Once we have a solid list of subdomains, we can map these back to there respective cloud services.

If you have not previously completed the **"Discovery and Recon"** lab, which should create a file with approximately 111 lines (cat /shared/all_subdomains.txt | wc -l) called /shared/all_subdomains.txt, please run the following command to download a known good copy of the file before preceding with this lab:

```
curl -o /shared/all_subdomains.txt https://public-astute-cloud-20200813-935672326788.s3.amazonaws.com/all_subdomains.txt
```

IP Lookup

Many cloud service providers (e.g. AWS, Azure, etc...) publicly list the IP addresses that are associated with their cloud services. For example, you can find the IP addresses associated with the following Cloud providers at the pages located below:

- Amazon AWS -> https://ip-ranges.amazonaws.com/ip-ranges.json
- Microsoft Azure -> https://www.microsoft.com/en-us/download/details.aspx?id=41653\
  - Azure is also now in JSON:
    - https://www.microsoft.com/en-us/download/details.aspx?id=56519
    - https://download.microsoft.com/download/7/1/D/71D86715-5596-4529-9B13-DA13A5DE5B63/ServiceTags_Public_20200727.json
- Google Cloud Compute Engine (GCE) -> https://cloud.google.com/compute/docs/faq#ipranges

We can use the nimbusland tool to query to see what cloud provider's are associated with a specific IP address:

```
root@ip-10-0-1-215:/shared# ping -c3 lizardblue.com
PING lizardblue.com (18.218.14.179) 56(84) bytes of data.

--- lizardblue.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2014ms

root@ip-10-0-1-215:/shared# cnoio_nimbusland 34.209.82.230
CRITICAL:root:

\ | _) | | |
\ | | __ `__ \ __ \ | | __| | _` | __ \ _` |
|\ | | | | | | | | | \__ \ | ( | | | ( |
_| \_| _| _| _| _| _.__/ \__,_| ____/ _| \__,_| _| _| \__,_|



CRITICAL:root:[+] Nimbusland - Alpha v0.0.6
CRITICAL:root:[-] sStartIp: 34.209.82.230
CRITICAL:root:[-] Checking AWS Network Ranges
CRITICAL:root:[+] Match Found in AWS! 34.209.82.230, 34.208.0.0/12, us-west-2, Aws, EC2
CRITICAL:root:[-] Checking Azure Network Ranges
```

Region Lookup

Depending on the AWS and/or Azure Service, the IP lookup may also tell use which regions are in use by the target organization.

In addition a few other basic region techniques sometimes reveal the region in use by services.

Region information can be very helpful once we have obtain some secrets and then wish to leverage them to collect more information about the target in the most stealthy manor possible.

Ping

We can sometimes easily discover the region a bucket is located within using the ping tool:

```
root@ip-10-0-1-215:/shared# ping -c3 cdn2.lizardblue.com
PING s3-w.us-east-2.amazonaws.com (52.219.100.28) 56(84) bytes of data.
64 bytes from s3-w.us-east-2.amazonaws.com (52.219.100.28): icmp_seq=1 ttl=58 time=0.985 ms
64 bytes from s3-w.us-east-2.amazonaws.com (52.219.100.28): icmp_seq=2 ttl=58 time=1.12 ms
64 bytes from s3-w.us-east-2.amazonaws.com (52.219.100.28): icmp_seq=3 ttl=58 time=1.11 ms

--- s3-w.us-east-2.amazonaws.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.985/1.076/1.127/0.070 ms
```

You can see in this output that the bucket is located within the "us-east-2" region of AWS.

