



The Definitive Guide to Securing AWS S3 Buckets: Best Practices and Risks Unveiled - Panoptica



Table of Contents

- 03 Introduction
- 04 Part 1: What are S3 buckets and how can you access them?
- 11 Part 2: Risks of misconfigured S3 buckets and what you can do about them
- 17 Conclusion
- 19 About Panoptica



Introduction

In this eBook we will dive into:

Part 1: What are S3 buckets and how can you access them?

Part 2: The risks of misconfigured S3 buckets and what you can do about them

Cloud is complicated. And with all the new features, capabilities, etc., it's not likely to get any less complicated with time. Sorry to be the bearers of slightly bad news. For example, did you know the [AWS S3 bucket user guide](#) is 1,515 pages long? Go ahead and check.

In the unlikely event that you plan to read all that documentation cover to cover (kudos if you do), we're offering a "cliff notes" version of said documentation explaining what S3 buckets are and how to properly secure them.

Join us as we review the common pitfalls of using S3 buckets and as we reveal best practices for keeping your cloud environment more secure.

Let's dive right in...



What are S3 buckets and how can you access them?

What is an S3 bucket?

S3 stands for simple storage service, and it is AWS's cloud storage service. S3 provides the ability to store, retrieve, access, and back up any amount of data at any time and place.

As S3 is object-based storage, this means that all data is stored as objects.

Each object has three main components:

- The object's content
- The object's unique identifier
- The object's metadata (including its name, size, URL)

An object cannot be independent, it must exist within a bucket. There can be hundreds of buckets in each Amazon account and within each bucket, there can be hundreds of objects.

How to access objects within an AWS bucket

Access to a bucket is granted in the same way as with any other AWS resource – you need an explicit 'allow' and no 'denies' to be given access.

The explicit allow can be given in three ways – bucket policy, bucket ACL, and object ACL.



S3 bucket policy

This is a resource-based AWS Identity and Access Management (IAM) policy. You can add a bucket policy to a bucket to grant other AWS accounts or IAM users access permissions to the bucket and the objects inside it. Object permissions apply only to the objects that the bucket owner creates.

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn More](#)

Edit

Delete

Copy

```
{
  "Version": "2012-10-17",
  "ID": "Policy1617714685693",
  "Statement": [
    {
      "Sid": "Stmt1617714659754",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::noga-test-policy"
    },
    (
      "Sid", "Stmt1617714681648",
      "Effect": "Allow",
      "Principal": "*"
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::noga-test-policy/*"
    )
  ]
}
```



S3 bucket ACL/object ACL

This is a sub-resource that is attached to every S3 bucket and object. It defines which AWS accounts or groups are granted access. If it defines access as public, that will allow everyone permissions, whereas if it defines access only to an authenticated user group, this means anyone with an AWS account will have permissions. It also defines the type of access these users have, such as read or write access.

When you create a bucket or an object, Amazon S3 creates a default ACL that grants the resource owner full control over the resource.

Bucket ACL

Access control list (ACL) Edit

Grant basic read/write permissions to other AWS accounts. [Learn More](#)

Public access is blocked because Block Public Access settings are turned on for this bucket.
To determine which settings are turned on, check your Block Public Access settings for this bucket. [Learn more about using Amazon S3 Block Public Access](#)

AWS doesn't recommend granting access to the Everyone grantee
Anyone in the world can access the objects in this bucket. [Learn More](#)

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: <input type="checkbox"/> d4de7dc65a8f8caaa0738ce51eed3877e9aa9f898daee0f2bd2ea7a3dd0f37bf	List, Write	Read, Write
Everyone (public access) Group: <input type="checkbox"/> http://acs.amazonaws.com/groups/global/AllUsers	List	Read
Authenticated users group (anyone with a AWS account) Group: <input type="checkbox"/> http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group Group: <input type="checkbox"/> http://acs.amazonaws.com/groups/s3/LogDelivery	-	-

Object ACL

Access control list (ACL) Edit

Grant basic read/write permissions to AWS accounts. [Learn More](#)

