AWS S3 Buckets Security

Avoid Common Mistakes When Deploying Cloud-based Services

CONTENTS

Bishop Fox Guide - AWS S3 Buckets Deployment
V1, updated July 2018-GK

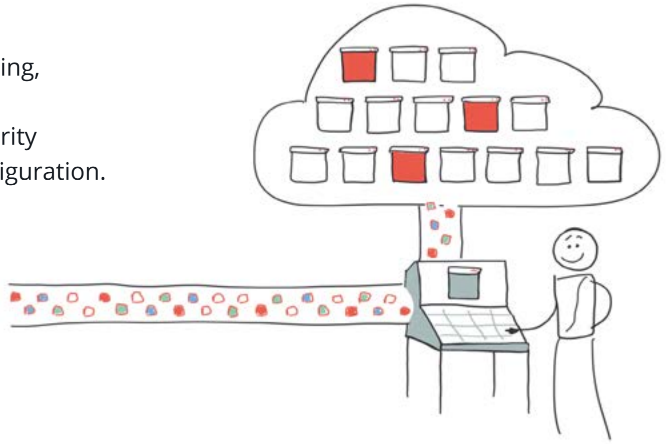
BISHOP FOX®

INTRODUCTION

Cloud-based services such as Amazon's Simple Storage Service, or S3 for short, lure businesses in with cost-savings, efficiency, and convenience. Enterprises have a myriad of uses for these services, S3 in particular, which can be used for hosting static content such as log files, backup files, images, and lots more.

But as with every convenience in life, there's usually a tradeoff accompanying it. With S3 often acting as a centralized storage repository for many different types of projects, mistakes can be made configuring access as organizations realize they may need to share stored resources not only across an enterprise, but also with business partners and contractors.

If the last 18 months have shown us anything, it's that even the most savvy businesses, government agencies, and resourced security teams can be undone by a simple misconfiguration.



The Danger of Insecure Buckets

Data leaks and manipulation of files carried out by unauthorized and malicious external attackers have dominated the headlines, fighting malware-based attacks such as ransomware and banking Trojans for attention and IT resources.

The mistakes are easily correctable—usually by leaving access to S3 buckets at their default setting of private—but often, admins in charge of an AWS S3 bucket may not be aware of the risks they're accepting by making data accessible to third parties. Alternatively, they may choose to embrace security by obscurity and hope that by hiding among the tens of thousands of AWS buckets and billions of data records stored in the cloud, they won't fall prey to an attacker. Regardless of the reason, the results can be devastating to organizations.

S3 Leaks: The Achilles Heels of AWS

Since the start of 2017, S3 leaks have been the center spoke in dozens of high-volume data leaks, and it's not just smaller, less-resourced IT shops that have been victimized. Giant telecommunications provider Verizon is among the victims, as is Accenture, a top-tier management and consulting operation, as well as the U.S. Army Intelligence and Security Command (INSCOM). Wrestling and sports entertainment kingpin World Wrestling Entertainment (WWE) and the National Football League also had fan and player information, respectively, spilled into the public realm by someone who accessed an insecure bucket from the outside.

Let's illustrate some of the most egregious data leaks and why they happened:

CHICAGO VOTER ROLL	VERIZON	INSCOM	ACCENTURE
1.8 million voters affected	14 million Verizon customers exposed	Top secret data exposed	encryption keys, secret API data and other sensitive information exposed

- **CHICAGO VOTER ROLL**: Voter registration data for all of Chicago's 1.8 million voters was found in an AWS S3 bucket in August 2017. The bucket was configured for public access by a partner, Election Systems & Software, a Nebraska-based voting machine and election management systems software vendor.
- **VERIZON**: Personal data belonging to as many as 14 million Verizon customers in the United States was exposed by NICE Systems, an Israel-based surveillance and analytics company. Terabytes of customer data was found in a S3 bucket configured for public access by an engineer at NICE Systems; Customer names, addresses, account details, and PINs used to authenticate customers at call centers were exposed.
- **INSCOM**: Twice in 10 days in November 2017, U.S. Department of Defense data was found in a publicly accessible S3 bucket belonging to a third-party defense contractor Invertix. Some of the data was top secret and not meant to be seen by foreign nationals, a potential national security risk.
- **ACCENTURE**: The massive consulting and management firm suffered a catastrophic leak in October 2017 when private encryption keys, secret API data, and other sensitive information was found in a S3 bucket configured for public access. The information could have been leveraged to not only attack the consulting firm, but also its clients.

