Get-AzurePasswords: Exporting Azure RunAs Certificates for Persistence

written by Karl Fosaaen | February 27, 2019

This post will be the first blog in a series that focuses around Azure Automation. I've recently run into a fair number of clients making use of Azure Automation Runbooks, and in many cases, the runbooks are being misconfigured. As attackers, these misconfigurations can provide us credentials, sensitive data, and some interesting points for escalation.

Here's the high level TLDR overview of a standard usage of the scenario outlined in this blog:

1. Gain access to an Azure AD account that has been provided runbook/Automation account permissions
2. Create and run a runbook that gathers available RunAs certificate files
3. Use the certificate files to maintain privileged access to an Azure environment
4. Automate the whole process to simplify persistence options

Intro to Automation Accounts

Automation accounts are Azure's way of handling subscription process automation. Most implementations that I have been exposed to are set up to generate reports, copy over logs, or generally manage Azure resources. In general, the runbooks that I have seen are fairly basic and act as scheduled jobs to handle basic tasks. That being said, Automation runbooks can handle many different modules to allow you to administer multiple aspects of an Azure subscription.

Before we get too far, there's a couple of base terms to

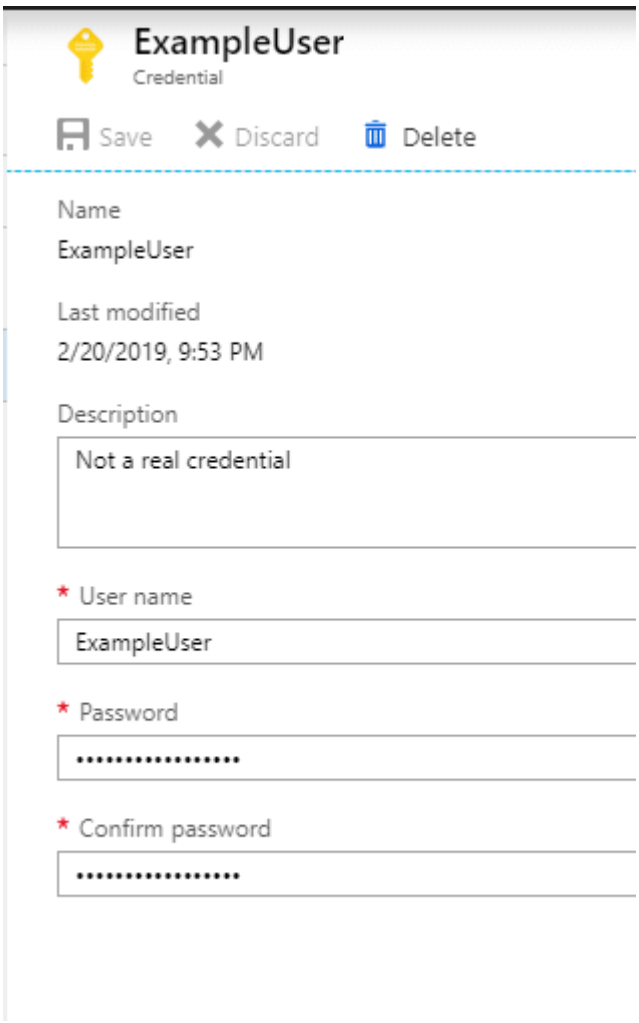
cover:

- Automation Account
  - This is how Azure defines the high level organization unit for runbooks
  - Automation accounts are the container for runbooks
  - Subscription inherited (Owner/Contributor/Reader) IAM/Access control roles are applied at this level
- Runbook
  - This is the specific code/script/workflow that you would run under an automation account
  - Typically PowerShell, Python, or a Workflow
  - These can also contain sensitive data, so looking over the code can be useful
- Automation credential
  - These are the credentials used by the Runbook to accomplish tasks
  - Frequently set up as a privileged (contributor) account
  - There are two types of credentials
    - Traditional username/password
    - RunAs certificates

