number of technologies, and these techniques may vary depending on what we need to scan and the scope. In this chapter, we will examine the most important techniques, the approaches, and the tools to use in relation to our scope and target.

## Footprinting and Scanning

In the cybersecurity space, the **Information Gathering** activities correspond to the operations aimed at researching, collecting, and organizing as much information as possible about a potential attack target. Then it will seek ways that could be exploited to get into the systems.

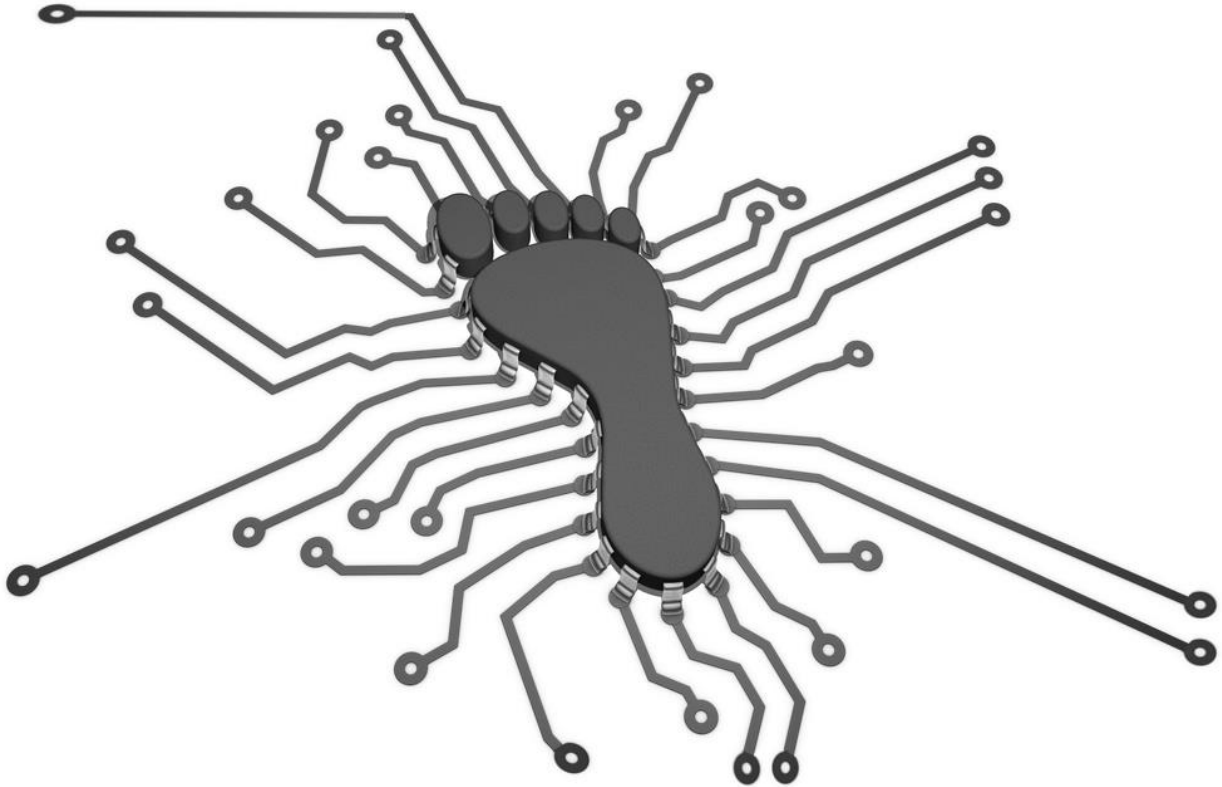
The phases shown in the pictures below are applicable both to malicious activities and to penetration activities tests and vulnerability assessments carried out for ethical hacking purposes.



For example, a pen tester or a hacker (ethical or not) can use specific tools, like port scanners tools, to help them get an excellent understanding of the potential target systems in the network and sometimes what operator system and software is on them. This information gathering endeavor may allow analysis to locate possible vulnerabilities. It can pinpoint what impact the different findings may have on the client and the subsequent exploitation phase, where the weaknesses are attempted to be exploited to get into the system.

## Footprinting

Footprinting means all observation, research, and collection of related information to a potential target. The aim is to have a vision, as complete as possible, of the activities and the equipment of the chosen victim of a cyber attack.



In this phase, the information is collected and organized so that it can be analyzed subsequently, with the aim of finding a viable attack methodology. Below is a non-exhaustive list of information usually collected during the footprinting phase:

- Generic information on the organizational structure of the target (employees, managers, CEO, and so on.);
- Information on the operating systems used (system names, users and groups, passwords, and so on);
- Information on the network (Domain Name, IP Addresses, Networking Protocols, and so on);
- Internet sites and domain names owned by the victim;
- Telephone numbers, emails and other contacts available via footprinting;

In essence, it is a matter of **collecting any useful information** that may, directly or indirectly, facilitate the hacking of systems or networks to be targeted.

We also distinguish two modes of footprinting:

- **Passive:** the information is collected without in any way contacting the victim organization.
- **Active:** information is collected by making direct contact with the potential victim using social engineering techniques.

If an attacker manages to present himself in disguise, for example, as a journalist, e contacts the victim's staff, the information collected in the interview is an example of **active footprinting**.

**Passive footprinting** techniques are much less invasive, resulting in the use of search engines, information posted by the staff of the target on social media, physical addresses of the various locations (where the target has more than one), and etcetera.

## Scanning

It essentially consists of the active verification of the information gathered in the previous phase, in order to determine if and which potential vulnerabilities are exploitable by the attacker.



In this sentence, only target verification activities are implemented, and no real attack is currently implemented. The purpose is to collect reliable information on active and exposed servers on the internet as well as on exposed services. It is emphasized that this phase actively involves the attacker and the attacked systems.

Below some types of scans, usually implemented during this phase:

- Network scan;
- Port scan;
- Vulnerability scan;
- Banner grabbing;

Nevertheless, these activities can put the possible attacker at risk, as he must actively expose himself to its target. In fact, numerous port-scan techniques are detected by the most common IDS systems, in addition to non-session sessions standards towards the network of the potential victim, such as "telnet", always leave traces and etcetera.

The attacker in this phase will, therefore, have to guarantee an adequate infrastructure, both in terms of size and of the technical skills required to carry out the activities. All of this is also applying appropriate evasion techniques from the victim's monitoring systems, with a considerable expense also of time. In the case of targeted attacks on infrastructures of considerable size and complexity, the resources required are huge.

## Gain Access

And of course, the first we need to do is gain access. And for that, we can use two types of ways:

- using the **Azure Account**: The Azure Account is the best option to use in tools like the Azure Portal and other UI's.
- Or the **Principal Account**: The Principal Account is the best option to use to run our scripts and tools.

For a good penetration test, I recommend to use both, and we need to use any possible tool.

