Guide for Creating a Moving Target Defense Experimental Demo

Setting Up AWS VPC

- Create VPC
  - On AWS, go to VPC on Management Console
  - Click on *Your VPCs*
  - Click on *Create VPC*
    - For *Name tag* field enter “MTD”
    - For *IPv4 CIDR Block* enter “10.0.0.0/16”
    - Click *Create*
  - Click *Close*
- Create Subnet
  - On the VPC Management Console, click on *Subnets*
  - Click on *Create subnet*
    - For *Name tag* field enter “MTD subnet1”
    - For *VPC* field select “MTD”
    - For *IPv4 CIDR block* enter “10.0.0.0/24”
    - Click *Create*
  - Click *Close*
- Create Certificates
  - Make sure AWS CLI is installed on your local machine
  - Run the following commands to build and import certificates (go to https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/authentication-authorization.html for more info on this step)

```
git clone https://github.com/OpenVPN/easy-rsa.git
cd easy-rsa/easyrsa3
./easyrsa init-pki
./easyrsa build-ca nopass
./easyrsa build-server-full server nopass
./easyrsa build-client-full client1.domain.tld nopass
mkdir ~/custom_folder/
cp pki/ca.crt ~/aws_certs/
cp pki/issued/server.crt ~/aws_certs/
cp pki/private/server.key ~/aws_certs/
cp pki/issued/client1.domain.tld.crt ~/aws_certs
cp pki/private/client1.domain.tld.key ~/aws_certs/
cd ~/aws_certs/
aws acm import-certificate --certificate file://server.crt --private-key
file://server.key --certificate-chain file://ca.crt --region us-east-2
aws acm import-certificate --certificate file://client1.domain.tld.crt --private-key
file://client1.domain.tld.key --certificate-chain file://ca.crt --region us-east-2
```

- Create Client VPN Endpoint
  - On AWS, go to VPC on Management Console
  - Click on *Client VPN Endpoints*
  - Click on *Create Client VPN Endpoint*

        - For *Client IPv4 CIDR* field enter “192.168.0.0/16”
        - For *Server certificate ARN* select the server certificate that you created earlier
        - For *Authentication Options* field select “Use mutual authentication”
        - For *Client certificate ARN* field select the client certificate that you created earlier
        - For *Connection Logging* select “No”
        - Click *Create Client VPN Endpoint*
    - Click *Close*
- Create Internet Gateway
    - On AWS, got to VPC on Management Console
    - Click on *Internet Gateways*
    - Click on *Create internet gateway*
        - For *Name tag* field enter “MTD internet gateway”
        - Click *Create*
    - Click *Close*
    - Select your newly created internet gateway, and click *Actions > Attach to VPC*
        - For *VPC* field select “MTD”
        - Click *Attach*
    - Click *Close*

- Set Up Tunnelblick
    - Download Tunnelblick VPN client at https://tunnelblick.net/downloads.html
    - On AWS, go to VPC on Management Console, and click on *Client VPN Endpoints*
    - Select your Client VPN Endpoint, and click *Download Client Configuration*
        - Click *Download*
    - Ensure that your config file is in the same folder as your client cert file and your client key file, and rename the folder “mtd-vpn.tblk”
    - Click on the “mtd-vpn.tblk” file to upload it to Tunnelblick

Building Kali Linux Instance

- On AWS, go to EC2 on Management Console
- Click on *Launch Instance*
- Click on *AWS Marketplace*
- Select Kali Linux Image, then click *Continue*
- Select type of “t2.micro”, then click *Review and Launch*
- Click *Edit Security Groups*
    - Click *Add Rule*
    - For Type choose “All traffic”
    - For CIDR enter “0.0.0.0/0”
    - Click *Review and Launch*
- Click *Edit* Instance Details
    - Change Number of Instances field to however many instances you want
    - Change Network to your VPC that you created

