WHITE PAPER

Cloud-Native Ransomware –

How attacks on availability leverage cloud services

by Kat Traxler, Principal Security Researcher – Public Cloud

VECTRA®
SECURITY THAT THINKS

TABLE OF CONTENTS

How Ransomware affects cloud-hosted enterprise data

Ransomware is a financially motivated crime with the goal of inhibiting business systems and obtaining a ransom payment. Historically, ransoming data residing in traditional on-premises enterprise workloads and government systems have resulted in ample financial gain for assailants using ransomware attacks. With the expanding cloud footprint of modern digital systems, organizations are now trying to determine if ransomware can affect cloud-based workloads to the same degree, and further pondering "will there be evolutionary pressure on attackers which forces them to evolve their tactics."

With recent observations of trends in cloud adoption and data migration, my conclusion is such: I do not see how ransomware COULD NOT become a larger problem for global business.

My thesis on this subject can be summarized simply as: Wherever critical data lives, ransomware will go. When business data resides in the Cloud, rather than, say, in an on-premises database, it makes financial sense for attackers to evolve their tactics to target cloud systems with the same objectives as on-prem systems.

This paper serves to outline paths a malicious actor in the cloud might take to affect the availability of data by using the tools provided by the Cloud Service Provider (CSP). In addition to attacker behaviors, I have outlined proactive steps to secure cloud APIs which provide cryptographic services, architectural patterns to make securing these systems easier and methods for detecting cloud-native ransomware.

Attacking Availability in the Cloud

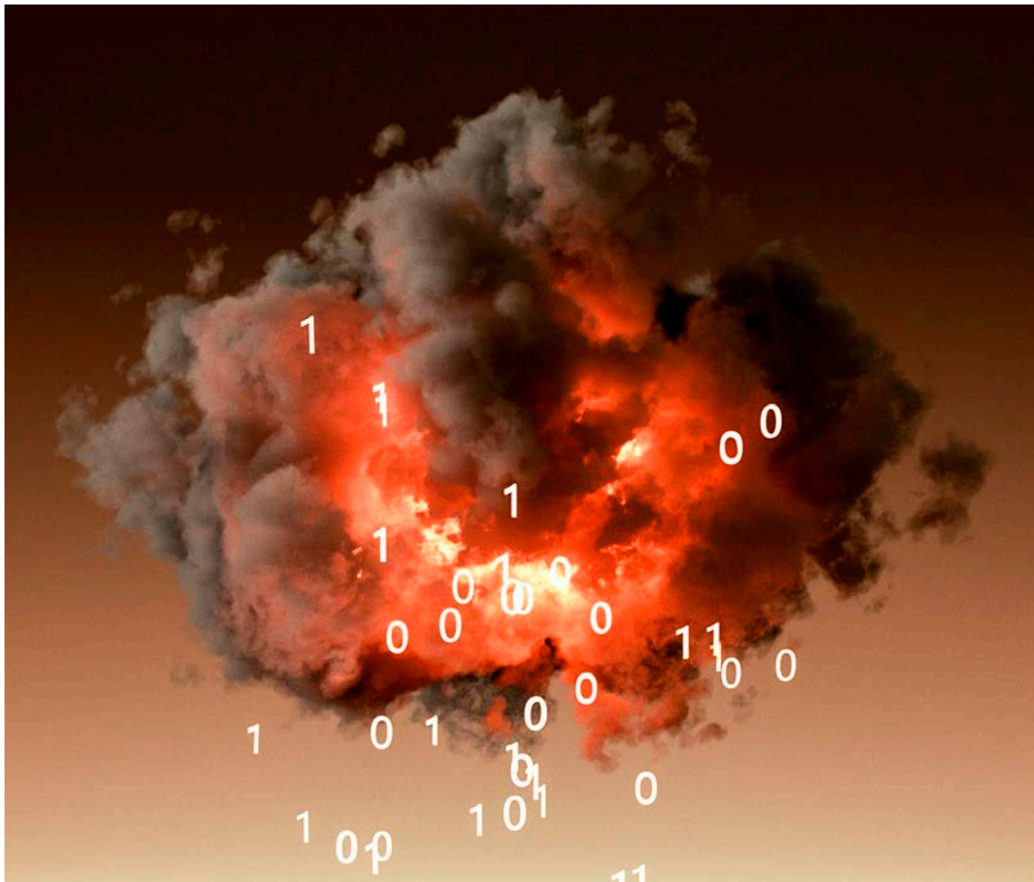
The first 10 plus years of cloud transformations have seen an acceleration of cloud migrations and the depositing of increasingly large datasets in the cloud. Even as we witness this shift, we often continue to think about ransomware solely in the context of on-premises environments, naively extending those concepts to the cloud.