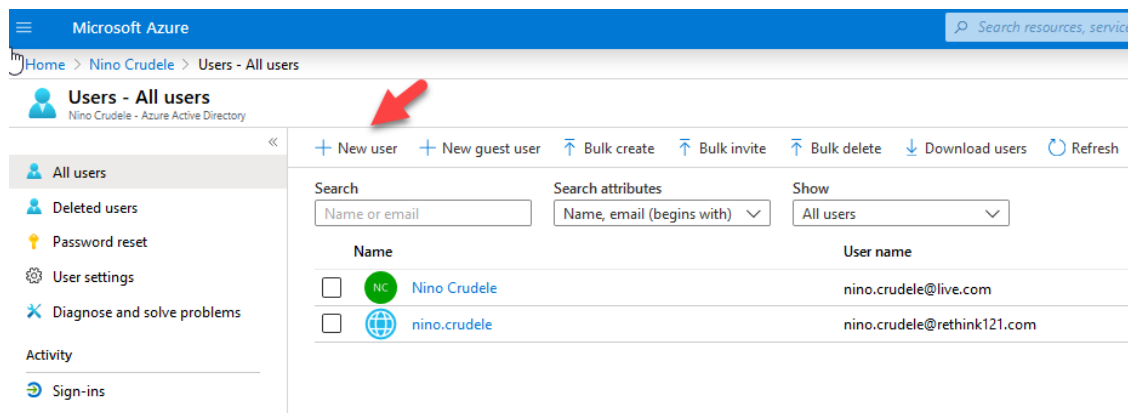


## Azure Account

The Azure Account, is in fact, an Azure Active Directory account. To create the **Azure Account**, we need Global admin permissions, usually, the customer will do that for us, but it is useful to know how to do that.

To create the account, we need to:

- Access to **Azure Active Directory**, select **Users**, and click **+ New User**.



- On the New user screen we need to set the following fields and then click **Create**:

- **User name** in this sample testuser;
- **Name** in this sample testuser also;
- and a password;

Microsoft Azure

Home > Nino Crudele > Users - All users > New user

**New user**  
Nino Crudele

Got a second? We would love your feedback on user creation →

☒ **Create user**  
 Create a new user in your organization. This user will have a user name like `alice@ninocrudelelive.onmicrosoft.com`.  
[I want to create users in bulk](#)

☐ **Invite user**  
 Invite a new guest user to collaborate with your organization. The user will be emailed an invitation they can accept in order to begin collaborating.  
[I want to invite guest users in bulk](#)

[Help me decide](#)

**Identity**

User name \*  @

The domain name I need isn't shown here

Name \*

First name

Last name

**Password**

☐ Auto-generate password

☒ Let me create the password

Initial password \*

**Groups and roles**

Groups [0 groups selected](#)

[Create](#)

**Note:** You can also ask the customer to invite you, and this is possible selecting **Invite user** and setting the email address, see below.

Microsoft Azure

Home > Nino Crudele > Users - All users > New user

**New user**  
Nino Crudele

Got a second? We would love your feedback on user creation →

☐ **Create user**  
 Create a new user in your organization. This user will have a user name like `alice@ninocrudelelive.onmicrosoft.com`.  
[I want to create users in bulk](#)

☒ **Invite user**  
 Invite a new guest user to collaborate with your organization. The user will be emailed an invitation they can accept in order to begin collaborating.  
[I want to invite guest users in bulk](#)

[Help me decide](#)

**Identity**

Name

Email address \*

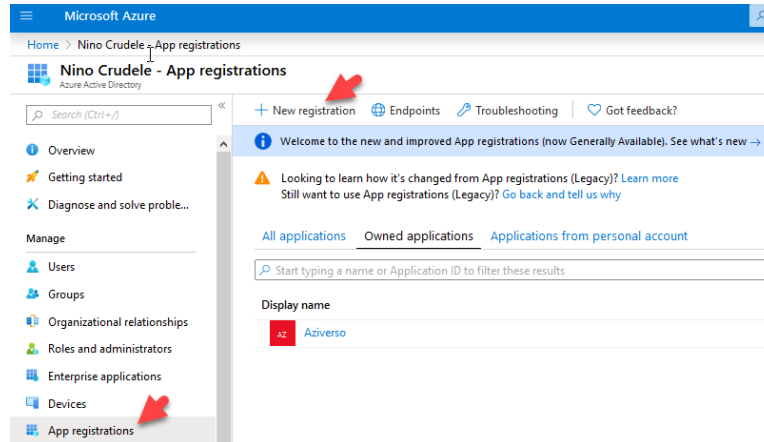
This second option is the most used, but I would recommend the customer **create a new user** and just **delete the user when the pentest will be completed**.

## Service Principal Account

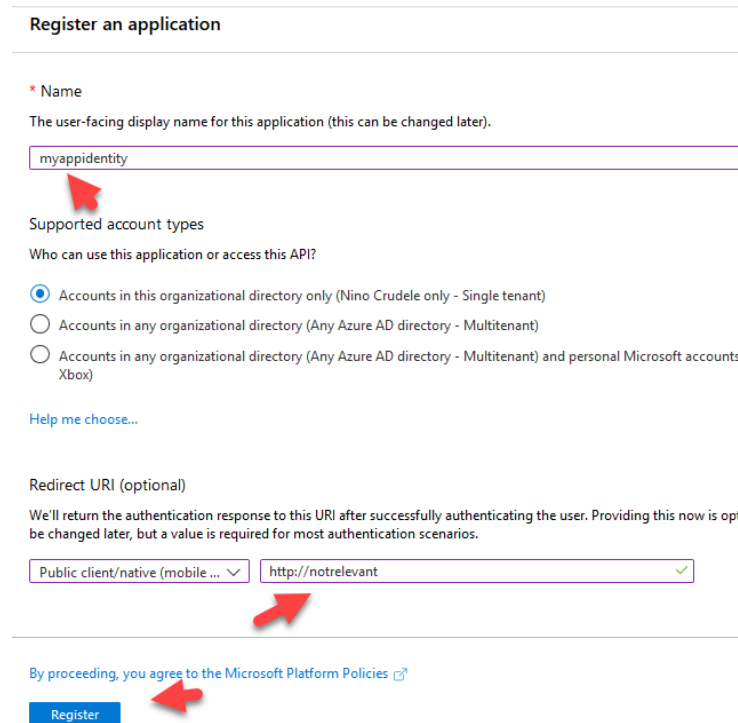
Besides the Azure Account, we have the **Azure Service Principal** which is a security identity used by user-created apps, services, and automation tools to access specific Azure resources. Think of it as a 'user identity' (login and password or certificate) with a specific role, and tightly controlled permissions to access your resources. It only needs to be able to do specific things, unlike a general user identity. It improves security if you only grant it the minimum permissions level needed to perform its management tasks.

To create a **Principal Account**, you need to:

- Select **Azure Active Directory**, click on **App registrations**, and then click on **+ New registration**.



- On the New user screen we need to set the following fields and then click **Register**:
  - **Name** in this sample myappidentity
  - and from the **Redirect URI** combo-box select the option **Public client/native**
    - Note: the URI value is not important;



**Register an application**

\* Name  
The user-facing display name for this application (this can be changed later).

Supported account types  
Who can use this application or access this API?

☒ Accounts in this organizational directory only (Nino Crudele only - Single tenant)  
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)  
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (Xbox)

[Help me choose...](#)

Redirect URI (optional)  
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional, but a value is required for most authentication scenarios.

By proceeding, you agree to the Microsoft Platform Policies [☐](#)



- When the application identity is created, select the identity and, on the **Certificates & secrets** tab, click on **New client secret** to generate the key.

**myappidentity - Certificates & secrets**

Search (Ctrl+/) <<

Overview  
Quickstart  
Manage  
Branding  
Authentication  
**Certificates & secrets**  
API permissions  
Expose an API  
Owners  
Roles and administrators (Prev...  
Manifest  
Support + Troubleshooting  
Troubleshooting  
New support request

Credentials enable applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

**Certificates**  
Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

[Upload certificate](#)

No certificates have been added for this application.