Grantee	Object	Object ACL
Object owner (your AWS account) Canonical ID: <input type="checkbox"/> d4de7dc65a8f8caaa0738ce51eed3877e9aa9f898daee0f2bd2ea7a3dd0f37bf	Read	Read, Write
Everyone (public access) Group: <input type="checkbox"/> http://acs.amazonaws.com/groups/global/AllUsers	Read	-
Authenticated users group (anyone with a AWS account) Group: <input type="checkbox"/> http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-



Block Public Access

Block public access (bucket settings)

Public access is granted to buckets through access lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn More](#)

Edit

Block all public access

⚠ Off

Block public access to buckets and objects granted through *new* access control lists (ACLs)

✔ On

Block public access to buckets and objects granted through *any* access control lists (ACLs)

⚠ Off

Block public access to buckets and objects granted through *new* public bucket or access point policies

✔ On

Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

✔ On

Creating your own S3 bucket permissions

Amazon provides the ability to define settings for blocking access points, buckets, and accounts to help you to manage public access to Amazon S3 resources. By default, public access to new buckets, access points, and objects is not allowed.

The **block public access** settings consist of four options that you can apply in any combination to a bucket or to an entire AWS account. If you apply them to a whole AWS account, these settings will apply to every bucket in the account.

Here are the four options, in the same order as on the console, if you're looking to manage and access S3 bucket using AWS cli names:

- 1. Block public Acls** While set to **TRUE**, no new ACL definitions are allowed, but the existing ones still apply. Meaning, if there is a bucket with an ACL granting public access, the BlockPublicAcls will not affect it.
- 2. Ignore public Acls** While set to **TRUE**, it causes Amazon S3 to ignore all public ACLs on a bucket and any objects that it contains. Meaning, all public access granted by a bucket or object ACL does not apply.
- 3. Block public policy** While set to **TRUE**, no new policies can be attached to the bucket, but existing ones still apply. Meaning, if there is a bucket policy that grants public access, the BlockPublicPolicy will not affect it.
- 4. Restrict public buckets** While set to **TRUE**, if there is a policy that allows public access, the access is restricted only to AWS service principals and authorized users within the bucket owner's account.