Dig

We can sometimes easily discover the region a bucket is located within using the dig tool:

```
root@ip-10-0-1-215:/shared# dig cdn2.lizardblue.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> cdn2.lizardblue.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38038
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;cdn2.lizardblue.com. IN A

;; ANSWER SECTION:
cdn2.lizardblue.com. 26 IN CNAME cdn2.lizardblue.com.s3.amazonaws.com.
cdn2.lizardblue.com.s3.amazonaws.com. 60 IN CNAME s3-w.us-east-2.amazonaws.com.
s3-w.us-east-2.amazonaws.com. 1 IN A 52.219.80.196

;; Query time: 14 msec
;; SERVER: 10.0.0.2#53(10.0.0.2)
;; WHEN: Sun Jul 08 15:28:41 UTC 2018
;; MSG SIZE rcvd: 140
```

You can see in this output that the bucket is located within the "us-east-2" region of AWS.

We can also check for reverse DNS entries using the "-x" switch with dig:

```
root@ip-10-0-1-215:/shared# dig -x 52.219.88.26

; <<>> DiG 9.10.3-P4-Ubuntu <<>> -x 52.219.88.26
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35665
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;26.88.219.52.in-addr.arpa. IN PTR

;; AUTHORITY SECTION:
88.219.52.in-addr.arpa. 60 IN SOA dns-external-master.amazon.com. root.amazon.com. 3 3600 900 604800 900

;; Query time: 18 msec
;; SERVER: 10.0.0.2#53(10.0.0.2)
;; WHEN: Sun Jul 29 03:33:26 UTC 2018
;; MSG SIZE rcvd: 125
```

Where we can see that the IP is associated with AWS.

Nslookup

We can sometimes easily discover the region a bucket is located within using the nslookup tool:

```
root@ip-10-0-1-215:/shared# nslookup cdn2.lizardblue.com
Server: 172.31.0.2
Address: 172.31.0.2#53

Non-authoritative answer:
cdn2.lizardblue.com canonical name = cdn2.lizardblue.com.s3.amazonaws.com.
cdn2.lizardblue.com.s3.amazonaws.com canonical name = s3-w.us-east-2.amazonaws.com.
Name: s3-w.us-east-2.amazonaws.com
Address: 52.219.88.228
```

You can see in this output that the bucket is located within the "us-east-2" region of AWS.

Automation

We can create basic scripts to help collect information from our subdomain list, for example the following script will perform a nslookup on each domain name and then we can grep through the results for regions and services names of interest:

```
root@ip-10-0-1-215:/shared# cd /shared/

root@ip-10-0-1-215:/shared# cat /shared/lookups/nslookups.sh
#!/bin/bash
#
# $1 is the fist argument passed to the script, the file containing one ip per line to use as input (e.g. all_subdomains.txt)
#
# $2 is the second argument passed to the script, the file to output results into (e.g. domain_lookups.txt)
#
echo '[+] Reading one IP address per line from:' $1
echo '[+] Writing the nslookup results to this file:' $2
while read lineinfile; do
echo '[+] Performing nslookup for:' $lineinfile
echo '[+] Performing nslookup for:' $lineinfile >> $2
nslookup $lineinfile >> $2
done < $1

root@ip-10-0-1-215:/shared# /shared/lookups/nslookups.sh /shared/all_subdomains.txt /shared/results_nslookups.txt

root@ip-10-0-1-215:/shared# tail /shared/results_nslookups.txt
Address: 52.219.80.40

[+] Performing nslookup for: www.lizardblue.com
Server: 10.0.0.2
Address: 10.0.0.2#53

Non-authoritative answer:
Name: www.lizardblue.com
Address: 18.222.129.248

root@ip-10-0-1-215:/shared# grep "s3" /shared/results_nslookups.txt
cdn2.lizardblue.com canonical name = cdn2.lizardblue.com.s3.amazonaws.com.
cdn2.lizardblue.com.s3.amazonaws.com canonical name = s3-w.us-east-2.amazonaws.com.
Name: s3-w.us-east-2.amazonaws.com
...

root@ip-10-0-1-215:/shared# grep "us-" /shared/results_nslookups.txt
...
staff.lizardblue.com canonical name = soundslike.lizardblue.com.s3.us-east-2.amazonaws.com.
soundslike.lizardblue.com.s3.us-east-2.amazonaws.com canonical name = s3-r-w.us-east-2.amazonaws.com.
Name: s3-r-w.us-east-2.amazonaws.com
```

Exercise

The Plan:

Core Ops:

- Resolve subdomains to IP addresses

-- Discover what cloud providers the domains / IP addresses are using.