In the cloud, the available tools developers and customers need to accomplish everyday tasks are built-in and offered as features by the cloud service providers. With access, the same tools and capabilities are often used by bad actors to overcome security controls, avoid detection and to accomplish specific goals. Using these features with ill intent is often referred to as *feature abuse*.

The openness of cloud-native services via APIs makes it easy for attackers to abuse, or rather misuse tools for the same.

The openness of cloud-native services via APIs makes it easy for attackers to abuse, or rather misuse tools for the same. APIs, created by each CSP are highly discoverable, and can be quickly understood and leveraged in unintended ways. *Feature abuse* presents a risk in addition to code vulnerabilities. Unlike the exploitation of a vulnerability, there is no existence of a code flaw to manipulate that may be detected by pattern matching or secured via patching. Instead, threat actors are leveraging the tools provided by the CSP used for deploying or maintaining production software and infrastructure to accomplish nefarious tasks.

Whatever environment they find themselves in, threat actors will leverage the tools available at their disposal to accomplish their nefarious tasks. In a sense, by abusing public APIs, ransomware in the cloud will continue the trend of "Living off the Land" where the "LOL Binaries" of the Cloud are its APIs, feature rich and public. In contrast to Windows executables which can be uninstalled as superfluous software, AWS cloud APIs are always on. Exposure, access, and availability continue to provide the openness services require and give way to the means for devastating ransomware attacks.



Where is your Cloud Data hosted?

To understand abuse techniques ransomware operators might use, one must first understand the systems where data is stored. On-premises data is likely to be warehoused across various technologies, from Oracle Databases to Microsoft SQL Servers. What these systems have in common is that they are physical hosts, fully under your control.

On-premises data warehouse servers are commonly physical hosts implemented in a full-stack environment which provide easy targets for malware due to the broad attack surface area. However, full-stack environments benefit from being secured by robust data protection strategies, employing security controls that have evolved over the past 20 years of modeling network-based threats. As such, traditional on-premises databases are tucked behind layers of network controls, corralled off into the deep recesses of corporate networks, and heavily monitored with agent-based threat detection.

The on-premises approach to securing traditional data stores does not translate to the cloud, nor should it. Data migrated to the cloud resides in systems where all end-users, including malicious actors, have limited, curated access to the underlying system. This means cloud data stores have dramatically different attack surfaces.

Each of the major cloud service providers (e.g., AWS, AZURE, GCP) has a unique version of a distributed, highly available, all-purpose data store: AWS S3, Azure Blob Storage and GCP Storage Buckets. Each is a core repository for unstructured data and is a ubiquitous, stable, and highly available service that integrates with many other services on their respective platforms, meeting nearly any customers' data storage needs. Cloud providers utilize storage services to build pipelines, and they serve as the backing data store for big data platforms or as public repositories for web content.

If you are an AWS customer, it is hard NOT to use S3. Which makes it a likely target of cloud-native ransomware authors.

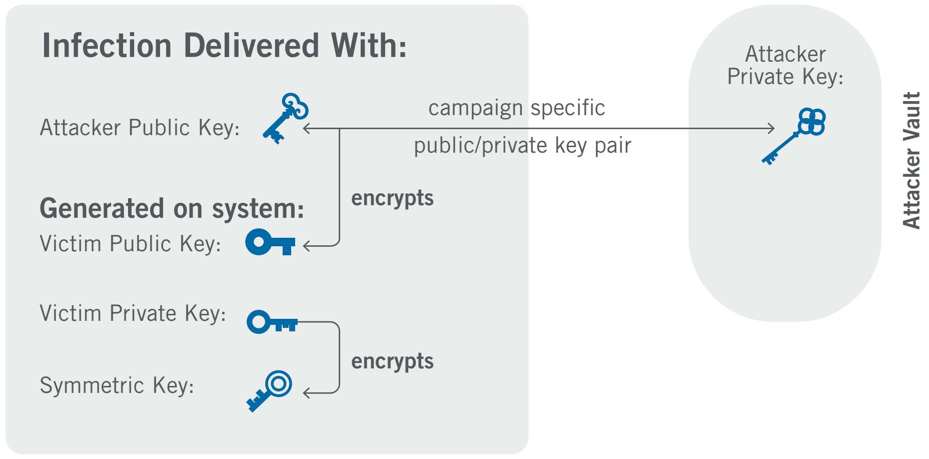
Traditional Ransomware Encryption Strategy

Ransomware targeting on-premises systems uses a hybrid encryption scheme, leveraging the best of both symmetric and asymmetric encryption, to work around the limitations of each[1].