Thumbprint	Start Date	Expires
------------	------------	---------

**Client secrets**  
A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

[New client secret](#)

Description	Expires	Value
-------------	---------	-------

No client secrets have been created for this application.

- On the **Add a client secret** panel set the **Key name** (description) and the **expiration time** and click the **Add** button;

**Add a client secret**

Description  
mykey

Expires  
☒ In 1 year  
☐ In 2 years  
☐ Never

[Add](#) [Cancel](#)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

[New client secret](#)

Description	Expires	Value
-------------	---------	-------

No client secrets have been created for this application.

After that, the key will be generated. Now you need to save the key in a safe place because you will need to use it to log in.

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

[New client secret](#)

Description	Expires	Value
mykey	11/28/2020	0WUB7Ko0?tnZ3c?wktFucK:A8w7=hP

## RBAC Authorization

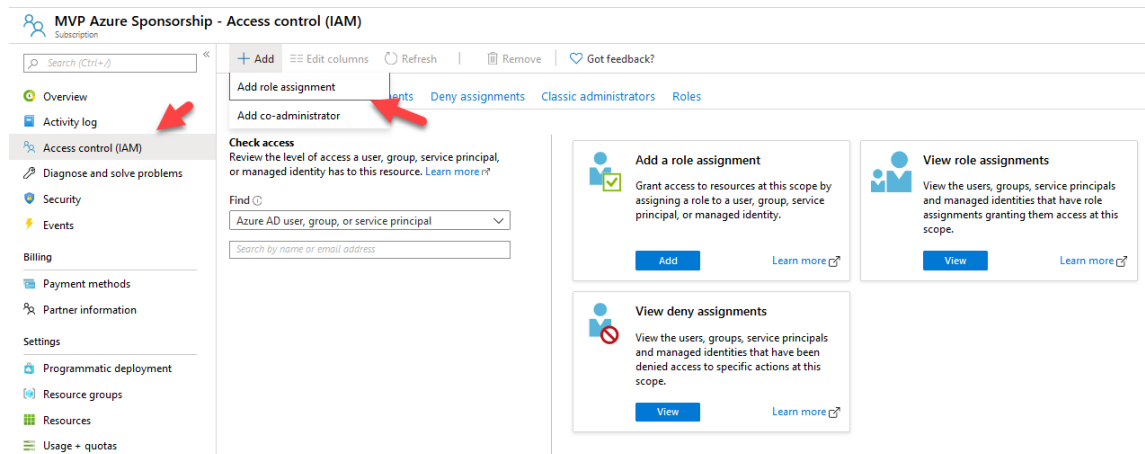
We now need to provide access to our account, and it can be the **Azure Account** or the **Principal Account**, the method will be the same. **Role-based access control (RBAC)** is a method of regulating access to a computer or network resources based on the roles of individual users within an enterprise.

The way you control access to resources using RBAC is to create role assignments. This is a key concept to understand – it's how permissions are enforced. A role assignment consists of three elements:

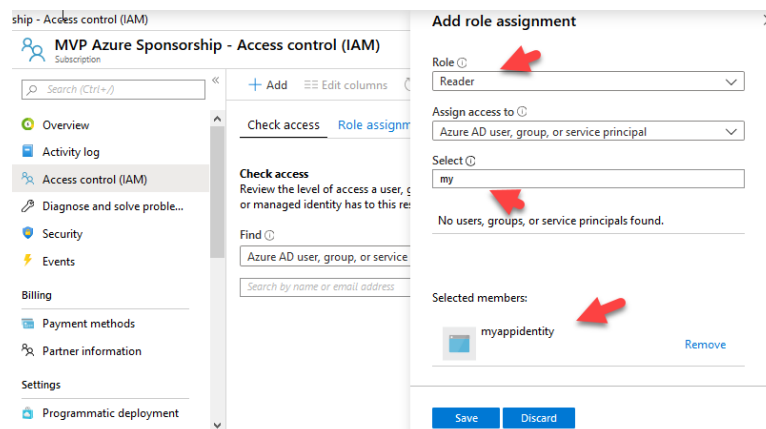
- **Security principal:** A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.
- **Role definition:** A role definition is a collection of permissions. It's typically just called a role. A role definition lists the operations that can be performed, such as read, write, and delete. Roles can be high-level, like the owner, or specific, like a virtual machine reader.
- And **scope:** Scope is the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if you want to make someone a Contributor, but only for one resource group.

To give access to our the **Azure Account** or the **Principal Account** we need to:

- Select your subscription, click on **Access control (IAM)** and **Add role assignment**.



- Select the **Role** you like to apply. I recommend you to use the **Reader** role, search for our new application identity, and click **Save**.



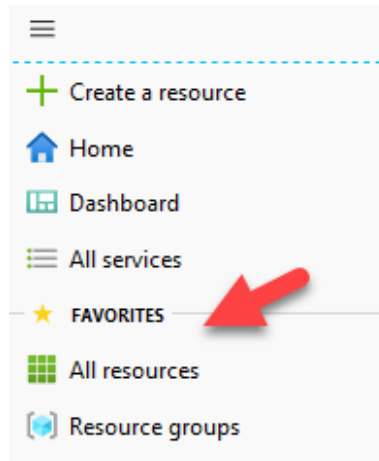
Now the account has reader access in the entire subscription, and we can use the same method with the Root Management group to gain access in all the subscriptions in the tenant.

We now have the Azure account. Let see how to use it.

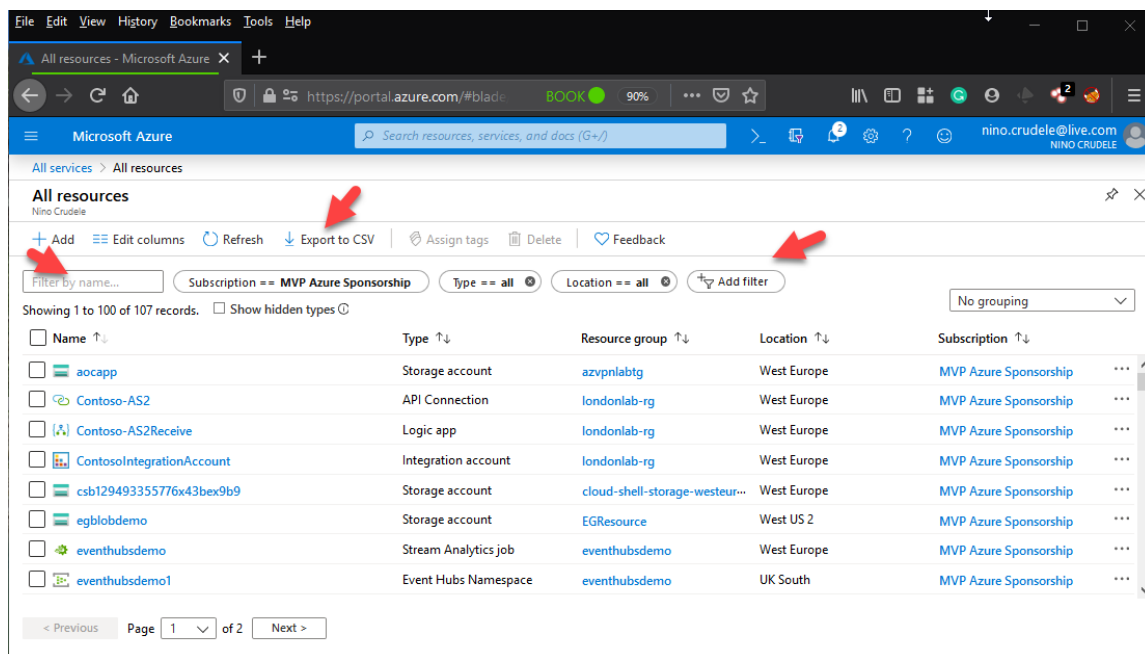
## The Azure Portal

The Azure Portal is an excellent UI for reconnaissance, we can log into the portal using our Azure account, and we can use some great searching techniques.