-- Discover what regions are in use.

- Leverage the bash script to perform DNS lookups on domain names

-- Find which sub-domains point to S3 buckets.

Above & Beyond:

- Improve the automation via enhanced collection and parsing of recon information.

Subdomain Takeover

When an organization ceases to use a cloud service but still has references to the service (e.g. DNS entries, links within websites, etc...), often times an attacker can register themselves for the services and then start masquerading as the targeted organization.

Management of an organizations DNS entries is frequently poorly executed due to various reasons. For example, within a larger organization you may have multiple teams each managing their domain names with their own unique processes and information systems. Individuals who manage solutions may fail to coordinate with the correct team when services are no longer in use, to ensure DNS entries for subdomains are still all point to the correct locations and that no extra subdomains still exist for their specific solution.

With the advent of Cloud services, individuals are able to quickly create and destroy information systems, often in a uncontrolled and/or unified manor. This is sometimes referred to as "Cloud Sprawl" and typically occurs when an organization lacks proper visibility and/or control over it's cloud computing resources.

We can leverage the information we discovered previously and test to see if we can find any subdomains still point to now unused cloud services using the "subjack" tool. This scenario where subdomain names are still pointing to cloud services which are now no longer in use is frequently referred to as a "subdomain takeover", as the attacker can frequently create an account with the service where the subdomain is still pointing to and then start leveraging the target organization's subdomain for whatever purpose they see fit.

We can test the subdomains which we previously discovered. One of these subdomains was still pointing to on-demand IT services which are no longer in use by the target organization and hence are available for another malicious user to register and then subsequently use to masquerade as the target organization to unsuspecting users.

Subjack is a tool that searches through a list of subdomains, found via Amass or Gobuster or another tool, and returns which subdomains are currently pointing to a cloud service which we may be able to hijack.

We can check out the help for the Subjack tool using the following command:

```
root@ip-10-0-1-215:/shared# cnoio_subjack -h
Usage of /root/go/bin/subjack:
-a Find those hidden gems by sending requests to every URL. (Default: Requests are only sent to URLs with identified CNAMEs).
-o string
Output results to file.
-ssl
Force HTTPS connections (May increase accuracy (Default: http://).
-t int
Number of concurrent threads (Default: 10). (default 10)
-timeout int
Seconds to wait before connection timeout (Default: 10). (default 10)
-v Display more information per each request.
-w string
Path to wordlist.
```

We can use Subjack with our list of subdomains like this to find all the potentially vulnerable ones:

```
root@ip-10-0-1-215:/shared# cnoio_subjack -w /shared/all_subdomains.txt -o /shared/subjack_results.txt
[S3 BUCKET] staff.lizardblue.com

root@ip-10-0-1-215:/shared# cat /shared/subjack_results.txt
[S3 BUCKET] staff.lizardblue.com
```

We can then perform an nslookup on the vulnerable domain name to frequently discover the exact s3 bucket name and region we need to use as part of our subdomain takeover:

```
root@ip-10-0-1-215:/shared# nslookup staff.lizardblue.com
Server: 172.31.0.2
Address: 172.31.0.2#53

Non-authoritative answer:
staff.lizardblue.com canonical name = soundslike.lizardblue.com.s3.us-east-2.amazonaws.com.
soundslike.lizardblue.com.s3.us-east-2.amazonaws.com canonical name = s3-r-w.us-east-2.amazonaws.com.
Name: s3-r-w.us-east-2.amazonaws.com
Address: 52.219.100.88
```

From this output we can see that the region for the bucket is "us-east-2", aka "US East (Ohio)", and that the name of the s3 bucket we need to create to takeover this subdomain is "soundslike.lizardblue.com".