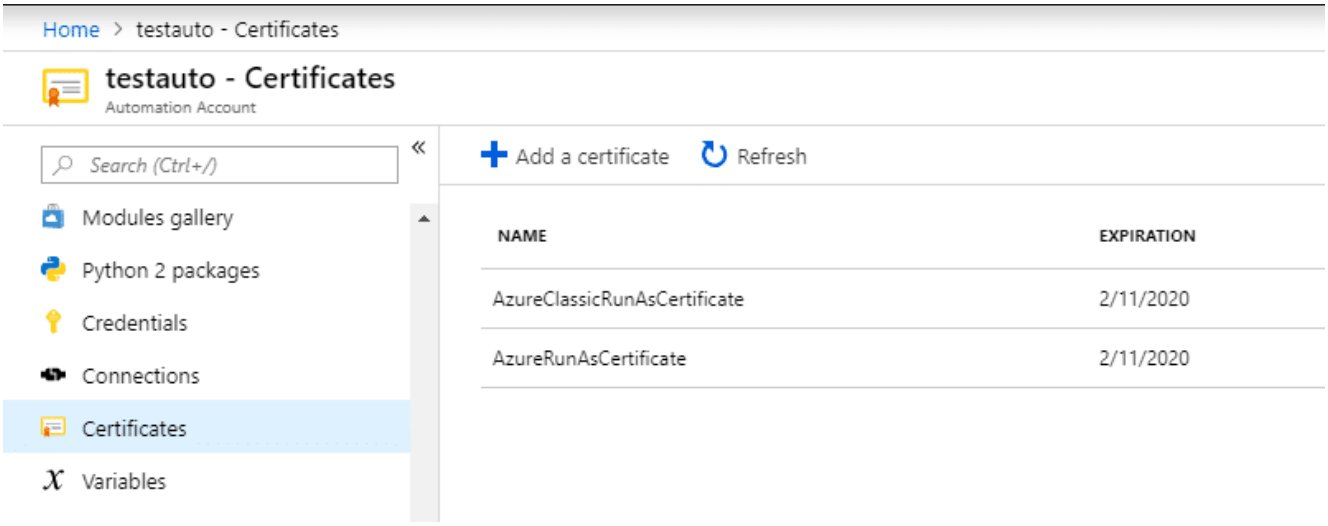
Both types of credentials (Passwords/RunAs) can typically be used to get access to subscriptions via the Azure PowerShell cmdlets.

Passwords versus Certificates

Automation Credentials are a cleartext username and password that you can call within a runbook for authentication. So if you're able to create a runbook, you can export these from the Automation account. If these are domain credentials, you should be able to use them to authenticate to the subscription. I already covered the cleartext automation credentials in a previous post, so feel free to read up more about those there.



Surprisingly, I have found several instances where the automation credential account is set up with Global Administrator permissions for the Azure tenant, and that can lead to some interesting escalation opportunities.



The primary (and more common) option for running Runbooks is with a RunAs certificate. When creating a RunAs certificate, a

RunAs account is registered as a Service Principal in Azure AD for the certificate/account. This "App" account is then added to the Contributors group for the domain. While some Azure admins may lock down the permissions for the account, we typically see these accounts as an opportunity to escalate privileges in an Azure environment.

If a "regular", lesser privileged user, has rights to create/run/modify Runbooks, they inherently have Contributor level rights for the subscription. While this is less likely, there is a chance that a user may have been granted contributor rights on an automation account. Check out this post by Microsoft for a full rundown of Automation Account Role Based Access Controls. At a high level, the subscription Owner and Contributor roles will have full access to Automation accounts.

Regardless of the privilege escalation element, we may also want to maintain access to an Azure environment after our initial account access is removed. We can do that by exporting RunAs certificates from the Automation account. These certificates can be used later to regain access to the Azure subscription.

Exporting Certificates

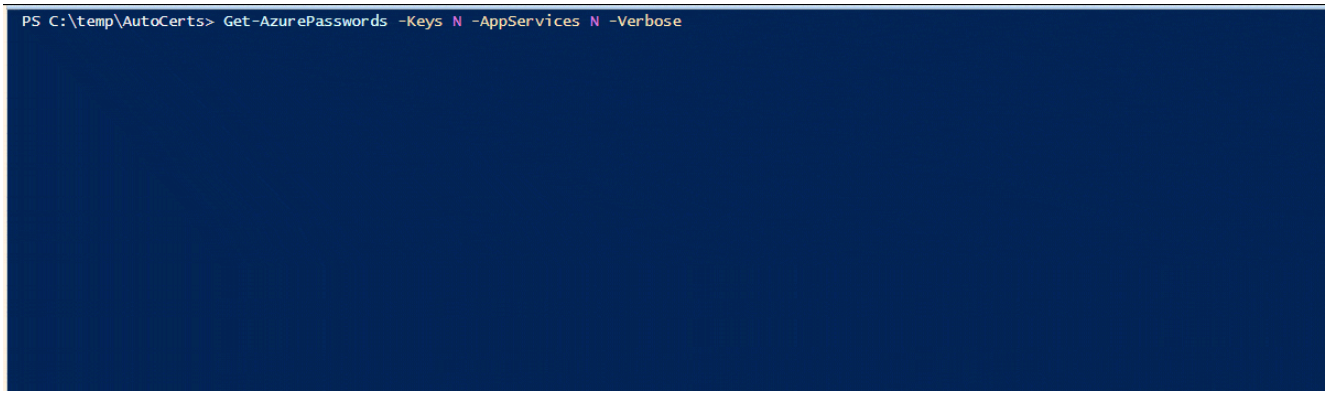
You will need to start by grabbing a copy of MicroBurst from the NetSPI Github.