For example, we can enter into the portal, click on the **three lines** on the top left, and select **All resources**.

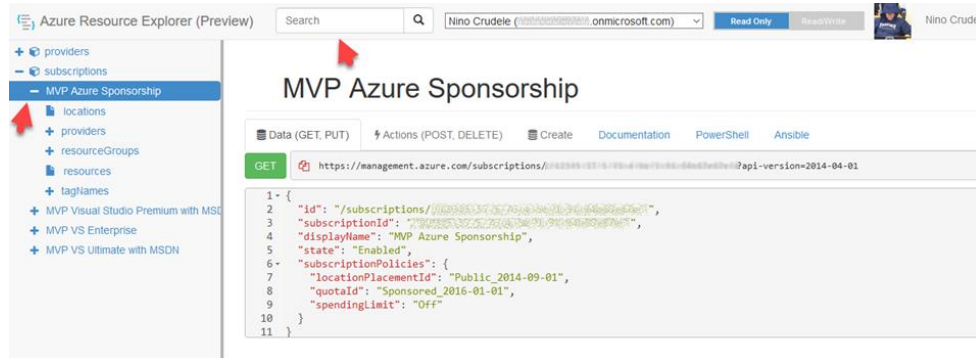


Now we can execute any complex query using filters, and we can also export these results in CSV. See the picture below showing you the filters.



## The Azure Explorer

Another great UI for reconnaissance is the Azure Explorer. We can navigate on <https://resources.azure.com/subscriptions>, login with our account, and select the **plus** on the left of your subscription, as you may see in the picture below:



We can also search for resources and much more.

The Azure Explorer is showing a raw representation of the Azure infrastructure.

## Examine the Naming Standards

A good (it may or not be ethical) hacker must consider any option to exploit his target, and the naming standard is one of these. The use of a good naming standard strategy is a good practice; on the other side, it is also an excellent opportunity and a good source of information.

The customer often uses projects and department names, and we can understand some business logic just looking at the name of the resources.

The department name is used at a management group level, and the subscriptions usually contain the project name or product name. Often people use the resource groups to organize internal teams and people, features or projects, and you may find some interesting information in there also.

You can find an official guideline at the here: <https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/naming-and-tagging>

The Microsoft guideline is providing just a little portion of that, but it covers the most used resource types.

## Advanced Scanning Technique

First, hackers need to find a **target** to hit and plan how to set up the **attack**. Generally, the reasons that move the hands of hackers are always the same: **economic reasons**, the possibility of appropriating precious data, or damaging the company. Once the target has been identified, the **study phase** begins.

Cybercriminals give priority on gathering information about the company and especially its security system. Hackers use a different technique, and social engineering is one of the most effective, I personally like to use it. The target is usually an internal employee in the company, and the goal is to find the big vulnerability of the person and use it to exploit it.

We all use socials, and unfortunately, many people don't realize how much they expose their personal life. In this specific case, the hacker will attack the employee, and he will use a phishing technique or ransomware to obtain what he needs.



For example, the hackers create a fake website, visit the structure, and also attend the events organized by the company, with the aim of knowing everything about the target. At this point, the planning of the attack starts. The hacker usually forces the target to collaborate with it, providing a password, or installing a rootkit.

An advanced scanning technique is used to identify possible targets and vulnerabilities. In this chapter, we will examine some of them. In the other chapters, we will also speak about the nastiest and elaborated.

The Azure REST API is one of the most used and powerful interfaces offered by Microsoft.

## Azure REST API

As I mentioned, the **Azure REST API** is the most used API interface in Microsoft Azure. The APIs expose any possible action type like list resources and any CRUD operation on any Azure resource type, with no exceptions.

Using the Azure REST API, we can almost do everything and anything, and for that reason, it is crucial to know how to use them.

The **Azure REST API reference** is the official portal. From this portal, we can search for any API, and we can get any relevant information. In the picture below, you can see an example of an API call to list all Virtual Network in a Subscription.

Microsoft Azure

Overview Solutions Products Documentation Pricing Training Marketplace Partners Support Blog More

Docs / Azure Stack Admin / Virtual Networks / List

Filter by title

- > Storage Systems
- > Subscriber Usage Aggregates
- > Subscriptions
- > Table Services
- > Update Encryption (Commercial)
- > Update Encryption (SubscriptionsAdmin)
- > Update Locations
- > Update Rules
- > Updates
- > VM Extensions
- > Virtual Networks
  - List
- > Volumes
- > Batch AI
- > Batch Management
- > Batch Service
- > Billing
- > Blockchain
- > Blueprints

### Virtual Networks - List

Service: Azure Stack Admin  
API Version: 2015-05-15

Get a list of all virtual networks.

HTTP [Copy](#) [Try it](#)

GET <https://adminmanagement.local.azurestack.external/subscriptions/{subscriptionId}/providers/Microsoft.Network.Admin/a>

With optional parameters:

HTTP [Copy](#)

GET <https://adminmanagement.local.azurestack.external/subscriptions/{subscriptionId}/providers/Microsoft.Network.Admin/a>

#### URI Parameters

Name	In	Required	Type	Description
subscriptionId	path	True	string	Subscription credentials which uniquely identify Microsoft Azure subscription. The subscription ID forms part of the URI for every service call.
api-version	query	True	string	Client API Version.

## Virtual Networks - List

Get a list of all virtual networks.

```
HTTP GET https://adminmanagement.local.azurestack.external/subscriptions/{subscriptionId}/providers/Microsoft.Network.
x
```

With optional parameters:

```
HTTP GET https://adminmanagement.local.azurestack.external/subscriptions/{subscriptionId}/providers/Microsoft.Network.
x
```

Name	In	Required	Type	Description
<b>subscriptionId</b>	path	True	string	Subscription credentials which uniquely identify Microsoft Azure subscription. The subscription ID forms part of the URI for every service call.
<b>api-version</b>	query	True	string	Client API Version.
<b>\$filter</b>	query		string	OData filter parameter.
<b>\$orderBy</b>	query		string	OData orderBy parameter.
<b>\$top</b>	query		string	OData top parameter.
<b>\$skip</b>	query		string	OData skip parameter.
<b>\$inlineCount</b>	query		string	OData inline count parameter.

# REST API Try It

Try the REST API with the inputs below.