It's Nearly Impossible to Tell if Your Data Was Accessed

In each case, the publicly exposed buckets were found by security researchers who privately reported their findings to the respective organizations, most of which took steps to remediate the problem and close off their exposure in hours or days.

But the scope of the risk posed by each leak is immeasurable. In each case, it's difficult to tell whether any of the data had been accessed or downloaded before the researchers stumbled upon it. Unless logging and other tracking had been turned on, it's impossible to tell whether the exposed data had fallen into a malicious party's hands.

Another commonality is that in each case above, a third party was responsible for the exposure. Data had been shared with a vendor, business partner, or contractor who needed a particular data set for a project and uploaded it to Amazon S3 without proper security controls in place, largely for the sake of convenience and sharing among colleagues such as developers and analysts.

The biggest transgression is that also in each example, the default private setting had been changed to public by a partner. A partner such as a contractor or developer will deliberately make the switch from private to public and fail to revert the configuration to private, leaving data exposed.

Amazon, for its part, makes a number of controls available to its customers. Publicly exposed S3 buckets are not the default configuration preferred by Amazon in its documentation and advice to customers. Instead, the cloud giant makes available the number of access controls it recommends users implement, including: access control lists for the buckets themselves, the objects stored in them, bucket security policies and identity, and access management policies. Amazon also provides versioning, multifactor authentication, logging, and encryption.

This paper will cover each security control option available to users and stress the importance of implementing each to keep secrets private and out of the hands of malicious entities.

PROVISIONING AND ACCESS CONTROLS

As demonstrated in the examples above, it's simple to make a devastating mistake when provisioning access to objects stored inside an Amazon S3 bucket. Managers should provision access based on the principle of least privilege and adhere to established identity and access management policies. A lack of policies or understanding of provisioning can put trade secrets, customer and employee data, and even data that threatens national security at risk for exposure.

Amazon allows administrators to manage S3 permissions in four ways:

- Identity and access management policies (IAM)
- Access control lists for buckets (ACL)
- Bucket policies
- Access control lists for objects stored in a bucket (Object ACL)

Each of the four approaches has a distinct purpose and can introduce some confusion if not strictly adhered to and properly implemented. It's recommended that organizations rely on an IAM policy to grant permissions because the other three should be reserved for specific use cases.

IAM policies are the recommended method for providing S3 access to users in the organization because they ensure that access permissions are managed from a central location. Permissions on other AWS services are managed from the IAM service; it is considered a best practice to use it to manage S3 as well.

IAM best practices are to specify permissions in a policy, attach the policy to a group, and place users into groups to provide them with permissions. Policies should never apply to users because this would conflate management as the Amazon Web Services environment grows relative to its organization, see illustration in figure 2. In addition, use AWS managed policies rather than creating new ones. Many AWS managed policies exist that are intended to provide organizational roles with access according to the principle of least privilege. With regard to how policies and Access Control Lists should interact, Amazon states that an explicit deny should overrule an explicit allow; an explicit allow should overrule an implicit deny; and without an explicit allow, access is implicitly denied.