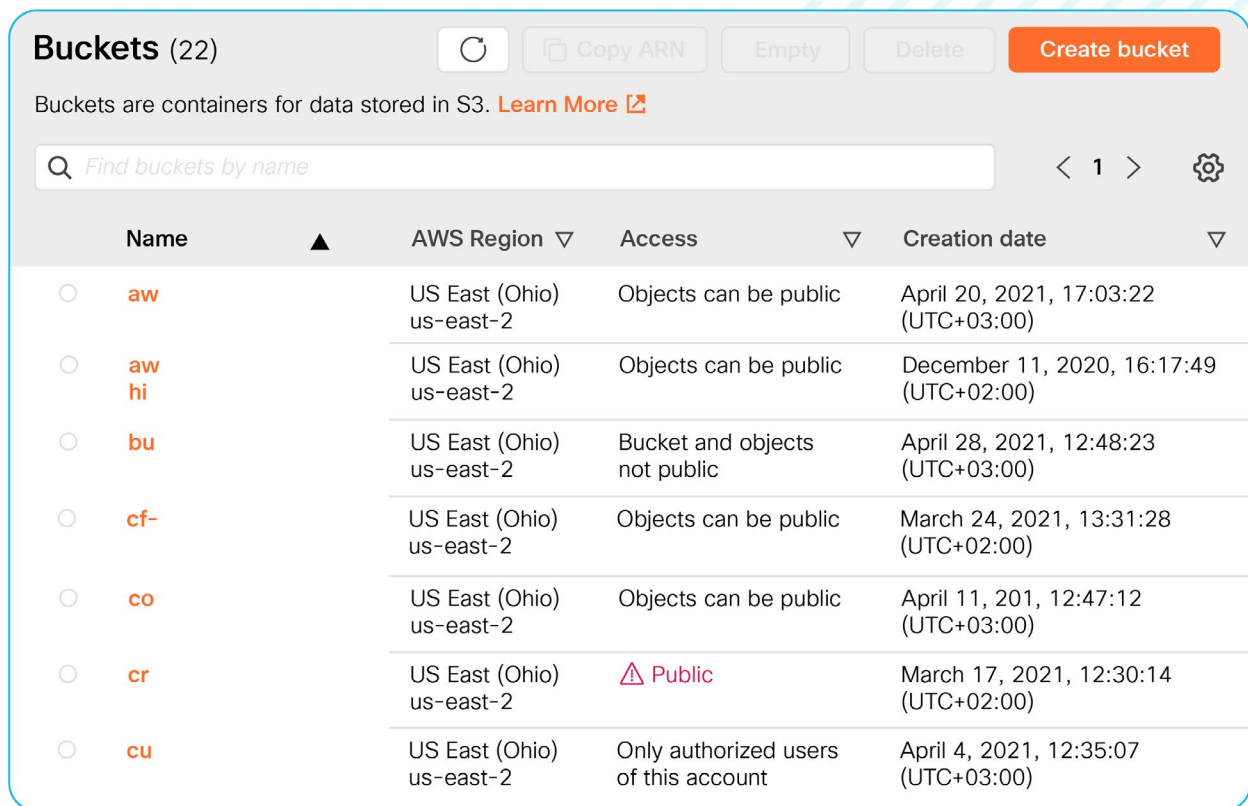
AWS bucket access

AWS evaluates the access scope for each bucket and presents this in the management console.

There are four access options:

- Public
- Objects can be public
- Only authorized users of this account
- Bucket and objects not public

These are the only potential outcomes of any access settings applied.