Limitations being:

- Where asymmetric encryption operations are slow, encrypting data with a symmetric key is fast!
- Symmetric encryption uses the same key for both encryption and decryption. A purely symmetric strategy often leaves the decryption key on the system, making recovery easier by forensics teams.

The strategies employed by ransomware authors to work around these limitations are the same techniques in-house crypto teams use. Whether for benevolent or malevolent intent, key hierarchies can be used to derive one set of keys from another and then encrypt the symmetric data keys.



By blending symmetric and asymmetric encryption, ransomware authors can achieve a more complex set of goals.

- Increased speed when performing encryption tasks
  - > By targeting data with symmetric keys, ransomware can often complete exploitation before defensive capabilities have a chance to respond.
- Obfuscating the symmetric key material hinders recovery during post-exploitation analysis.
  - > By encrypting the symmetric data key with an asymmetric key, recovery of the key material is challenging if not impossible for forensics professionals.

The creation of malware to perform complex encryption techniques is necessary for the on-premises encrypters. However, this encryption strategy is likely to be cumbersome and completely unnecessary when targeting data in the cloud. Outlined in this paper are methods an attacker could use which leverage the tools offered by cloud service providers and make traditional ransomware techniques obsolete.

Encrypting S3 Objects with Attacker Controlled KMS Keys

Cloud Service Providers have done the heavy lifting to maintain secure roots of trust for their customers who utilize their cryptographic services. Attackers can piggyback on these conveniences to affect the availability of cloud-hosted data.

AWS Key Management Service (AWS KMS), allows its customers to generate keys, control access to those keys, and use them to perform cryptographic operations such as signing, verifying, and managing the envelop encryption process required for S3 Server-Side Encryption (SSE).

When not using KMS, encrypted objects on S3 are encrypted with an Amazon-Master key. When an authorized party requests access to an object in this bucket, AWS transparently decrypts the data in the background. Instead of allowing AWS to generate and manage the SSE backing keys, customers can use a KMS key and leverage the resource-based policy attached to the key as another layer of access control around the encrypted data. The ability to apply policy and restrict access to a key leaves an opening for attackers to affect the availability of data encrypted with the key.

Cloud Service Providers have done the heavy lifting to maintain secure roots of trust for their customers who utilize their cryptographic services.

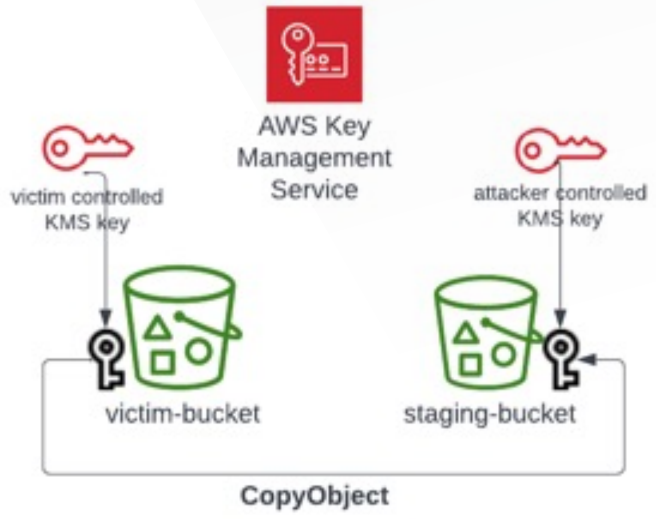
To demonstrate the role of KMS in SSE, note scenario one below, where a ransomware operator gains significant access to S3 and KMS in an AWS Account. The malicious actor can read, copy, and delete objects from S3 plus obtain the permissions to create a new KMS key. The scenario further describes how a threat actor could use access to affect data availability and demand ransom for its restoration.

Scenario: Demonstration of Cloud-Native Ransomware on S3

The scenario laid out has a threat actor targeting the data in the aptly named ‘victim-bucket’. On this bucket, Server-Side Encryption (SSE) is enabled, specifying that objects are encrypted transparently with an Amazon-managed Master Key (SSE-S3). An end-user who is only granted access to retrieve objects (action s3:GetObject) from this bucket would have sufficient permissions to download the cleartext of the stored objects. No additional permissions are required to decrypt the objects when encrypted with an Amazon-managed Master Key.