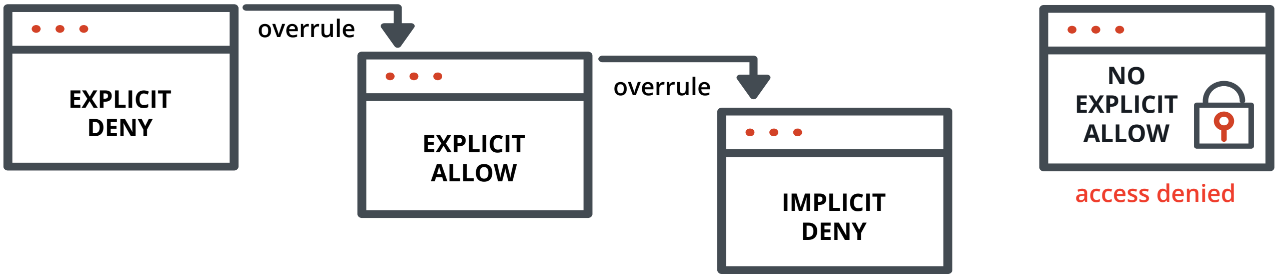


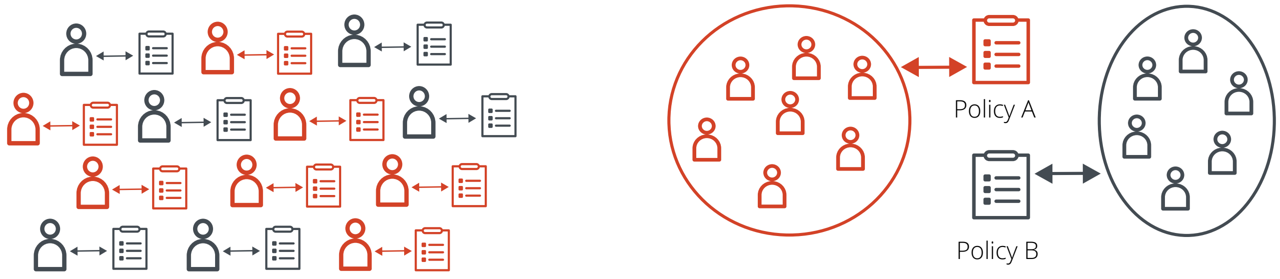
Figure 1: Policy and Access Control List interaction recommended by Amazon

For example, if a policy attached to a user account explicitly allows access to a bucket, but a bucket policy explicitly denies access to the user, then the user is denied access to the bucket. This works the same way if the bucket policy explicitly allows access but a user policy explicitly denies access, or if a bucket ACL allows access but a policy denies it.

Bucket Access Control Lists, or ACLs, grant the Amazon S3 Log Delivery group write-access to the bucket. To have Amazon S3 deliver access logs to a user's bucket, an admin will need to grant write-permission on the bucket to the Log Delivery group. The only way to grant necessary permissions to the Log Delivery group is via a bucket ACL.

A bucket policy is applied to a bucket rather than a user or group, unlike an identity and access management policy. A bucket policy can be used to provide bucket access for another AWS account; IAM policies do not allow for cross-account access.

Finally, **Object Access Control Lists** may also grant cross-account access to objects rather than buckets. Object ACLs should be used if access permissions need to vary between objects in a bucket in situations where cross-account access is required.



Attaching a policy to each user is not scalable

Attaching a policy to a group is scalable.

Figure 2: Illustration of policy scalability

EVERYONE AND AUTHENTICATED USERS

Many administrators responsible for an Amazon S3 bucket get into trouble with the Everyone and Authenticated Users groups and it's here where many public exposures of sensitive data can be traced. While the names may imply a reasonable difference in the permission allocated between the two groups, as it turns out, they're not too dissimilar.

Although most administrators understand the implications of using the Everyone group, there is often confusion around the Authenticated Users group.

If an S3 bucket is accessible to "Authenticated Users", all that someone needs to do is sign into their personal account and AWS will provide access to the S3 bucket objects. Alternatively, S3 buckets configured to grant access to the "Everyone" and the "Authenticated Users" groups can easily be accessed by an attacker via the AWS command line interface.

For example, for a bucket configured to "Everyone," by adding an argument called no-sign-request, the Amazon Web Services command line interface (CLI) is instructed not to load credentials. An attacker who can install the AWS CLI would be able to access an S3 bucket configured in this manner. If the bucket is configured for "Authenticated Users," and an attempt is made to list the objects in a bucket without loading credentials, access is denied. An attacker who can configure the AWS CLI with an access key from a free AWS account would be able to remove the no-sign-request argument and access the objects.

Many administrators assume that "Authenticated Users" means that any user authenticated to the organizational AWS account will have access to the S3 bucket, but this is incorrect. Instead, the group includes anyone authenticated to any AWS account. In other words, there is little difference between the "Everyone" and the "Authenticated Users" group because anyone can sign up for a free AWS account, see figure 3 below. If an S3 bucket is accessible to "Authenticated Users," any user who signs into their personal account will receive access to that S3 bucket.