    - Click *Review and Launch*
- Click *Launch*
- Select “Create a new key pair” from dropdown
- For Key pair name field enter “kali-linux-key”
- Click *Download Key Pair,* and make sure that you save the “kali-linux-key.pem” file somewhere that you remember
- Click *Launch Instances*
- On the EC2 Management Console, select the security group for the instance that you just created
- Click on the Security group ID
- Click on *Edit inbound rules*
- Click on *Add Rule*
- For Type choose “All traffic”
- For Source enter “0.0.0.0/0”
- Click *Save Rules*

Building Metasploitable3 Instance

- Build Metasploitable3 Instance on VirtualBox
    - Make sure that VirtualBox is installed on your machine
    - Make sure that vagrant is installed on your machine
- Run following commands (go to https://github.com/jocic/AWS.Metasploitable3 for more info on this step)

```
mkdir metasploitable3-workspace
cd metasploitable3-workspace
curl -O https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile &&
vagrant up
```

- It takes a while, but once the instance “Metasploitable3-ub1404” has been created on VirtualBox, click on the instance, and then click *Start*
- Log in to the instance using username “vagrant” and password “vagrant”
- Pre-install necessary items using following commands

```
git clone https://github.com/MovingTargetDefenseCapsone/semester1.git
cd semester1
./set_up_migration.sh
```

    - When prompted, select “apache2”
    - When prompted, enter “Y”
    - When prompted, select “No’
- Power off the instance
- Export the VM from VirtualBox
    - Select the instance in VirtualBox
    - Click on *File* > *Export Appliance*
    - In the window that appears, select *Continue* > *Continue* > *Export*

- VM Import (go to https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html for more info on this step)
    - On AWS, go to S3 on the Management Console
    - Click on *Create Bucket*
        - Name the bucket “mtdbucket”
        - Click on *Create Bucket*
    - Select your newly created S3 bucket
    - Click on *Upload*
        - Click on *Add Files*
        - Locate your “Metasploitable3-ub1404.ova” image that you exported earlier, select the file, and Click *Open*
        - Click *Upload* and wait for file to upload
    - Create new file on your local machine called “trust-policy.json”

```
{
   "Version": "2020-04-01",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": { "Service": "vmie.amazonaws.com" },
         "Action": "sts:AssumeRole",
         "Condition": {
            "StringEquals":{
               "sts:Externalid": "vmimport"
            }
         }
      }
   ]
}
```

- Create new file on your local machine called “role-policy.json”

```
{
   "Version":"2020-04-01",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::mybucket",
            "arn:aws:s3:::mybucket/*"
         ]
      },
```

```
      {
         "Effect":"Allow",
         "Action":[
            "ec2:ModifySnapshotAttribute",
            "ec2:CopySnapshot",
            "ec2:RegisterImage",
            "ec2:Describe*"
         ],
```

    - Run commands to attach “trust-policy.json” and “role-policy.json”

```
aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

    - Create new file on your local machine called “containers.json”

```
[
  {
    "Description": "metasploitable3 ",
    "Format": "ova",
    "UserBucket": {
        "S3Bucket": "mtdbucket",
        "S3Key": "Metasploitable3-ub1404.ova"
    }
  }
]
```

    - Run commands to import the instance and check on its status