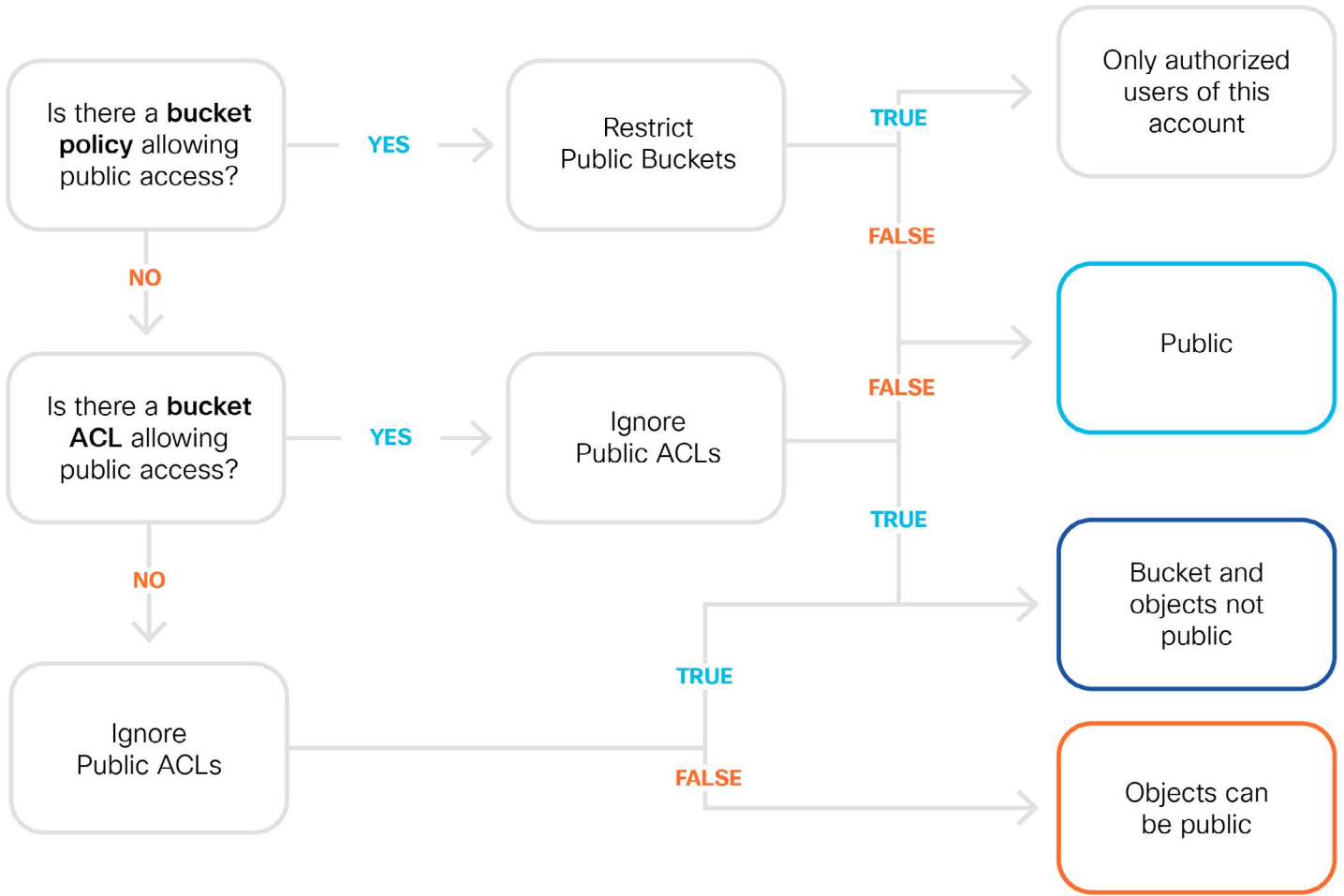


The screenshot shows the AWS Management Console interface for the 'Buckets' section. At the top, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. Below the buttons is a search bar with the placeholder text 'Find buckets by name'. The main content is a table with the following columns: Name, AWS Region, Access, and Creation date. The table lists seven buckets with their respective details.

Name	AWS Region	Access	Creation date
aw	US East (Ohio) us-east-2	Objects can be public	April 20, 2021, 17:03:22 (UTC+03:00)
aw hi	US East (Ohio) us-east-2	Objects can be public	December 11, 2020, 16:17:49 (UTC+02:00)
bu	US East (Ohio) us-east-2	Bucket and objects not public	April 28, 2021, 12:48:23 (UTC+03:00)
cf-	US East (Ohio) us-east-2	Objects can be public	March 24, 2021, 13:31:28 (UTC+02:00)
co	US East (Ohio) us-east-2	Objects can be public	April 11, 201, 12:47:12 (UTC+03:00)
cr	US East (Ohio) us-east-2	Public	March 17, 2021, 12:30:14 (UTC+02:00)
cu	US East (Ohio) us-east-2	Only authorized users of this account	April 4, 2021, 12:35:07 (UTC+03:00)



We have created the following diagram to better explain how AWS evaluates the access:



The problem with AWS's evaluation

As you can see, these four access options don't necessarily allow you to provide definitive answers to whether your objects are public or not, and which buckets are secure. While 'Public' is a black and white outcome, and so is 'Bucket and objects not public,' the other two are more open ended and could lead to some confusion. In particular, the outcome of 'Objects can be public' leaves your security teams none the wiser about whether items are accessible or not.

Let's break it down with an S3 bucket example

Let us assume we have a developer who works with a bucket, and in it, they put a folder with objects, using its ACL to make it publicly accessible. At some point, they want to store some sensitive information in the same folder, so they need to make it non-public.

To do so, they use the block public access settings, specifically IgnorePublicACL, and the bucket and everything in it, is no longer public. The official status is 'Bucket and objects not public.'

Time goes by, our developer leaves the company, and a new developer joins the team and takes over their role.

This new developer wants to create a new public folder in the same bucket for the company's website, so he changes the IgnorePublicACL back to false, and grants public access to the new folder using its ACL. The bucket status has now changed and is now 'Objects can be public.' This new developer is not aware that their actions have changed the status of other information inside the bucket, as they do not receive an alert about the old developer's folder with the sensitive information that has now become public again following their actions.

The question is... what can we do about it?

How can we be sure that we know what objects are public, and which are not? In the next section, we will be discussing the specific risks of misconfigured S3 buckets and human error (as described above), and we will dive into our own Panoptica research that uncovered a cross-account attack path you should know about.



Risks of misconfigured S3 buckets, and what you can do about them?

Risks involving misconfigured buckets

Amazon S3 comes from simple storage service – three words beginning with the letter “S,” – and S3 was born!

There are two main risks when considering misconfigurations on AWS buckets, read and write access permissions.