Completing the Takeover

The above commands will show you all of the s3 buckets vulnerable to a takeover, for example we should see output similar to the following...

```
root@ip-10-0-1-36:/shared# cat /shared/subjack_results.txt
...
[S3 BUCKET] lizardbluereports0017.lizardblue.com
[S3 BUCKET] lizardbluereports0018.lizardblue.com
[S3 BUCKET] lizardbluereports0019.lizardblue.com
[S3 BUCKET] lizardbluereports001.lizardblue.com

[S3 BUCKET] lizardbluereports0020.lizardblue.com

[S3 BUCKET] lizardbluereports0021.lizardblue.com
[S3 BUCKET] lizardbluereports0022.lizardblue.com
[S3 BUCKET] lizardbluereports0023.lizardblue.com
...
```

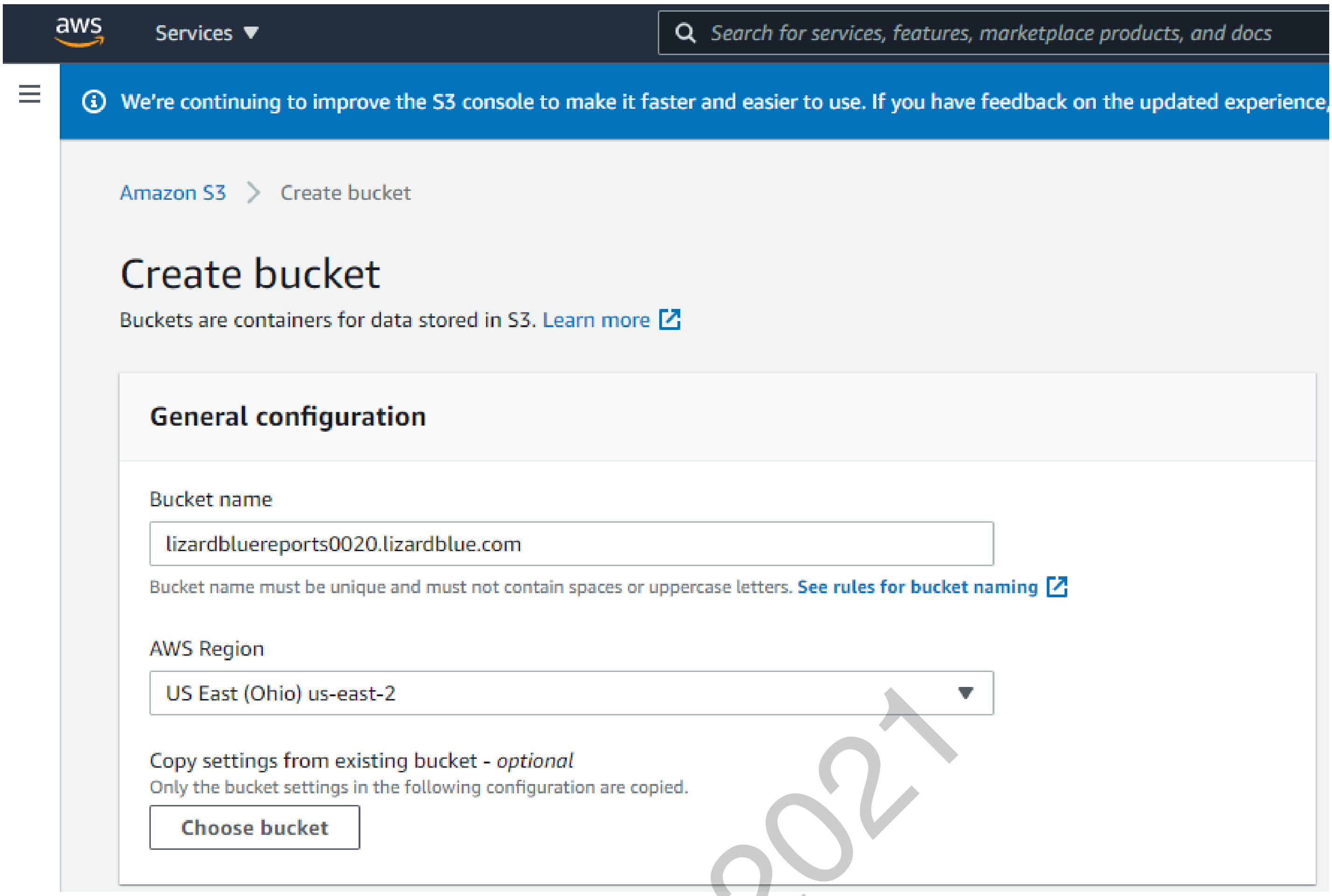
Let's pick the bucket that correlates with our student number (e.g. 055) and take it over!