Sign out

Request URL

GET https://adminmanagement.local.azurestack.external/subscriptions/e45dfe82-9f21-47d4-8619-0a186e72e4e3/providers/

Parameters

subscriptionId\*

MVP VS Enterprise

api-version\*

2015-06-15

name

value

+

Headers

name

value

+

Request Preview

HTTP

Copy

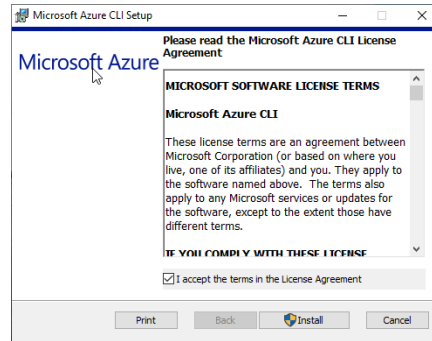
GET https://adminmanagement.local.azurestack.external/subscriptions/e45dfe82-9f21-47d4-8619-0a186e72e4e3/providers/Microsoft.Authorization?Bearer=eyJ0bm9jaWJ0YXNjcG10c2l1bnR1ZmlzIng1aC10bGVjCOENLR1IzeWpPhKx0doHodk1pTDRwZmpleyIsImtpcyCC18InQCOENLR1I

< [REDACTED] >

Run ▶

## PowerShell Scanning Techniques

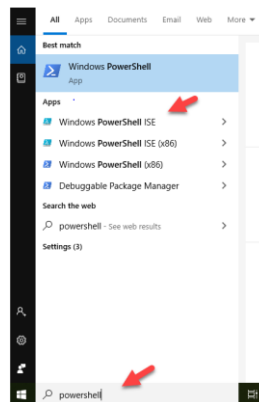
- Execute the file **azure-cli-<version>.msi**
- Accept the agreements as below and click the **Install** button.



You can check all possible functions you can use on the Azure CLI portal: <https://docs.microsoft.com/en-us/cli/azure>

We now have Azure CLI installed. To test it, we need to open PowerShell ISE. To do that we need:

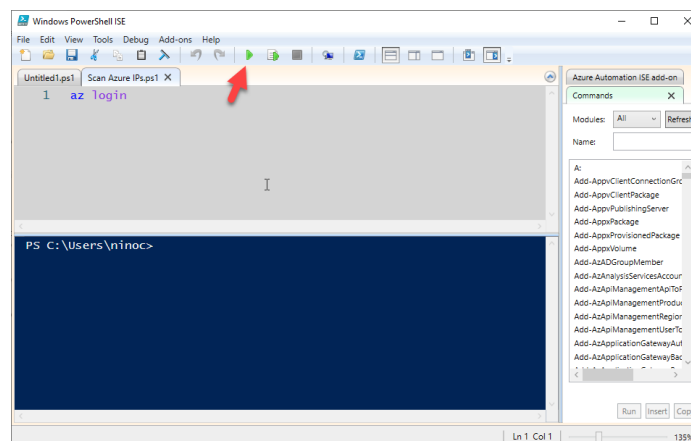
- To search the **PowerShell ISE** in Windows search and click on **Windows PowerShell ISE**, see the picture below.



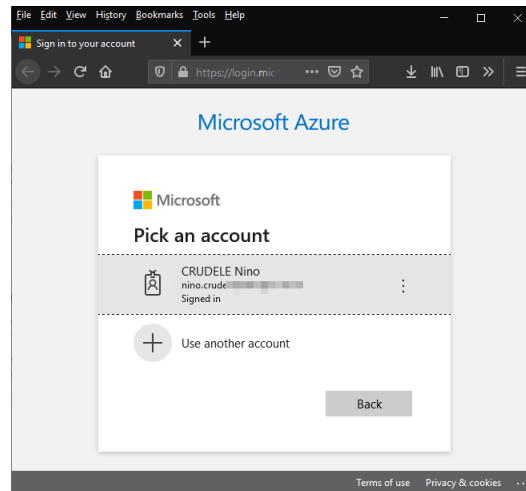
- In the console, panel write the Azure CLI command

```
az login
```

- Click on **File** → **Save** and name the file, for example, **Scan-Azure-IPs.ps1**.
- Now click on the green run button to execute the command.



- The command will open the browser, log in to Microsoft Azure, by clicking on the account, if you have already signed before, or enter your Azure account credentials.



Perfect, we have installed Azure CLI, and we have signed in our Azure environment. Now we can start writing our scanning script.

The script below will collect all the public IP addresses in all the subscriptions, and it executes an Nmap scanning.

```
Write-Output "Login into the Azure Account"
az login

#Variables
$ipListFileInput = "C:\" + $subscriptionName + "_ipList.txt"
$ipListFileOutput = "C:\" + $subscriptionName + "_ipListoutput.txt"

Write-Output "List subscriptions"
#Filtering the output
$allsubscriptions = az account list --query '[][.id, name]' -o tsv

#Cycling for each subscription
foreach ($subscription in $allsubscriptions) {
    $arrValues = $subscription.Split(" ")
    $subscriptionId = $arrValues[0]
    $subscriptionName = $arrValues[1]
    Write-Output "Get IP for subription id " + $subscriptionId.ToString()
    $allPublicIP = az network public-ip list --subscription $subscriptionId --query '[][.ipAddress]' -o tsv
    #Create the empty array for ips
    $OutputArray = @()
    #Cycling for each ip
    foreach ($ip in $allPublicIP){
        $OutputArray += $ip
    }

    Write-Output "write to file"
    [System.IO.File]::WriteAllLines($ipListFileInput, $OutputArray)

    Write-Output "Execute the scan"
    nmap -v -p 1-65535 -sv -O -sS -T5 -iL $ipListFileInput | Out-File $ipListFileOutput
}
```



THIS POWERSHELL SCRIPT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND.

We save all public IP found using the command below:

```
[System.IO.File]::WriteAllLines($ipListFileInput, $OutputArray)
```

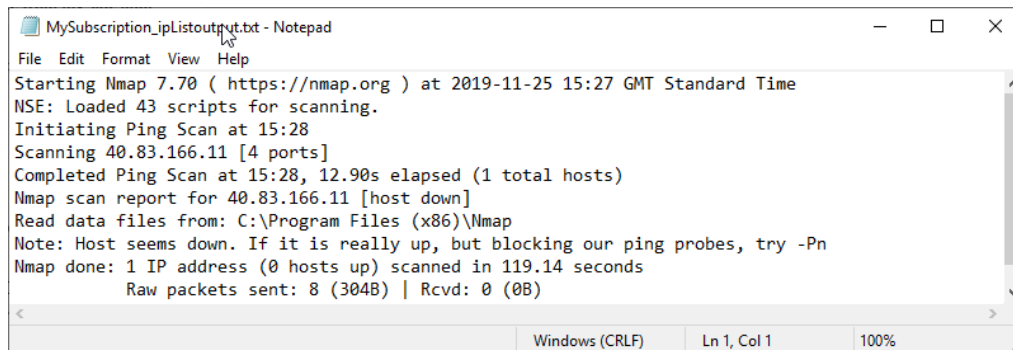
It is important to know that the command WriteAllLines will write a file without the Byte Order Mark on top, something the other PowerShell commands do.



After we have the list, we can now execute a massive scan of the IP addresses using the **Nmap** command -iL on the file created before.

```
nmap -v -p 1-65535 -sv -O -sS -T5 -iL $ipListFileInput | Out-File $ipListFileOutput
```

The Nmap command prints a verbose output, and it runs a stealth syn scan with T5 timing with OS detection, and the scan result will be saved in a file. Below an example of an output file created by the scan:



```
MySubscription_ipListOutput.txt - Notepad
File Edit Format View Help
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-25 15:27 GMT Standard Time
NSE: Loaded 43 scripts for scanning.
Initiating Ping Scan at 15:28
Scanning 40.83.166.11 [4 ports]
Completed Ping Scan at 15:28, 12.90s elapsed (1 total hosts)
Nmap scan report for 40.83.166.11 [host down]
Read data files from: C:\Program Files (x86)\Nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 119.14 seconds
Raw packets sent: 8 (3048) | Rcvd: 0 (0B)
```

We can use this mixing technique between PowerShell and other hacking tools in different combinations. The important thing to understand is that the use of hacker tools is not more important than the use of our abilities to scanning the Azure infrastructure. Tools like Nmap, Metasploit, or SQLMap, and many others don't have any real capability of scanning at the infrastructure level.