1. First, a misconfigured S3 bucket that allows public read access can lead to a customer data breach.
2. Second, a misconfigured S3 bucket allowing public write access can be used to serve or control malware, damage a website hosted on a cloud storage service, store any amount of data at your expense, and even encrypt your files for the purposes of demanding a ransom.

Suddenly nothing so “simple storage service” about it, eh?



Real life examples of these kinds of S3 bucket security issues

There are several high-profile examples of access management issues or customer data breaches and security incidents that occur from misconfigured S3 buckets.

Booz Allen Hamilton is a leading U.S. government contractor, famous for [a data breach that involved misconfigured buckets](#). Booz Allen Hamilton left sensitive data on AWS S3, publicly accessible, exposing 60,000 files related to the Department of Defense. In a case like this, the baseline expectation is that there would at least be default encryption, but a huge amount of data in the S3 bucket was stored with no encryption at all. The company claimed that they had no need to encrypt data as the data itself was not classified, but it included credentials to sensitive government systems, credentials belonging to senior engineers at Booz Allen, and more.

Another example is Verizon, an American wireless network operator, who failed to secure their security controls or bucket policies. This company suffered from two data breaches only a few months apart, [exposing more than 6 million customer accounts](#). That's a huge amount of data!

Both breaches were caused by S3 misconfigurations.

60,000

files exposed related to the Department of Defense

Booz Allen Hamilton is a leading U.S. government contractor, famous for a data breach that involved misconfigured buckets.



Researching the current status of AWS buckets

To understand the scope of the issue, we started by searching a few AWS customer environments to examine their public S3 buckets, and their “objects can be public” S3 buckets. (See part one for a list of definitions).

The results were quite interesting. The average amount of public buckets stands at almost 4% per company, and the average amount of “objects can be public” is around 42%.

This means that almost 50% of a company’s buckets could potentially be misconfigured!

To describe our research in more detail, here is what we tried to accomplish:

We wanted to get a hermetic image of the possible outcomes of using the block public settings. We simulated every combination, trying to access the bucket and objects, and then documented every result.

The most interesting cases were when AWS evaluated the bucket status as “objects can be public,” and as mentioned in part 1. That’s not surprising because this is the most confusing definition.

It’s clear that because AWS evaluation is happening at the bucket level, and it does not take into consideration the objects’ ACLs, this will invariably open companies up to risk as this is the key for understanding where they might be vulnerable.

Current status of AWS buckets

We started by searching a few AWS customer environments to examine their public S3 buckets, and their “objects can be public” S3 buckets.

4%

Average amount of public buckets

42%

Average amount of “objects can be public”

50%

Average amount of buckets that could potentially be misconfigured



Panoptica's Python research

As our research indicates, the “object can be public” status is very common. This nearly 50% occurrence prompted us to build a solution that could help to mitigate this issue.

An unexpected find!

Cross-account attack on S3 buckets

While performing our research, we spent a significant amount of time analyzing bucket policy examples, so that we could fully understand which ones allow public access.

It was then that we came across an interesting case, where the policy grants AWS Config and CloudTrail service access to a bucket. These two services use S3 Buckets to store their output, and while setting them, the user must choose whether to create a new bucket, to use an existing one from their own account, or to use an existing one from another account.

CloudTrail example

```
{
  "Sid": "AWSCloudTrailWrite20150319",
  "Effect": "Allow",
  "Principal": {"Service": "cloudtrail.amazonaws.com"},
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::myBucketName/[optional prefix]/AWSLogs/myAccountID/*",
  "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
}
```



Config example

```
{
  "Sid": "AWSConfigBucketDelivery",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "config.amazonaws.com"
    ]
  },
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::targetBucketName/[optional] prefix/AWSLogs/sourceAccountID-WithoutHyphens/Config/*",
  "Condition": {
    "StringEquals": {
      "s3:x-amz-acl": "bucket-owner-full-control"
    }
  }
}
```

Many users can ease the process of specifying several different accounts as a list by creating a general path instead.

