← Blog

Setting secure defaults on AWS and avoiding misconfigurations

Wiz cloud security researcher, Scott Piper, suggests measures organizations can adopt to ensure secure defaults on AWS and improve their security posture.

Scott Piper
December 21, 2023

8 minute read

Table of contents

There are no features of AWS that are bad to use, but there are some features that may increase the risk of a security incident. One example is the use of IAM User access keys, which we discussed the risks of and showed how to avoid in a blog series titled **How to get rid of AWS access keys**. In this blog post, we'll discuss preventative measures you can take to ensure secure defaults on AWS, prevent risky features from being used, and prevent modification of these guardrails.

In an earlier blog post, **Using Service Control Policies to protect security baselines,** we showed SCPs that can be implemented with almost no risk of breaking existing workloads, but in this blog post we'll show how to deny or change functionality that may be in use by legitimate workloads. As such, you are encouraged to do some investigation before deploying these changes to ensure you don't have a production outage. This can be done by reviewing CloudTrail logs to ensure the functionality that is going to be prevented is not currently in use.

In many cases Access Advisor data in AWS also provides this information, which can even be easily checked across the entire Organization from the Organization Management account's view. In cases where this functionality is used, you can create exceptions in SCPs if necessary, such as for specific principals or AWS accounts. We believe that the SCPs we present here have a security benefit strong

enough to merit this investigation and a possible adjustment to existing ways of doing things.

This blog post assumes you apply the SCPs from the earlier blog post. There are limits on the number of SCPs that can be used in AWS, so you will need to combine many of these into a single SCP.

Preventing resources from being made public

Prevent S3 buckets from being made public

The first risk many think of for the cloud is having public S3 buckets with sensitive contents. S3 buckets have never been public by default, but until a few years ago, it was easy to make them public. AWS has added a lot of friction over the years for the process of making an S3 bucket public, with new S3 buckets now having S3 Public Block Access enabled by default as of **April 2023**. To make an S3 bucket public, turn off that feature and then change the bucket policy or ACLs.

This feature exists at both the account and bucket level and has a few different configuration settings if you still need some of the functionality that it prevents. Older S3 buckets do not have this enabled by default, so we recommend enabling it on all existing S3 buckets that are not public, and enabling the account level setting in all AWS accounts that do not have existing public S3 buckets.

For each AWS account, run the following to enforce all S3 buckets in the account to not be public:

```
aws s3control put-public-access-block \
    --account-id 123456789012 \
    --public-access-block-configuration '{"BlockPublicAcls": true, "Ignor
ePublicAcls": true, "BlockPublicPolicy": true, "RestrictPublicBuckets": t
rue}'
```

Alternatively, for each S3 bucket, run the following to enforce them to not be public:

```
aws s3api put-public-access-block \
    --bucket my-bucket \
    --public-access-block-configuration "BlockPublicAcls=true,IgnorePubli
cAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
```

Then apply the following SCP to prevent this control from being modified:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutAccountPublicAccessBlock",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": "*",
            "Effect": "Deny"
        }
    ]
}
```

Prevent AMIs from being public

EC2 instances are created from Amazon Machine Images (AMIs). Operating system vendors provide these publicly and customers can create their own. Unless you are an operating system vendor, you likely do not want to make an AMI public ever. In September of this year, AWS **released** a new feature to protect customers from making their AMIs public. New accounts, and accounts that had no AMIs public on October 16, 2023, automatically had this protection enabled.

Prevent this protection from being disabled with:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DisableImageBlockPublicAccess"
            ],
            "Resource": "*",
            "Effect": "Deny"
        }
    ]
}
```

Prevent public sharing of SSM documents

AWS Systems Manager can be used to send commands to one or more EC2s. This is done by sending documents to them. These documents can be shared with other accounts or even made public. It is possible these could contain information you do not want to have public, so you can prevent your accounts from sharing them publicly with the following command as described **here** (this needs to be run in each region with the region changed):

```
aws ssm update-service-setting  \
    --setting-id /ssm/documents/console/public-sharing-permission \
    --setting-value Disable \
    --region 'us-east-1'
```