**The hacking tools usually don't implement any Azure capabilities, and we need to mix tools, code, and scripting techniques to be really effective.** We first scan the infrastructure to collect the endpoints, and we use the specific hacking tool to scan each endpoint found.

Another great example is this Azure CLI command that lists all the subscription, where the account has access, and extract the id and the name only.

```
#Let make some important considerations regarding the script above.
#Something I really like about Azure CLI is the possibility to query the data, check the line below.
$allsubscriptions = az account list --query '[].[id, name]' -o tsv
```

We will now examine some advanced techniques.

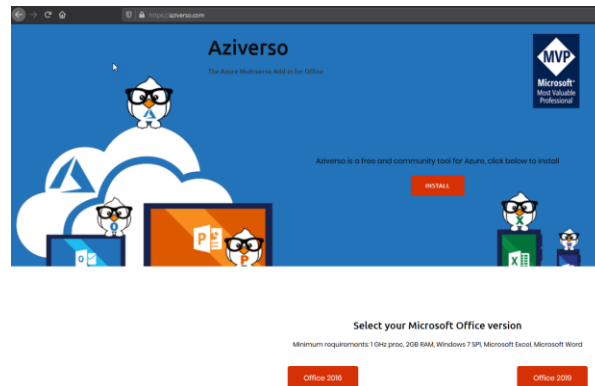
## Advanced Coding Techniques

We can implement more advanced code techniques, and in fact, we can integrate them on custom tools or in other commercial products like Excel. An excellent example of this is my implementation of the Nmap scanning capabilities in [Aziverso](#). Using this technique, I can scan any IP address in any subscription directly from Microsoft Excel and create a handy report for the teams and the customer.

But first of all, what is Aziverso? Aziverso is a free add-in for Microsoft Office composed of many features that can speed up the daily job to any Azure professional in the most challenging and critical situations. It offers excellent capabilities to manage the Azure costs, troubleshooting, naming standards, and much more. Aziverso is a smart tool for smart people. The internal features can be combined in a different way to solve many situations and problems.

To show this in action, we will now perform a full scan using Aziverso, and I will also explain to you the technical challenge behind that. So, to do that we first need to download and install the Excel add-in, we can do that by:

- Navigate to Aziverso web site <https://aziverso.com>.



- Select the appropriate Office version you need and install the tool.

After the tool is installed, the first step we need is to identify the scope of our scanning. For that reason, we will be listing all the subscriptions we want to scan.

- Click on **Add-Ins** and then click on **Subscriptions**. Now we have the list of subscriptions we want to scan.

id	subscriptionid	displayName	state	authorizationSource	locationPlacementid	quotaId	spendingLimit
1	e3	MVP VS Enterprise	Enabled	RoleBased	Public_2014-09-01	MSDN_2014-09-01	On
2	0e	MVP Azure Sponsorship	Enabled	RoleBased	Public_2014-09-01	Sponsored_2016-01-01	Off
3	3d	MVP Visual Studio Premium with MSDN	Enabled	RoleBased	Public_2014-09-01	MSDN_2014-09-01	On
4	8d	MVP VS Ultimate with MSDN	Enabled	RoleBased	Public_2014-09-01	MSDN_2014-09-01	On

- **Note:** we can remove the subscriptions we don't want from the list by simply delete the row on the Excel sheet.

Now we will use the Recon to retrieve all Public IPs in all the subscriptions. The recon is a very smart feature. It lists more than 2000 Azure APIs, and we can call any one of them using any Excel sheets to map the values we want to pass to the API. We will be speaking more about this type of advanced techniques in the book.

- Click on **Recon** and select the Subscription Excel Sheet.

mapped

filter for params available for the call only

Refresh

Number of API calls: 2020

Get parameters from: Subscription-72726343

Filters and Options

One of the fields: Group Name Title Description Rest Call Rest Call Description

NetworkManagementClient - Microsoft.Network.publicIPAddresses

(1728) - Resource-manager - NetworkAdminManagementClient - Microsoft.Network.Admin-adminPublicAddresses

/subscriptions/{subscriptionid}/providers/Microsoft.Network/publicIPAddresses?api-version=2018-12-01

NetworkNetworkManagementClientThe Microsoft Azure Network management API provides a RESTful set of web services that interact with Microsoft Azure Networks service to manage your network resources. The API has entities that capture the relationship between an end user and the Microsoft Azure Networks service 2018-12-01Gets all the public IP addresses in a subscription.

☐ New Sheet for each Excel sheet row ☐ Clean it

Ok Close

- Click on the filter to clean up the list, as you can see, the tool will pick up the parameters from the Excel Sheet with any Azure API able to use them.
- Now search for the specific Azure REST API to list all the public IP addresses, write IP addresses into the text box
  - Select the first API, see the picture above, and click the **OK** button.
  - The scan will now start, and it will produce the list of any public IP address in any subscription in the list.

The picture below shows the list produced by the scan, and the blank is not assigned IP, which means not used.

SubscriptionName	subscriptionId	Filter	properties.ipAddress	name	id
MVP Azure Sponsorship	12			Plaz-VM-1-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			Plaz-VM-2-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		190	testgateway	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			internal	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		1.28	VMW2012R2DTS05-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		33.47	extnlb1ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			extnlb2ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			testnlb1-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			testnlb2-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			VMW2012R2DTS01-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12			VMW2012R2DTS01ip745	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		4.228	VMW2012R2DTS02-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		.76	VMW2012R2DTS03-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		.208	VMW2012R2DTS04-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e

You can now understand the importance of using the correct scanning technique, and there is not a hacking tool around able to provide these capabilities out of the box. We can achieve a similar result using PowerShell and Azure CLI but without the Excel integration and spending hours writing a not reusable script. Excel integration gives us also the opportunity to include a direct link to the resource.

Customers are extremely pleased by this type of scans because they are easily readable and usable by any person in the company. As you can see, the IP Excel sheet produced contains the column IP address; this is the column we need for Nmap.

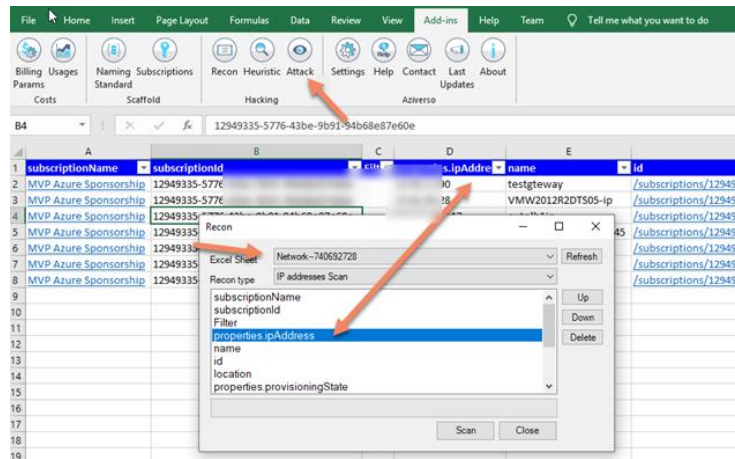
We can identify the unused resources checking the column **property.ipaddress**. A blank value means that the resource is not used. We can easily filter these values by using the Excel filter and eliminating them from the list.