A general path can be:

- `arn:aws:s3:::{bucket-name}/*`
- `arn:aws:s3:::{bucket-name}/AWSLogs/*`
- `arn:aws:s3:::{bucket-name}/AWSLogs/*/
Config/*`

This case is common when using the AWS Organizations and Control Tower, which creates a dedicated account for logging and auditing. When you have only a few accounts, that can work with a bucket policy with several resources for the other accounts. However, when you have 100 or 1,000 accounts, wildcards are mostly in use, which can open you up to risk.

The tricky thing in this case is that since the principal is the AWS service, the source account is evaluated from the resource, as opposed to normal bucket policies.

Each one of the general resource path patterns above will enable any AWS account to define the bucket as their Config bucket, and by doing that, use it to store their data at your expense. It does not allow them to read objects that are stored in the bucket, only to write new ones, making it the second category of risk we discussed above.

This configuration opens your bucket to unauthorized writes from any Amazon Web Services account.

****Note:** In our research, we tested AWS CloudTrail and Config but this issue may be valid for other AWS Services that use S3 Buckets to store their data by default. In other cases, the permission may also be a read permission that can lead to a serious data breach.**



Securing your S3 buckets

You wouldn't ignore your physical security risk, and this is no different. Here are the top tips for staying informed and proactive of file storage in Amazon S3 buckets and S3 bucket misconfigurations and access management.

- 1. Stay alert.** Continuously assess the S3 bucket states in your account and your access control lists – you can use config rules or AWS “Access Analyzer for S3” for that. Notice that the “Access Analyzer for S3” does not cover the two main cases we discussed – the cross-account attack and the bucket with “objects can be public” status that has publicly accessible objects.
- 2. Don't assume.** Never assume that buckets are private. To deny public access, it is preferable to use the block public access, and not a deny policy.
- 3. Avoid wildcards.** Your buckets and objects are your responsibility. Try to make your policies as specific as you can.



Conclusion

With any cloud service, there are associated risks and possible vulnerabilities. Amazon's S3 is no exception. This is a widely used service and due to human error or simple out of the box default settings, your cloud environment or organizational data could be at risk. As with any service, it is essential to conduct due diligence and ensure a deep understanding of how services handle access rights and privileges to put best practices in place.

As part of enjoying the services provided by AWS and other cloud service providers, there is shared responsibility to ensure proper access and authentication occurs to keep your data safe and secure. While AWS certainly works hard on their side to ensure they implement the highest levels of protection, we have highlighted in this eBook, that even the best of services can suffer from common misconfigurations that lead to unintended risks.

To ensure you are doing your part, an initial step is leveraging a tool that provides as much visibility as possible across your cloud environment.



The Way Forward

A single, comprehensive platform to meet dynamic shifts in cloud security requirements

With rapid changes to the cloud occurring on a constant basis, a complete cloud security platform is required to protect and secure your cloud stack and business assets, from build to runtime. At its core, best practices indicate that organizations, to minimize unintended risks, including those connected to misconfigured S3 buckets, should partner with vendors who offer:

Through a unified platform that provides all aspects of cloud security under the umbrella of a singular, integrated platform – improve your ability to proactively protect your most valuable assets while reducing the effort required from your team.

- **Graph-based technology at the heart of the system:** Gain complete visibility of the entire cloud stack across cloud providers and better understand all potential areas vulnerable to attack.
- **Proactive cloud security research:** Preemptively deep dive into cloud security issues, staying one step ahead of security trends and potential attackers.
- **Prioritization:** Examine a variety of layers within multi-cloud environments to build the story behind each disparate security finding and better understand the relevant context between findings to surface what risks really matter.
- **Remediation:** Out of the box remediation to reduce the time it takes for your Sec/DevOps teams to better secure any discovered vulns. Dynamic deny guardrails can be tailored to your organization's specific requirements, meaning no one size fits all. Instead, a customized solve for your specific risks.





Panoptica is Outshift by Cisco's cloud native application protection platform (CNAPP) that uncovers and remediates vulnerabilities during development through to production, ensuring your applications and cloud environments are secure and compliant. Through graph-based technology, the platform is able to unlock visual insights, critical attack paths, and speed up remediation to safeguard your modern apps across multiple hybrid cloud platforms.

To learn more visit: <https://www.panoptica.app/>