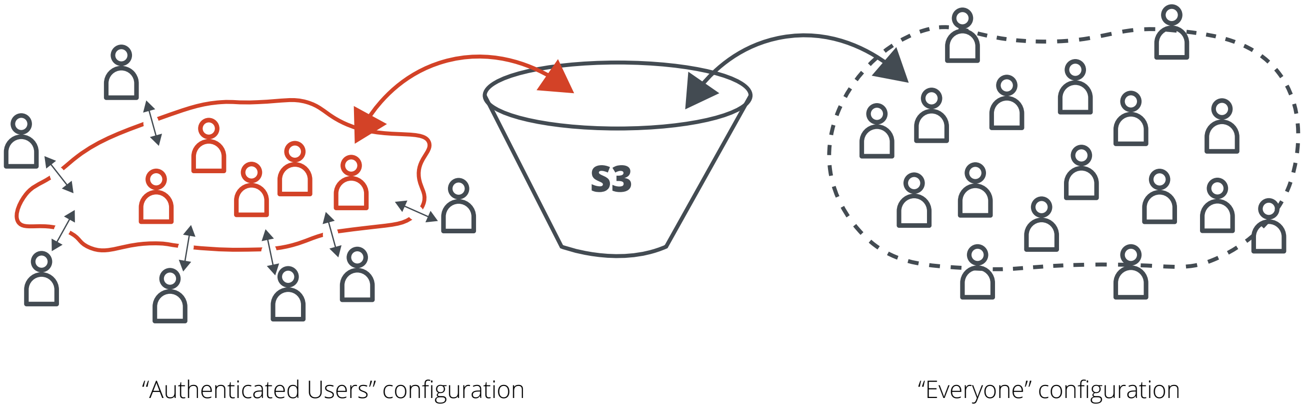


Figure 3: Illustration of the little difference between "Authenticated Users" group and "Everyone" group.

Do not use the "Everyone" and "Authenticated User" groups when configuring access unless the bucket is meant to be accessible by absolutely everyone. Configuring S3 buckets to allow access from either of these groups has led to the public exposure of information that was meant to only be accessible internally.

VERSIONING AND MULTIFACTOR AUTHENTICATION DELETE

Versioning is a critical feature available to S3 bucket administrators. Unintended user actions or application failures that may corrupt an object could be fatal to the integrity and availability of data. Versioning allows an admin to easily recover from such situations, for instance, allowing one S3 bucket to store two objects with the same key but different version IDs.

With versioning, overwriting an object results in a new object version in the bucket while allowing users to still retrieve previous versions. Similarly, deleting an object inserts a deletion marker in the bucket while still allowing users to retrieve previous versions. Enabling versioning on a bucket does not incur costs to the AWS account, but each object version in the bucket counts towards the total S3 storage cost.

This can be a crucial factor for an organization. To keep storage costs down, administrators can create a lifecycle policy. Lifecycle policies allow automatic actions on objects after a specified amount of time has passed. For example, old versions of objects can be automatically moved to Amazon Glacier for long term storage or be deleted after the specified time period. Glacier is a separate Amazon service that is secure and low-costs and allows for long-term data archiving.

To further protect buckets, a separate versioning feature requires multifactor authentication for the deletion of objects stored in a bucket. MFA Delete forces users to include a multifactor authentication code, such as a PIN sent to a mobile device, in requests to delete objects. While it may make sense for S3 administrators to enable MFA Delete as an additional layer of integrity and protection for buckets and stored objects, there are some requirements that may prevent administrators from opting in to the feature for all S3 buckets.

- Enabling and disabling MFA Delete can only be done with the AWS root account through the API; MFA Delete cannot be configured through the console
- After enabling MFA Delete, only the AWS root account can delete object versions from the S3 bucket, and then only through the API; deleting object versions from the console will no longer work
- A valid MFA token needs to be included with each delete request and a delete request can only be issued for one object and version at a time, making the deletion of many objects a manual and time-consuming task
- Before administrators can enable MFA Delete, existing lifecycle rules need to be deleted

Enabling and implementing Multifactor Authentication Delete requires some heavy lifting on an administrator's part. While authorized non-root user accounts can still delete objects from an S3 bucket, those deletions merely result in the creation of a delete marker. To fully remove an object from an S3 bucket, including all versions, the root account must be used through the AWS API. This requires the creation of API keys for the root account, which is against security best practices. An administrator would need to:

- create a new set of API keys for the root account;
- issue S3 delete requests through the API individually with a new MFA code for each request;
- delete the API keys for the root account.

Alternatively, an administrator may use the root account to disable MFA delete on the S3 bucket and use the console to mass-delete objects from the bucket. They would then have the option of re-enabling MFA delete once this is complete. This is a faster but riskier method for object deletion. Due to the administrative burden of using MFA Delete, it is recommended to enable it on mission-critical buckets, but to protect less-critical buckets through other methods described in this document.