```
aws ec2 import-image --description "metasploitable3" --license-type BYOL --disk-
containers file://containers.json
aws ec2 describe-import-image-tasks --import-task-ids import-ami-<fill-in-here>
```

- Launch instance using Metasploitable3 image
    - On AWS, go to EC2 on Management Console
    - Click on *Launch Instance*
    - Click on *My AMIs*
    - Select newly imported instance image
    - Click *Review and Launch*
    - Click *Edit Security Groups*
        - Click *Add Rule*
        - For Type choose “All traffic”
        - For CIDR enter “0.0.0.0/0”
        - Click *Review and Launch*
    - Click *Edit* Instance Details
        - Change Number of Instances field to however many instances you want
        - Change Network to your VPC that you created
        - Click *Review and Launch*
    - Click *Launch*

    - Select “Create a new key pair” from dropdown
    - For Key pair name field enter “metasploitable3-key”
    - Click *Download Key Pair,* and make sure that you save the “metasploitable3-key.pem” file somewhere that you remember
    - Click *Launch Instances*
- On the EC2 Management Console, select the security group for the instance that you just created
- Click on the Security group ID
- Click on *Edit inbound rules*
- Click on *Add Rule*
- For Type choose “All traffic”
- For Source enter “0.0.0.0/0”
- Click *Save Rules*

Accessing AWS VPC and Connecting to Kali Linux and Metasploitable3 Instances

- Associate Client VPN endpoint
    - On AWS, go to VPC on Management Console
    - Under *Virtual Private Network(VPN),* click on *Client VPN Endpoints*
    - Select the correct endpoint, and then click on *Associations*
    - Click Associate
        - For the *VPC* field, select your VPC that you created earlier
        - For the *Choose subnet to associate* field, select your subnet that you created earlier
        - Click *Associate*, and then click *Close*
    - Wait until the *State* field of the endpoint says “Available” with a green dot next to it, and then proceed to the next step
- Start Instances
    - On AWS, go to EC2 on Management Console
    - Click on *Running Instances*
    - Select the instances that you would like to start, and make note of the private IP address for each one
    - Click on *Actions > Instance State > Start*
    - Click *Yes, Start*
    - Wait until the *Instance State* field says “running” with a green dot next to it, and then proceed to the next step
- Start Tunnelblick
    - Click on the Tunnelblick icon on your toolbar
    - Click on *Connect mtd-vpn*
- Connecting to Metasploitable3 instance (Click on *Connect* on EC2 Management console for more information)
    - Navigate to the directory where you saved “metasploitable3-key.pem”
    - From command line, run the following commands

```
chmod 400 metasploitable3-key.pem
ssh -i “metasploitable3-key.pem” vagrant@private-ip-address
```

    - Enter “yes” if prompted
    - Enter password “vagrant”
- Connecting to Kali Linux instance (Click on *Connect* on EC2 Management console for more information)
    - From command line, run the following commands

```
chmod 400 kali-linux-key.pem
ssh -i “kali-linux-key.pem” ec2-user@private-ip-address
```

    - Enter “yes” if prompted

Shutting Down VPC

- Type “exit” in command line to exit instances
- Exit Tunnelblick
    - Click on Tunnelblick icon on toolbar
    - Click *Disconnect mtd-vpn*
- Shut down instances
    - Select instances that are running
    - Click on *Actions > Instance State > Stop*
- Disassociate Client VPN Endpoint
    - On AWS, go to VPC on Management Console
    - Under *Virtual Private Network(VPN),* click on *Client VPN Endpoints*
    - Select the correct endpoint, and then click on *Associations*
    - Select Association, and click *Disassociate*
    - Click *Yes, Disassociate*

Building Attack Script

- Create attacker.sh, attack-script.sh, and command_shell.sh on local machine (all can be found at https://github.com/MovingTargetDefenseCapstone/semester1)
- Copy attack scripts to Kali Linux instance on AWS
    - On local machine, navigate to the directory where you saved “kali-key.pem” and run the following commands

```
git clone https://github.com/MovingTargetDefenseCapstone/semester1.git
scp -i kali-key.pem semester1/attack-script.sh  ec2-user@private-ip-address:~/
scp -i kali-key.pem semester1/attacker.sh  ec2-user@private-ip-address:~/
scp -i kali-key.pem semester1/command_shell.sh  ec2-user@private-ip-address:~/
```

    - The attack scripts should now appear on the Kali Linux machine

Building Configuration Migration Scripts

- On the Metasploitable3 machine, if they do not already exist, create the following scripts in semester1 directory (can be found at https://github.com/MovingTargetDefenseCapstone/semester1)
    - mysql_payroll_app.php
    - postgres_payroll_app.php
    - mysql_payroll_app.py
    - postgres_payroll_app.py
    - add_rows_to_users.mysql
    - add_rows_to_users.psql
    - mysql_php_to_pg_php.sh
    - mysql_php_to_mysql_py.sh
    - mysql_php_to_pg_py.sh
    - pg_php_to_pg_py.sh
    - pg_php_to_mysql_py.sh
    - pg_php_to_mysql_php.sh
    - pg_py_to_mysql_php.sh
    - pg_py_to_mysql_py.sh
    - pg_py_to_pg_php.sh
    - mysql_py_to_mysql_php.sh
    - mysql_py_to_pg_php.sh
    - mysql_py_to_pg_py.sh
- Run the following commands to change executable permissions on migration scripts