See the picture below.

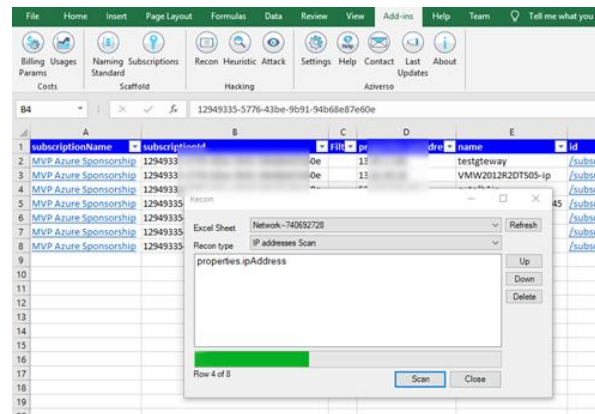
subscriptionName	subscriptionId	Filter	property.ipAddress	name	id
MVP Azure Sponsorship	12		13.1	testgateway	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		13.1	VMW2012R2DTS05-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		52.1	extnlb1ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		13.1	VMW2012R2DTS01ip745	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		13.1	VMW2012R2DTS02-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		13.1	VMW2012R2DTS03-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e
MVP Azure Sponsorship	12		13.1	VMW2012R2DTS04-ip	/subscriptions/12949335-5776-43be-9b91-94b68e87e60e

Now we can start the scan with Nmap by selecting **Attack** to pop up the scanning windows.

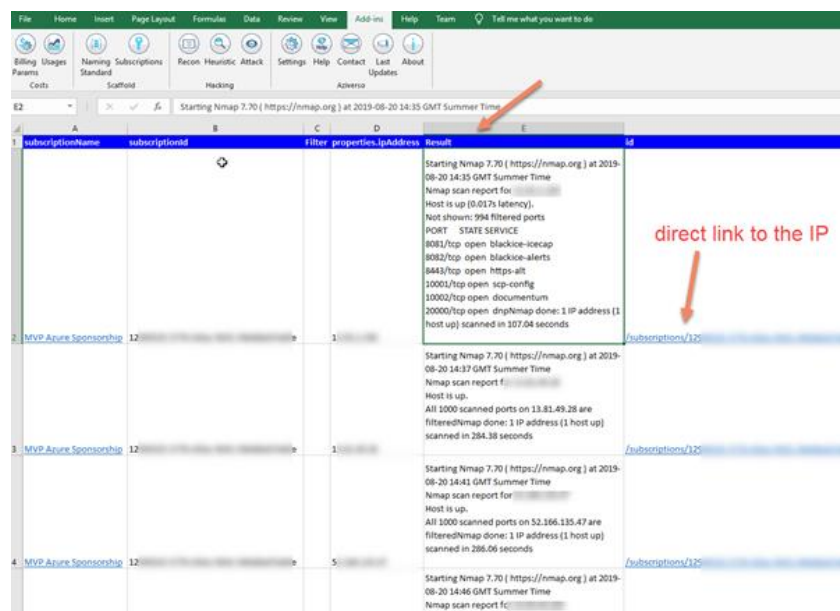
- In the windows, select the **Network Excel Sheet** and select **IP addresses Scan** in the Recon Combobox.
- Clean up all parameters and keep **the properties.ipAddress** only, this is the parameter used by Nmap.



- The scan will start. You can now take a coffee or do other things because this task may require some time. You can always check the status progress of the task.



- In the end, a report with all the vulnerabilities found for each IP will be produced.



Another great thing that Aziverso will also allow us to have a direct link for each specific Azure resource scanned, and we can directly jump into the portal from Excel. Once again, this thought for improving efficiency and timesaving.



At the time of writing this white paper, I was still working on this feature, and I was planning to release it to the public in a new version of the tool very soon. You may feel free to contact me for any questions and especially for collaboration, any kind of collaboration, code, design, hacking.

I have a roadmap of features planned that include many other hacking implementations like including advanced Nmap scans, Metasploit, Social-Engineer Toolkit, and with the inclusion of Windows Subsystem for Linux (WSL2), for example, performing a scan on all the App Service endpoints in the entire tenant.

The tool logs into <https://management.azure.com> to call any specific API. The code below shows the login implementation using:

```
using (var request = new HttpRequestMessage(method, newUrl))
{
    request.Headers.Authorization = new AuthenticationHeaderValue(scheme: "Bearer", parameter: token);

    var response = httpClient.SendAsync(request).Result;

    if (!response.IsSuccessStatusCode)
    {
        var errorMsg = "An error occurred! The service returned: " + response;

        var x = response.Content.ReadAsStringAsync();
        x.Wait();
        errorMsg += "Content: " + x.Result;
        throw new Exception(errorMsg);
    }

    var readTask = response.Content.ReadAsStringAsync();
    readTask.Wait();
    return readTask.Result;
}
```

The API will return a JSON stream, I flattened the JSON response and I filtered it for the appropriate fields.

```
private static void FillDictionaryFromJToken(Dictionary<string, object> dict, JToken token, string prefix)
{
    switch (token.Type)
    {
        case JTokenType.Object:
            foreach (var prop in token.Children<JProperty>())
            {
                FillDictionaryFromJToken(dict, prop.Value, Join(prefix, prop.Name));
                break;
            }

        case JTokenType.Array:
            var index = 0;
            foreach (var value in token.Children())
            {
                FillDictionaryFromJToken(dict, value, Join(prefix, name: index.ToString()));
                index++;
            }

            break;

        default:
            dict.Add(prefix, ((JValue) token).Value);
            break;
    }
}
```

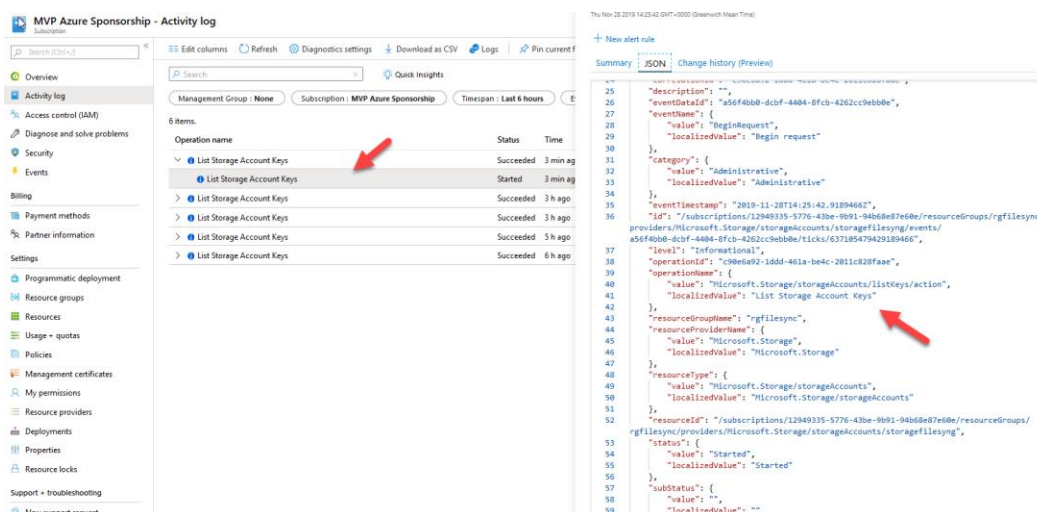


Using this technique, I can represent any information in the Excel Sheet.