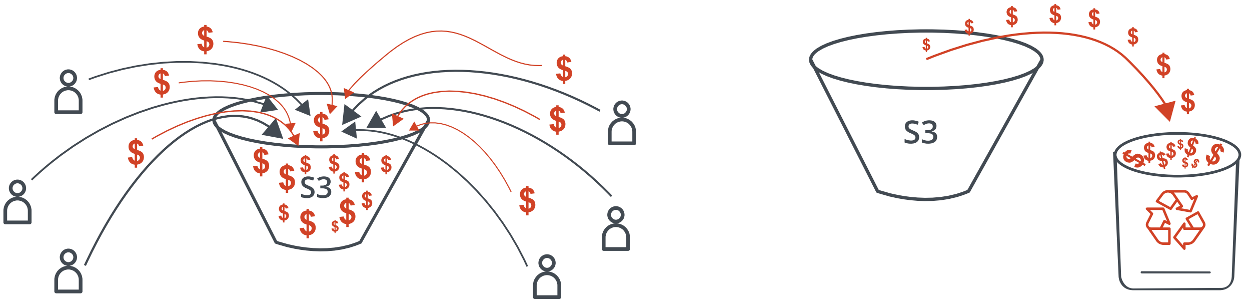
LOGGING AND MONITORING

As mentioned earlier, when objects stored inside an S3 bucket are exposed, it's unlikely an organization will have a handle on whether sensitive data or secrets have been accessed unless some type of monitoring and logging is turned on. As evidenced by the multitude of public exposures that have been discovered, this hasn't often been the case.

Administrators can turn on access logging, which is found under the Bucket Properties table; this provides an audit trail for activity on the bucket and is recommended for particularly sensitive buckets or objects.

Likewise, if there is a need to know which users accessed a particular bucket at certain times, access logging should be enabled. However, there are costs associated with access logging, so administrators should also consider financial resources when deciding which buckets should enable access logging.

One way to limit the costs associated with access logging is to migrate the log files and then delete them from the bucket. Billing for S3 is determined on storage use, so logging becomes increasingly expensive with every log file left in the bucket. Leveraging existing internal infrastructure to store log files long-term allows an organization to enjoy the benefits of bucket logging while keeping expenses down, see figure below.



Expensive log files

Cost saving migration

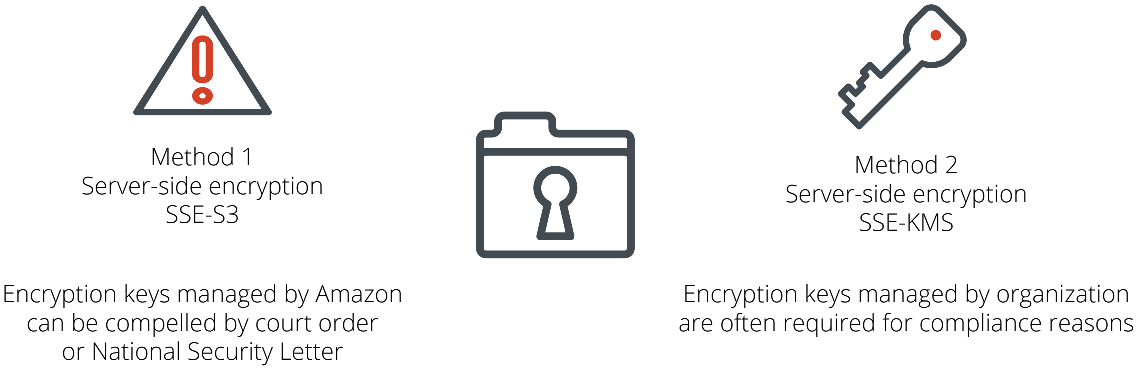
Figure 4: Illustrates how to keep expenses down by migrating the log files and deleting them.

ENCRYPTION

Encryption is the quickest path to data protection for enterprises and consumers. It's not always straightforward to configure and use, but it is available and becoming a base security standard in most circles. S3 buckets have this capability as well, and administrators can configure them to automatically encrypt all stored objects. The decision whether to do so rests on the contents of the buckets and the overhead costs associated with deploying encryption. Any bucket that stores sensitive information that, if publicly disclosed, would cause significant financial or legal issues, should be protected with server-side encryption.