The script (Get-AzurePasswords) was previously released, and already had the ability to dump the cleartext passwords for Automation accounts. If that is not an option for an Automation account that you're attacking, you now have the functionality to export a PFX file for authentication.

The script will run an Automation Runbook on the subscription that exports the PFX file and writes it out to a local file. Once that is complete, you should have a ps1 file that you can

use to automate the login for that RunAs account. I initially had this script pivoting the PFX files out through a Storage Account container, but found an easier way to just cast the files as Base64, and export them through the job output.

Example Command: Get-AzurePasswords -Keys N -AppServices N -Verbose



Using the Authentication

The initial need for collecting automation certificates came from a recent assessment where I was able to pull down cleartext credentials for an automation account, but multi-factor authentication got in the way of actually using the credentials. In order to maintain contributor level access on the subscription, I needed to gather Automation certificates. If the initial access account was burned, I would still have multiple contributor accounts that I could fall back on to maintain access to the Azure subscription.

It’s important to make a note about permissions here. Contributor access on the subscription inherently gives you contributor rights on all of the virtual machines.

As mentioned in a previous post, **Contributor rights on Virtual Machines grants you SYSTEM on Windows and root on Linux VMs**. This is really handy for persistence.

When running the AuthenticateAs scripts, make sure that you are running as a local Administrator. You will need the rights

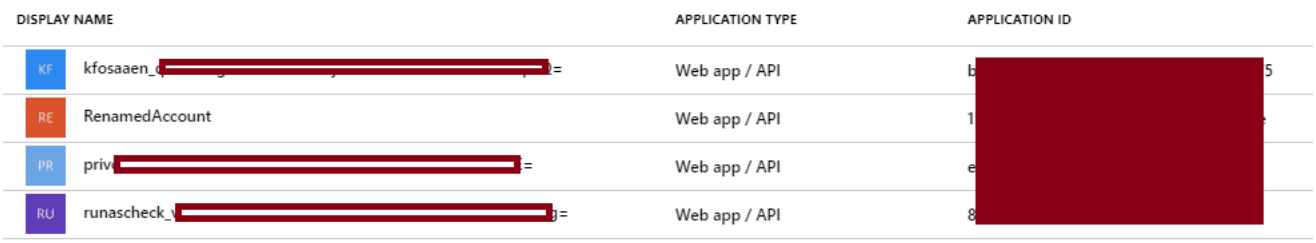
to import the PFX file.

It’s not very flashy, but here is a GIF that shows what the authentication process looks like.

One important note about the demo GIF:

I purposefully ran the Get-AzureRmADUser command to show that the RunAs account does not have access to read AzureAD. On the plus side, that RunAs account does have contributor rights on all of the virtual machines. So at a minimum, we can maintain access to the Azure VMs until the certificate is revoked or expires.

In most cases, you should be able to run the “AuthenticateAs” ps1 scripts as they were exported. If the automation RunAs Service Principal account (Example: USER123_V2h5IGFyZSB5b3UgcmVhZGluZyB0aGlzPw==) has been renamed (Example: User123), the “AuthenticateAs” ps1 script will most likely not work. If that is the case, you will need to grab the Application ID for the Automation account and add it to the script. You may have to search/filter a little to find it, but this ID can be found with the following AzureRM command: Get-AzureRmADServicePrincipal | select DisplayName,ApplicationId



The script will prompt you if the account has been renamed, so keep an eye out for the warning text.

The AppIDs are also captured in the Domain_SPNs.csv (in the Resources folder) output from the Get-AzureDomainInfo function in MicroBurst.

Next Steps

Since I've been seeing a mix of standard Automation credentials and RunAs certificates lately, I wanted to have a way to export both to help maintain a persistent connection to Azure during an assessment. In the next two blogs, I'll cover how we can make use of the access to get more sensitive information from the subscription, and potentially escalate our privileges.