M346, AWS CLI, Phishing-Server.

Thema:
Infrastructure as Code

Lernziele
- Sie automatisieren die Provisionierung eines kompletten Phishing-Servers mit Hilfe eines Scripts und der AWS CLI.
- Sie verwenden den AWS Secrets Manager zu Speicherung eins Passworts.

Sozialform
Einzelarbeit oder Arbeit im Zweierteam

Umgebung
Diese Übung ist für das **AWS LearnerLab** angelegt. Wird die Übung auf einer anderen Umgebung ausgeführt, muss zusätzlich noch ein LabInstanceProfile erzeugt werden.

Warnhinweis:
SocialFish ist eine Awareness-Tool und darf nur zur Sensibilisierung verwendet werden!
Die Verwendung zum Stehlen von Passwörtern ohne vorherige Zustimmung ist illegal und strafbar!

Aufgabe 1: Provisionierung des Phishing-Server mit Hilfe von CLI-Commands

Erstellen Sie ein Schlüsselpaar, das im Verzeichnis ~/.ssh abgelegt wird (Siehe Schlüsselpaar erstellen):

```
aws ec2 create-key-pair \
--key-name aws-gbs-cli \
--key-type rsa \
--query 'KeyMaterial' \
--output text > ~/.ssh/aws-gbs-cli.pem
```

Erstellen Sie eine neue Sicherheitsgruppe, die eingehende Requests an Port 22 für SSH und Port 5000 für TCP zulässt (Siehe Sicherheitsgruppe erstellen):

```
aws ec2 create-security-group \
--group-name gbs-sec-group \
--description "Phishing-Server"

aws ec2 authorize-security-group-ingress \
--group-name gbs-sec-group \
--protocol tcp \
--port 5000 \
--cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress \
--group-name gbs-sec-group \
--protocol tcp \
--port 22 \
--cidr 0.0.0.0/0
```

Damit das Passwort des Phishing-Servers nicht im Init-Script gespeichert werden muss, legen Sie das Phishing Server Passwort im Secret-Manager ab (Siehe aws secretmanager create-secret):

```
aws secretsmanager create-secret \
--name PhishingServerPwd \
--secret-string myPassword
```

Für den ersten Start einer EC2-Instanz kann in *user-data* ein Initial-Skript konfiguriert werden.

Erstellen Sie im aktuellen Verzeichnis eine Datei *aws-phishing-server-init.txt* mit folgendem Inhalt:

```
#!/bin/bash
sudo apt-get update
sudo apt-get -y install python3-pip
sudo apt-get -y install nmap
sudo apt-get -y install awscli
pip install PyLaTeX==1.4.1
pip install python3-nmap==1.6.0
pip install qrcode==7.4.2
pip install Werkzeug==2.3.7
pip install Flask==2.3.3
pip install Flask_Login==0.6.2
pip install python-secrets==23.4.2
pip install python-nmap==0.7.1
git clone https://github.com/UndeadSec/SocialFish.git
cd SocialFish
chmod +x SocialFish.py

PWD=$(aws secretsmanager get-secret-value \
--region us-east-1 \
--secret-id PhishingServerPwd \
--query SecretString \
--output text)

./SocialFish.py gbs $PWD
```

Nun ist alles bereit für die Erzeugung der EC2-Instanz (Siehe aws ec2 run-instances):

```
EC2_INSTANCE_ID=$(aws ec2 run-instances \
--image-id ami-08c40ec9ead489470 \
--count 1 \
--instance-type t2.micro \
--key-name aws-gbs-cli \
--security-groups gbs-sec-group \
--iam-instance-profile Name=LabInstanceProfile \
--user-data file://aws-phishing-server-init.txt \
--query 'Instances[*].InstanceId' \
--output text)
```

Anhand der InstanceId in Variable EC2_INSTANCE_ID wird die PublicId ermittelt (Siehe aws ec2 describe-instances) und mit dem echo-Befehl ausgegeben:

```
EC2_PUBLIC_IP=$(aws ec2 describe-instances \
--filters Name=instance-id,Values=$EC2_INSTANCE_ID \
--query 'Reservations[*].Instances[*].[PublicIpAddress]' \
--output text)

echo "Server-URL: http://$EC2_PUBLIC_IP:5000/neptune"
```

Hinweis: Die Initialisierung des Servers dauert 3-4 Minuten.

Das Einrichten und Durchführen einer Phishing-Demo ist in *A346-09-2-Sicherheit-Phishing-Demo* beschrieben.

Hinweis: Die Initialisierung der EC2-Instanz wird in Datei /var/log/cloud-init-output.log protokolliert. Fehler im Initialisierungs-Skript können mit folgendem Befehl in einer Bash der EC2-Instanz angezeigt werden: `cat /var/log/cloud-init-output.log`

Aufgabe 2: Phishing Server und Ressourcen löschen

Sie die aktive Instanz. Dazu wird die Instance-Id benötigt.

```
aws ec2 terminate-instances \
--instance-ids [instande-id]
```

Die verbleibenden Ressourcen löschen Sie mit folgenden Befehlen.:

```
aws secretsmanager delete-secret \
--secret-id PhishingServerPwd \
--force-delete-without-recovery
```

```
aws ec2 delete-security-group \
--group-name gbs-sec-group
```

```
aws ec2 delete-key-pair \
--key-name aws-gbs-cli
```

```
rm ~/.ssh/aws-gbs-cli.pem
```

Hinweis: Die Lösch-Anweisungen werden asynchron ausgeführt. Falls eine Ressource nicht gelöscht werden kann (Fehlermeldung), warten Sie einen Moment und versuchen es erneut.

Aufgabe 3: Script erstellen

Erstellen Sie nun eine Script-Datei, die alle vorher einzeln ausgeführten Kommandos als Batch ausführt.

Tipps:

- Geben Sie mit *echo* ausführliche Statusinformationen aus. Das hilft Ihnen einen allfälligen Fehler einfacher zu lokalisieren.
- Prüfen Sie, ob Ressourcen wie z.B. Schlüsselpaar, Sicherheitsgruppe, etc. bereits vorhanden sind. Erstellen Sie sie nur, wenn sie noch nicht vorhanden sind.
- Prüfen Sie am Ende des Scripts alle paar Sekunden, ob der Server bereit ist (HTTP-Status 200) und geben Sie eine Meldung aus, sobald der Server verwendet werden kann.