Browse to the S3 service within the AWS console: https://s3.console.aws.amazon.com/s3/home?region=us-east-2#

Click the "+ Create bucket" Button

Create a bucket with the following settings:

- Bucket name: lizardbluereports0###.lizardblue.com
    - NOTE: ### is your student number, 020 in this example
- Region: Ohio


aws
Services ▼
Search for services, features, marketplace products, and docs

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience,

Amazon S3 > Create bucket

Create bucket

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name

lizardbluereports0020.lizardblue.com

Bucket name must be unique and must not contain spaces or uppercase letters. **See rules for bucket naming**

AWS Region

US East (Ohio) us-east-2 ▼

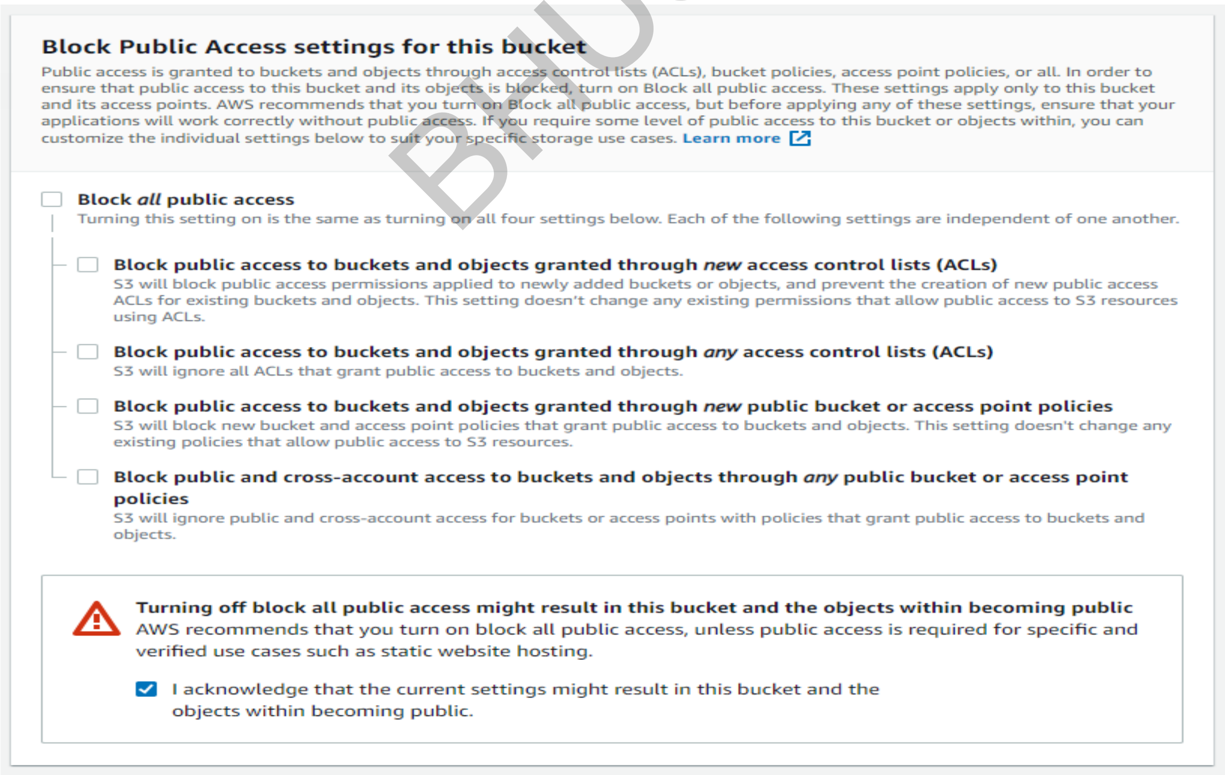
Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Scroll down