In this scenario, we consider that a malicious actor has compromised an end-user with the intent of holding the data for ransom. The malicious actor has created a new S3 bucket we’ll call ‘staging-bucket’, which they will use as a landing zone for targeted S3 data. The newly created ‘staging-bucket’ also requires uploaded objects to be encrypted but with a KMS key. We generated the KMS key in AWS, and stored it within an AWS HSM, then relied upon customer-managed policy to enforce access control on the key.

When migrating data from the ‘victim-bucket’ to the ‘staging-bucket,’ the objects are re-encrypted with the newly created KMS key. Permissions associated with the S3 objects and KMS key dictate access to the objects. Expect access denied responses where requests originate from callers that lack explicit permissions to access both the S3 objects and the KMS key. If an end-user has an explicit ‘deny’ permission on either the S3 object or the KMS key, expect access requests to also be denied.


AWS Key Management Service
victim controlled KMS key
attacker controlled KMS key
victim-bucket
staging-bucket
CopyObject

At this point, our fictional threat actor has set the stage for a ransomware attack by migrating the S3 objects to a new datastore and re-encrypted the objects with keys under their control. The next step in this narrative involves impacting access to the key for cryptographic operations, such as decryption.

Key Policy Lock-Out - "*DENY, except when; ALLOW, only when*"

The technique of locking out an AWS customer from a KMS key housed in their own account was first described by Spencer Geitzen at the Cloud Village at DEFCON in 2020[2].

Let's look at what a malicious update to key policy might look like.

- DENY – all actions on the KMS key - except when "aws:sourceIp" condition key equals the attacker-controlled IP.
- ALLOW – all actions on the KMS key — only when the caller is from the attacker-controlled account

This style of resource-based policy jargon can be referred to as: *"DENY, except when; ALLOW, only when"*.

You can imagine other conditions being leveraged by a ransomware operator looking to employ this pattern. Requiring the caller to originate from a specific source IP is in effect a mechanism to restrict all activity on a key, however just as effective could be the use of the following AWS global key conditions:

- aws:PrincipalArn
- aws:PrincipalAccount
- aws:sourceVPC
- aws:SourceVPCe

Any condition where a value is required by the caller that is unique and attacker-controlled can be leveraged to lockout an AWS customer from their resources.

When updating a resource-based policy attached to the KMS key to the "*DENY, except when; ALLOW, only when*" pattern, the victim account is effectively locked out of their newly encrypted data. Even the root user is unable to access the encrypted S3 data.

Only callers originating from the attacker-controlled AWS Account from the attacker-controlled IP would be able to access the KMS Key and decrypt the S3 data.

The final cleanup from a cloud-native ransomware attack would include deleting the original 'victim-bucket' and uploading a ransom note to a new, unencrypted bucket.

Existing KMS Keys

Locking out a victim from a KMS key is not the only way to affect the availability of S3. However, it is one of the more interesting mechanisms for a Security Researcher to model.

This technique is not limited to new KMS keys. To affect availability by impacting an existing KMS key would require even sparser permissions from the threat actor. Updating an existing KMS key policy to restrict access to it for cryptographic operations would have the same debilitating effect on data availability as would re-encrypting S3 data with a new, attacker-created key.

In both previous scenarios, access to the symmetric key used in server-side encryption (SSE-KMS) is held hostage by malicious actors through manipulation of the key policy.

Any condition where a value is required by the caller that is unique and attacker-controlled can be leveraged to lockout an AWS customer from their resources.

When The Victim Pays Up

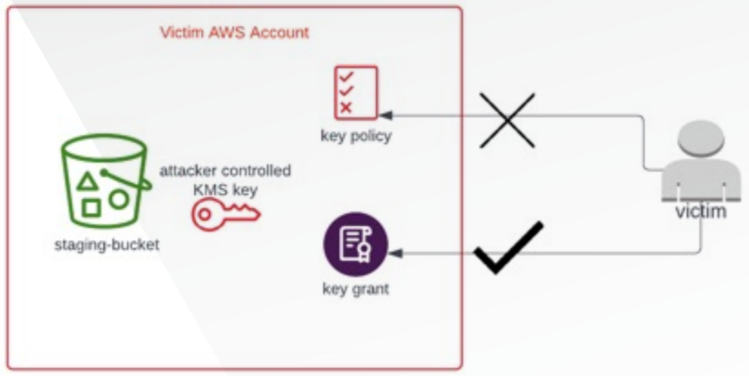
We cannot presume to know what it would look like for a cloud-native ransomware gang to relinquish control of the data decryption key based on a traditional on-premises ransomware.

In the cloud, the process of "turning over the keys" to encrypted data will look different even as it remains a necessary component to ransomware's lifecycle. After all, the product ransomware delivers to its consumer is a mechanism to recover encrypted data. Like any other business, ransomware operators need a reliable and trusted means to deliver products to their 'customers.'