You should then apply an SCP to prevent this from being undone:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenySSMDocPublicSharingSetting",
            "Effect": "Deny",
            "Action": [ "ssm:UpdateServiceSetting", "ssm:ResetServiceSett
ing" ],
            "Resource": "arn:aws:ssm:*:*:servicesetting/ssm/documents/con
sole/public-sharing-permission"
        }
    ]
}
```

Prevent Lambda URLs from being public

The serverless feature on AWS known as Lambda gained the ability in 2022 to be called directly, instead of through an API Gateway or other mechanism. These function URLs can be called either with or without authentication, so the following SCP should be applied to prevent the direct anonymous calling of them. This SCP will enforce that if function URLs are used, they must use IAM authentication.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:CreateFunctionUrlConfig",
                "lambda:UpdateFunctionUrlConfig"
            ],
            "Resource": " arn:aws:lambda:*:*:function:*",
            "Effect": "Deny",
            "Condition": {
                "StringNotEquals": {
                    "lambda:FunctionUrlAuthType": "AWS_IAM"
                }
            }
        }
    ]
}
```

Prevent RAM from sharing resources to external accounts

AWS Resource Access Manager (RAM) allows you to share resources such as VPCs with other AWS accounts. These can even be shared with accounts outside of your AWS Organization, and external accounts can also share resources with you. To prevent that, apply the following SCP:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ram:CreateResourceShare",
                "ram:UpdateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "ram:RequestedAllowsExternalPrincipals": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "ram:AcceptResourceShareInvitation",
            "Resource": "*"
        }
    ]
}
```

Learn more about AWS foundations

Prevent risky features

Prevent S3 ACLs from being used

S3 ACLs are commonly viewed as being deprecated in favor of using S3 bucket policies. S3 ACLs can be applied to either buckets or objects and can create difficulties in understanding if S3 data has been shared, and with whom. All new S3 buckets have this capability blocked, and you have to change the ownership control to allow it. We recommend setting existing buckets to “Bucket owner enforced”, which will disable ACLs as explained **here**. This can be done for existing buckets with:

```
aws s3api put-bucket-ownership-controls --bucket EXAMPLE-BUCKET --ownersh
```

```
ip-controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

To prevent the ownership controls from being modified, the following SCP should be applied:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:PutBucketOwnershipControls",
            "Resource": "*",
            "Effect": "Deny"
        }
    ]
}
```

Prevent IAM users and access keys

I wrote a 3-part blog series on **How to get rid of AWS access keys** and what the better alternatives are. As a brief summary: access keys never expire and they can end up in places they shouldn't, where attackers find them. You should use IAM roles instead of IAM users.

You can prevent IAM users from being created, along with any access keys for existing users, using the following SCP:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": ["iam:CreateAccessKey", "iam:CreateUser", "iam:CreateLo
ginProfile"],
        "Resource": "*"
    }
}
```

Prevent IMDSv1

EC2 instances obtain their credentials for their IAM roles through the Instance Metadata Service (IMDS). IMDSv2 is the most recent and most secure version of this service, which mitigates risks of SSRF (server-side request forgery) and some other attacks. The older version, IMDSv1, should therefore be prevented. The following SCP comes from the AWS SCP examples **here,** and enforces IMDSv2 by preventing IMDSv1 from being used. The default creation of EC2 instances from the APIs, unfortunately is for them to allow IMDSv1 so this SCP can be more difficult to roll out.

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Effect":"Deny",
            "Action":"ec2:RunInstances",
            "Resource":"arn:aws:ec2:*:*:instance/*",
            "Condition":{
                "StringNotEquals":{
                    "ec2:MetadataHttpTokens":"required"
                }
            }
        },
        {
            "Effect":"Deny",
            "Action":"ec2:RunInstances",
            "Resource":"arn:aws:ec2:*:*:instance/*",
            "Condition":{
                "NumericGreaterThan":{
                    "ec2:MetadataHttpPutResponseHopLimit":"3"
                }
            }
        },
        {
            "Effect":"Deny",
            "Action":"*",
            "Resource":"*",
            "Condition":{
                "NumericLessThan":{
                    "ec2:RoleDelivery":"2.0"
                }
            }
        },
        {
            "Effect":"Deny",
            "Action":"ec2:ModifyInstanceMetadataOptions",
            "Resource":"*"
        }
    ]
}
```

Learn more about AWS security best practices

Enforce security measures

Restrict network modifications

One of the risks of the cloud is the ease with which resources can be made public in a default environment. To prevent resources from becoming network-accessible to the Internet, some organizations limit which users can set up network environments. To accomplish this, accounts are set up with an initial landing zone that provides them with a VPC where resources can be created and that does not have public IPs. This VPC is connected to a separate, dedicated network account. This might be done through VPC peering, sharing a VPC through RAM, a transit gateway, or other solutions.

The result is that you have a single account where configurations are performed to expose resources on the network and establish network monitoring or defenses. As part of this, you'll want to apply an SCP to the other accounts to prevent them from making their own paths to the Internet. AWS provides **guidance** on how to accomplish this, with this SCP:

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Effect": "Deny",
        "Action": [
            "ec2:AttachInternetGateway",
            "ec2:CreateInternetGateway",
            "ec2:CreateEgressOnlyInternetGateway",
            "ec2:CreateVpcPeeringConnection",
            "ec2:AcceptVpcPeeringConnection",
            "globalaccelerator:Create*",
            "globalaccelerator:Update*"
        ],
        "Resource": "*"
    }
}
```