Uncheck the box for "block all public access"...

And check the box for "I acknowledge that the current settings might result in this bucket and the objects within becoming public."



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. **Learn more**

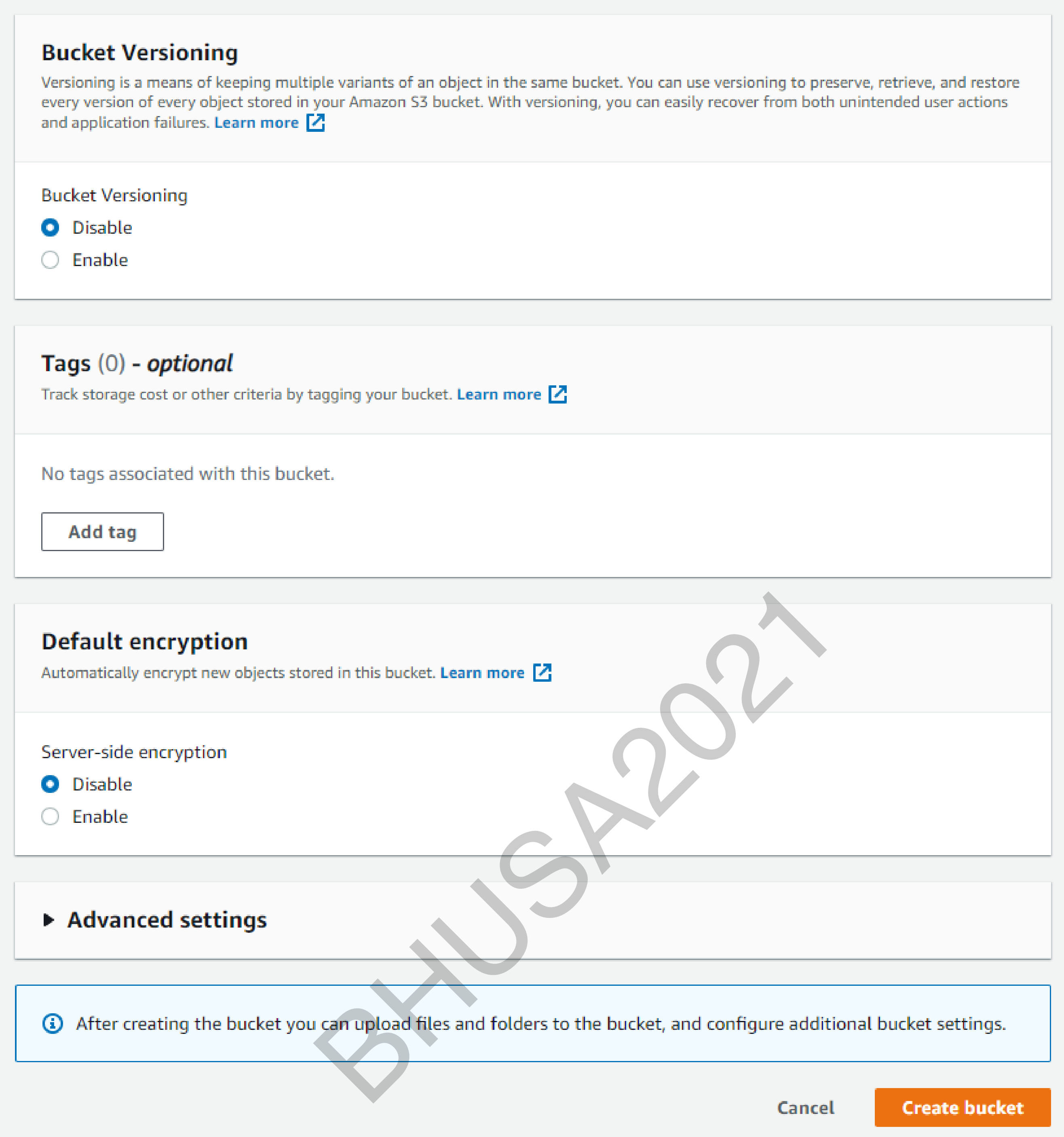
□ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- □ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- □ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
  S3 will ignore all ACLs that grant public access to buckets and objects.
- □ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- □ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☑ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Scroll down to the bottom of the page



Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

● Disable

○ Enable

Tags (0) - *optional*

Track storage cost or other criteria by tagging your bucket. Learn more

No tags associated with this bucket.

Default encryption

Automatically encrypt new objects stored in this bucket. Learn more

Server-side encryption

● Disable

○ Enable

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Click the "Create bucket" Button

Now re-run subjack...

```
root@ip-10-0-1-215:/shared# cnoio_subjack -w /shared/all_subdomains.txt -o /shared/subjack_results_002.txt
...
```

And then look for your bucket in the output:

```
root@ip-10-0-1-215:/shared# cat /shared/subjack_results_002.txt | sort -u > /shared/subjack_results_002_sort_u.txt


root@ip-10-0-1-215:/shared# cat /shared/subjack_results_002_sort_u.txt
...
[S3 BUCKET] lizardbluereports0017.lizardblue.com
[S3 BUCKET] lizardbluereports0018.lizardblue.com
[S3 BUCKET] lizardbluereports0019.lizardblue.com
[S3 BUCKET] lizardbluereports001.lizardblue.com
[S3 BUCKET] lizardbluereports0021.lizardblue.com
[S3 BUCKET] lizardbluereports0022.lizardblue.com
[S3 BUCKET] lizardbluereports0023.lizardblue.com
...
```

You should no longer see your S3 bucket in the output because it is now registered... by the attacker!

```
root@ip-10-0-1-36:/shared# cat /shared/subjack_results_002.txt | grep "019"
[S3 BUCKET] lizardbluereports0019.lizardblue.com

root@ip-10-0-1-36:/shared# cat /shared/subjack_results_002.txt | grep "020"

root@ip-10-0-1-36:/shared# cat /shared/subjack_results_002.txt | grep "021"
[S3 BUCKET] lizardbluereports0021.lizardblue.com
```

Exercise

The Plan:

```
- Leverage subjack to find which of these subdomains is vulnerable to a takeover.

-- Create an S3 bucket within your Student AWS account that will takeover the targeted s3 bucket.

--- Re-run subjack to see that the S3 bucket no longer appears vulnerable, because you have now registered the S3 bucket.
```

References:

Check out the following references for more information:

- explainshell - https://explainshell.com/
- subjack - https://github.com/haccer/subjack
- Subdomain takeover due to unclaimed Amazon S3 bucket - https://hackerone.com/reports/121461
- tko-subs - https://github.com/anshumanbh/tko-subs
- HostileSubBruteforcer - https://github.com/nahamsec/HostileSubBruteforcer
- autoSubTakeover - https://github.com/JordyZomer/autoSubTakeover
- explainshell - https://explainshell.com/
- Can I take over XYZ? - https://github.com/EdOverflow/can-i-take-over-xyz