```
chmod +x mysql_php_to_pg_php.sh
chmod +x mysql_php_to_pg_py.sh
chmod +x mysql_php_to_mysql_py.sh
chmod +x pg_php_to_pg_py.sh
chmod +x pg_php_to_mysql_php.sh
chmod +x pg_php_to_mysql_py.sh
chmod +x pg_py_to_mysql_py.sh
chmod +x pg_py_to_mysql_php.sh
chmod +x pg_py_to_pg_php.sh
chmod +x mysql_py_to_mysql_php.sh
chmod +x mysql_py_to_pg_php.sh
chmod +x mysql_py_to_pg_py.sh
```

- Create a script called migration.py (can be found also in the GitHub repository) that randomly switches between configurations after waiting a specified amount of time.

Troubleshooting

When you are executing one of the state change scripts you may get the following error:

```
ERROR 2002 (HY000): Can't connect to local MySQL server through
socket '/var/run/mysqld/mysqld.sock' (2)
```

If this happens, run the following command:

```
sudo mv /var/run/mysql-default /var/run/mysqld
```

Guide for Running the Demo and Expected Results

Attacker

- When the attacker.sh script is run, it takes three parameters: the CVE attack used for probing for the server language, the target IP address, and the source IP address.
- The options for the CVE attack are "CVE-2015-3306", "CVE-2014-3704", and "CVE-2010-2075".
- Once the script starts running, the attack occurs in three stages.
- The first stage is the probing of the server language on the target machine. This is done by one of three CVE attacks, which is specified by the parameter that the user enters when running the script.
- The time for extracting the information depends on which language it is, as well as which CVE is being used.

Probing Times (Server Language)

	CVE-2015-3306	CVE-2014-3704	CVE-2010-2075
PHP	14.630 s	19.135 s	14.567 s
Python	14.727 s	19.915 s	15.012 s

- The probing attack "CVE-2014-3704" runs more slowly than the other two, so most likely would not be selected by the attacker. For the other two attacks there is no advantage to using one over the other, so the attacker could use either one.

- Once the attacker has probed and extracted the server language, the script enters the second stage, which consists of using SQLMAP to probe for the database language.

- The time for extracting the information depends on the server language and the database language

Probing Times (Database Language)

	PHP	Python
MySQL	12.733 s	16.615 s
PostgreSQL	12.824 s	25.560 s

- The third and final stage of the attacker script is the actual attack using the information extracted from the probing attacks.
- The script runs a final SQLMAP attack using the server language and database language.
- The attack does not take much time because it has the specific information needed to exploit the system.
- The attack times recorded are for 5000 data rows, which can be adjusted by changing the add_rows_to_users.mysql file and the add_rows_to_users.psql file.

Attack Times (5000 data rows)

Defender Configuration	**Time**
MySQL, PHP	7.137 s
MySQL, Python	8.138 s
PostgreSQL, PHP	11.001 s
PostgreSQL, Python	11.986 s

Defender

- While this attack is going on, the defender can change its state by running the migration.py script, which changes the configuration after a certain amount of time specified in the script.
- After the time interval has elapsed, there is still a certain amount of time that it takes for the configuration to change. The time depends on how many data rows there are, which can be changing by editing the add_rows_to_users.mysql file and the add_rows_to_users.psql file, as mentioned before.
- The times displayed below are for 5000 data rows, which matches up with what we did for the attacker.
- The table is organized with the vertical axis being the starting configuration and the horizontal axis being the configuration it switches to.

Switch Times (5000 data rows)

	MySQL, PHP	MySQL, Python	PostgreSQL, PHP	PostgreSQL, Python
MySQL, PHP	0 s	0.101s	0.588 s	0.612 s
MySQL, Python	0.103 s	0 s	0.606 s	0.565 s
PostgreSQL, PHP	2.443 s	2.584 s	0 s	0.108 s
PostgreSQL, Python	2.409 s	2.449 s	0.105 s	0 s