This type of encryption is also known as encryption of data at rest. In these cases, S3 encrypts data at the object level as it is written to disk in its data centers and decrypted when accessed by a user who has permission to access the contents.

Amazon provides two methods for server-side encryption: SSE-S3 and SSE-KMS. In SSE-S3, Amazon manages the encryption keys. With SSE-KMS, the organization manages the encryption keys through the Key Management Service (KMS).



Method 1
Server-side encryption
SSE-S3

Method 2
Server-side encryption
SSE-KMS

Encryption keys managed by Amazon
can be compelled by court order
or National Security Letter

Encryption keys managed by organization
are often required for compliance reasons

Figure 5: Illustration of the two methods of server-side encryption provided by Amazon.

The end result is the same, but some organizations prefer to manage their own encryption keys or are required to do so for compliance reasons. In cases where Amazon manages the encryption keys, it could be compelled by a court order or National Security Letter to hand over content and objects stored in a S3 bucket, a happenstance some organizations should consider.

Administrators may also choose to implement client-side encryption. In these cases, objects are encrypted before they're uploaded to Amazon S3 and need to be decrypted after they are downloaded from a bucket.

AUDITING

Auditing is the final key component to a secure S3 environment. Users, roles, access requirements, and stored objects all change over time. Without a defined and documented process to manage and audit S3 buckets over time, chances are that a secure environment will eventually degrade into an insecure environment.

Audit S3 access permissions for all buckets on a regular basis, ideally every three months. For each bucket:

- Determine the user accounts who have access and what level of access they have
- Verify that access permissions follow the principle of least privilege (i.e., that users do not have more access than necessary)
- Include managers in this process to establish accountability and make sure they sign off on the level of permissions that their subordinates require.

While auditing S3 bucket access, verify the level of sensitivity of the objects stored in the bucket. A bucket with objects that are deemed low sensitivity might not have access logging or encryption enabled, but if users have started storing sensitive data in this bucket, the bucket configuration probably needs to change.

CONCLUSION

Amazon S3's beauty lies in Simple Storage Service. That's exactly what it is and what it provides to organizations of all sizes who are looking for a cloud-based storage solution. It lives up to its promises of cost-savings, efficiency, and convenience, but given that enterprises are likely to store sensitive data in S3, it can also be a central point of failure. Even better-resourced organizations may be undone by a configuration mistake, or a careless partner who fails to contain access to objects stored in a bucket.

Amazon recommends a number of security controls, including access control lists for buckets and objects, along with versioning, encryption, and more. At the end of the day, securing data in the cloud follows the same script as keeping data safe that's stored on premises:

Restrict access as much as possible

Implement logging and monitoring

Encrypt sensitive data

Audit access regularly

Most organizations have these processes documented for local storage but forget to implement similar processes for their cloud-stored data. This leads to insecurely configured buckets that allow more access than necessary and that do not take advantage of available security options. Addressing any such shortcomings is a huge step towards creating and maintaining secure S3 environments.

RESOURCES

Chicago voter roll - https://app.chicagoelections.com/documents/general/CBOE-Statement-Re-ESS-Voter-Data-2017-08-17.pdf
Accenture - https://www.upguard.com/breaches/cloud-leak-accenture
Verizon - https://kromtech.com/blog/security-center/verizon-wireless-employee-exposed-confidential-data-online
Military leak - https://www.upguard.com/breaches/cloud-leak-inscom
Access control stuff - https://labs.detectify.com/2017/07/13/a-deep-dive-into-aws-s3-access-controls-taking-full-control-over-your-assets/?utm_source=blog&utm_campaign=s3_buckets
AWS identity management guide - https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
AWS encryption - https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html
AWS logging - https://aws.amazon.com/answers/logging/centralized-logging

ABOUT THE AUTHOR

Gerben Kleijn is a Senior Security Analyst at Bishop Fox, a security consulting firm providing services to the Fortune 500, global financial institutions, and high-tech startups. In this role, Gerben focuses on compliance gap assessments, cloud deployment reviews, as well as firewall and VPN reviews. He also has significant experience with security monitoring and alerting. Gerben has worked on both the offensive and defensive side of security. Notable projects include securing information from the Dark Web and conducting gap assessments against the Critical Security Controls (CSC) Top 20 in addition to ISO 27001 and the National Institute of Standards and Technology (NIST) Cybersecurity Framework.

Gerben Kleijn, Senior Security Analyst