Enforce EBS encryption

EBS volumes (the hard drives of EC2 instances) by default are unencrypted at rest. They can be encrypted by enabling a region-wide feature, which will cause future EBS volumes to be encrypted, but not existing volumes.

To enable this feature, in all regions in all accounts, run the following, while changing the region each time:

```
aws ec2 enable-ebs-encryption-by-default --region 'us-east-1'
```

The following SCP is **provided** by AWS to prevent this feature from being disabled:

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Effect": "Deny",
        "Action": [
            "ec2:DisableEbsEncryptionByDefault"
        ],
        "Resource": "*"
    }
}
```

Restrict admin access with network controls

Admin access should be limited as much as possible in production environments or wherever sensitive data may live. Infrastructure as code and review processes should be used, and cases where a human needs direct access should be rare. When that does happen, identity providers can enforce various controls for authentication, but many still find VPNs helpful for additional control. The following SCP can be used to enforce that a specific role named “admin” can only be used from the IP address 1.2.3.4:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": ["1.2.3.4/32"]
            },
            "Bool": {"aws:ViaAWSService": "false"},
            "ArnLike": {
                "aws:PrincipalARN": [
                    "arn:aws:iam::*:role/admin"
                ]
            }
        }
    }
}
```

Conclusion

We believe the modifications to AWS accounts and the SCPs presented in this article represent security improvements that all organizations should consider implementing. Although any SCP added to an environment presents a risk of

breaking the environment, we've tried to limit ourselves to higher return-on-investment restrictions that don't require architectural changes. To continue further, we recommend looking into implementing a data perimeter, and the SCPs that AWS makes available for that are **here**.

Tags #Security

Continue reading

Lateral movement risks in the cloud and how to prevent them – Part 3: from compromised cloud resource to Kubernetes cluster takeover

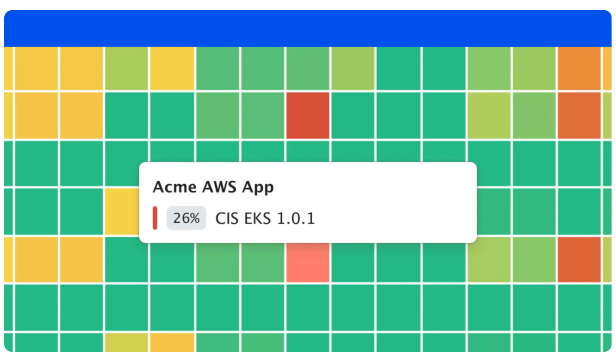
Lior Sonntag
February 23, 2023

In this third blog post, we will discuss lateral movement risks from the cloud to Kubernetes. We will explain attacker...

Compliance made easy with Wiz

Shaked Rotlevi, Daniel Klein
March 13, 2023

Stay compliant with Wiz's 100+ compliance frameworks, generate quick compliance reports, and remediate...



Assess your cloud compliance posture in minutes

Mattan Shalev
December 1, 2021

With Wiz, you can assess your compliance posture across industry standards and business units at a...

GET A PERSONALIZED DEMO

Ready to see Wiz in action?

w that if Wiz identifies
ıg as critical, it actually is."

Greg Poniatowski
Head of Threat and Vulnerability Management

Erreur du player

Le player a rencontré un problème. Nous le remettrons en service dès que possible.