## Covering the Tracks

Acting as a Red Team, you must operate as a real real simulation attack. Most of the operations we do in Azure are monitored and tracked. The last important operation that any ethical hacker must do is covering the tracks and cleanup the environment, in this way the Blue won't be able to conduct any useful investigation.

A scanning precludes a security issue, and any operation will be logged in Azure, in the picture below, you can see a log related to a Storage account List operation:



The screenshot displays the Azure Activity log for the 'MVP Azure Sponsorship' subscription. The left sidebar shows the 'Activity log' section selected. The main pane shows a list of operations, with 'List Storage Account Keys' highlighted. A red arrow points to this entry. The right pane shows the JSON details of the selected operation, with another red arrow pointing to the 'resourceName' field, which identifies the specific storage account.

Operation name	Status	Time
List Storage Account Keys	Succeeded	3 min ago
List Storage Account Keys	Succeeded	3 min ago
List Storage Account Keys	Succeeded	3 h ago
List Storage Account Keys	Succeeded	5 h ago
List Storage Account Keys	Succeeded	6 h ago

```
{
  "description": "",
  "eventDataId": "a56f4bb8-dcbf-4484-8fcb-4262c8ebbb6e",
  "eventTime": {
    "value": "BeginRequest",
    "localizedValue": "Begin request"
  },
  "category": {
    "value": "Administrative",
    "localizedValue": "Administrative"
  },
  "eventTimestamp": "2019-11-28T14:25:42.9189468Z",
  "id": "/subscriptions/12949335-5776-43be-9091-94b68e7e60e/resourceGroups/rgfilesync/providers/Microsoft.Storage/storageAccounts/storagefileysync/events/a56f4bb8-dcbf-4484-8fcb-4262c8ebbb6e/ticks/637385479425384660",
  "level": "Informational",
  "operationId": "c98e6a92-1ddd-461a-be4c-2811c828faae",
  "operationName": {
    "value": "Microsoft.Storage/storageAccounts/ListKeys/action",
    "localizedValue": "List Storage Account Keys"
  },
  "resourceGroupName": "rgfilesync",
  "resourceProviderName": {
    "value": "Microsoft.Storage",
    "localizedValue": "Microsoft.Storage"
  },
  "resourceType": {
    "value": "Microsoft.Storage/storageAccounts",
    "localizedValue": "Microsoft.Storage/storageAccounts"
  },
  "resourceId": "/subscriptions/12949335-5776-43be-9091-94b68e7e60e/resourceGroups/rgfilesync/providers/Microsoft.Storage/storageAccounts/storagefileysync",
  "status": {
    "value": "Started",
    "localizedValue": "Started"
  },
  "subStatus": {
    "value": "",
    "localizedValue": ""
  }
}
```

It is possible to delete an activity log alert using different techniques. We can call the Azure REST API below:

- <https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/microsoft.insights/activityLogAlerts/{activityLogAlertName}?api-version=2017-04-01>

the call below is a practical example:

```
DELETE https://management.azure.com/subscriptions/187f412d-1758-44d9-b052-169e2564721d/resourceGroups/Default-ActivityLogAlerts/providers/microsoft.insights/activityLogAlerts/SampleActivityLogAlert?api-version=2017-04-01#
```

We can also use PowerShell using the command `Remove-AzureRmActivityLogAlert`, an example below.

```
Remove-AzureRmActivityLogAlert -ResourceGroup "Default-Web-CentralUS" -Name "myAlert"
```

## Countermeasures and Best Practices

Our Azure infrastructure is daily scanned by hundreds of malicious companies and people across the world. It is a good practice to apply the proper countermeasure to hide any sensitive information from the public internet.

The best countermeasure and practice to avoid a scan is provided by a good security layer, and we need to limit access to our resources. The use of management groups is able to provide great flexibility.

It is more important to educate the employee to identify possible threats and especially about not using any company information improperly. As I said before, people are vulnerable, and the only real defense is removing

any critical security responsibility to the human. There are some many practices to adopt in order to control the most important type of treats:

- Robust email security solutions are actually the best option, also filtering any email containing the windows.net domain.
- Educate employees about recognizing different types of phishing attacks and avoid clicking any link.
- Use multi security layers, scanning email, antivirus, and use the red team to test malicious attack.
- Educate everybody in the companies, also the very top management.
- Use Multi-Factor Authentication in any sensitive location of the company.



## Summary

In this whitepaper, you learned the most important aspects of reconnaissance and scanning techniques. I described some of the most important concepts, and it is crucial to understand the strategies and the bases than the tools options itself. In the next chapters, we will use different scanning techniques, and in some cases, even social engineering and nastier example.

The main problem is the unpredictable way used by the corporation on deploying changes in any software. The UI may change, or a new security feature has been implemented, and we are not aware of that. Sometimes a new feature can be a good thing, and some other time could be a problem because we are not aware of that, and an employee may use it in a wrong way and create a vulnerability.

The number of new features deployed every day by Microsoft is not controllable, and the best practice is to use a clear, consolidated security strategy, avoid useless experimentations, and create our internal laws using policies.

In the next chapter, we will attack the most important asset, the Azure Network. We will learn the most important aspect of the network, and we will examine the most effective attacks and countermeasures.

# About the Author

**Write by Nino Crudele [Microsoft Azure MVP and Ethical Hacker]**



Nino Crudele is a freelance living in the United Kingdom. He is Global Azure Lead and Cybersecurity expert in Hexagon Manufacturing Intelligence, a global manufacturing company. He is responsible for leading the Microsoft Azure Cloud area, supporting and advising the Company to select the most appropriate cloud strategies and solutions from high-level design to implementation.

He is Microsoft Azure MVP since 2006 and Certified Ethical Hacker, Nino is also an international speaker, author, and a very active community member.

You can contact Nino at <mailto:mnino.crudele@live.com> (Twitter [@ninocrudele](https://twitter.com/ninocrudele))

LinkedIn: <https://www.linkedin.com/in/ninocrudele>



# About the Reviewers

## Sandro Pereira [Azure MVP & MCTS BizTalk Server 2010]



Sandro Pereira lives in Portugal and is currently working as an Integrator consultant at DevScope ([www.devscope.net](http://www.devscope.net)). In the past years, he has been working on implementing Integration scenarios both on-premises and cloud for various clients, each with different scenarios from a technical point of view, size, and criticality, using Microsoft Azure (API Management, Logic Apps, Service Bus, Event Hubs, PowerApps, Power Automate, ...), Microsoft BizTalk Server and different technologies like AS2, EDI, RosettaNet, SAP, TIBCO and so on.

Sandro is very active in the BizTalk community as blogger (<https://blog.sandropereira.com>), member and moderator on the MSDN BizTalk Server Forums, TechNet Wiki author, Code Gallery and GitHub contributor, member of several online communities, guest author at BizTalk360 and Serveless360, public speaker and technical reviewer of several BizTalk and Azure books and whitepapers, all focused on Integration. He is also the author of the book **BizTalk Mapping Patterns & Best Practices**.

He has been awarded the Microsoft Most Valuable Professional (MVP) since January 2011, for his contributions to the world-wide BizTalk Server community (<https://mvp.microsoft.com/en-us/PublicProfile/4030655>). He currently holds MCTS: BizTalk Server 2006 and BizTalk Server 2010 certifications.

You can contact Sandro at [sandro-pereira@live.com.pt](mailto:sandro-pereira@live.com.pt) (Twitter: [@sandro\\_asp](https://twitter.com/sandro_asp))