A real-time, reliable mechanism for returning control to the victim cannot involve an update to key policy.

In scenario one, only callers from the attacker account can access the KMS Key which are needed to decrypt business-critical data, but updating the key policy is an action that an attacker cannot be perform cross-account. So, a real-time, reliable mechanism for returning control to the victim cannot involve an update to key policy. Rather, a ransomware gang may turn their sights to key grants, an alternative access-control mechanism for KMS keys used to delegate permissions for cryptographic operations.

A KMS key grant[3] is a delegation of permissions to a grantee which returns a token used to perform cryptographic operations on that key. By creating a key grant and allowing the victim account to use the hijacked key for decryption, the ransomware gang can effectively return availability to its paying 'customers', bypassing restrictions imposed by the constrained key policy. A KMS key grant would allow the victim to access the KSM key and begin the process of recovering their encrypted data.



Could AWS save you from a Cloud-Native Ransomware Attack?

Given the previous insights, when faced with a Ransomware Attack in a cloud-native setting, it is reasonable to ponder where the "shared responsibility model" factors in. Below are a few considerations worth examining.

The first scenario for an AWS intervention hypothesizes that AWS could access KMS key material from an AWS HSM. This conjecture is so implausible, that it feels completely outside of the realm of possibility. It is unthinkable that AWS could retrieve a KMS key from one of their HSMs housed in their data centers. AWS has gone to great lengths to ensure that no person could retrieve key material and make earnest public statements to that effect.

The remaining two scenarios in any AWS intervention are much more of an open question. The first avenue for assistance would involve AWS changing the victim's environment. There is no evidence that AWS can reverse a maliciously applied key policy in a victim account, restoring access to a KMS Key. There is no indication they have any '*hand-of-god*' capabilities over resource-based policies. There also is not much incentive to publicly admit the ability if they did.

The final assisted remediation option to consider would have AWS commandeer the account in which the malicious actors are operating. The Trust and Safety team at AWS will quarantine and close rogue accounts which they have found to be violating terms of service, such as those used in Botnet campaigns. However, quarantining and closing accounts is much different from seizing control over the resources, which would be required to regain control over a hijacking KMS key. While it is not clear whether AWS has a process to take over accounts, there is some evidence to suggest that they do.

A mechanism exists at AWS to change the Root email address for an account. You would see this in action if you ever forgot the password for your Root User and lost access to the email associated with the Root User. AWS Support will require you to provide a notarized attestation of your account ownership before innating a change to the underlying root email account. This internal process suggests AWS has the power to seize control over accounts, not simply close them but access the resources housed in them with administrative capabilities.

Like the '*hand-of-god*' theory, AWS has little incentive to publicize an ability to confiscate accounts. From a defenders' point of view, without a clearly documented process for assisted recovery from AWS with guaranteed Service-Level Agreements, theoretical remediation assistance from AWS is not useful. Planning for response and recovery from a ransomware event should not be centered on help from AWS and should not be expected.

Planning for response and recovery from a ransomware event should not be centered on help from AWS and should not be expected.

Prevention, Detection, and Response to Cloud-Native Ransomware

Rooted in the understanding that it is unlikely AWS could help during a ransomware event, we turn our sights to the bread-and-butter of all security programs, preventive controls, and detection and response capabilities.

SCP Restricting KMS Keys

Mitigations via Service Control Policy (SCP) require a certain degree of maturity in your cloud security program, but that does not mean their use should not be an operational goal. Creating a bespoke policy for KMS keys requires understanding which keys should be authorized to perform cryptographic operations on your data and who should have access to them.

A Service Control Policy (SCP) naming the specific KMS keys allowed to encrypt objects could be a good start in preventing a cloud-native ransomware attack of the type described in this paper. Restricting cryptographic operations to specific keys would prevent an attacker from maliciously encrypting S3 objects with either a newly created key or an external key of their control, but not hijacking the policy of an existing, approved key.

Often this type of SCP is coupled with an architectural pattern that corrals all KMS keys into a single account. It should be a preferred design pattern if not strictly for ransomware resiliency but also for all the benefits centralization brings such as auditability and simplified key rotation.

Using this centralized "Fort Knox" approach to Key Management creates the 'all your eggs in one basket' design. It also allows for the 'all your security controls in one basket' approach as well. A central Key Management account allows a security organization to enforce the principal of least privilege in the strictest sense, establish a baseline for normal key usage patterns, and monitor for abnormal transactions.

S3 Bucket Configurations

Not all S3 buckets can be configured to be fortresses but it is worth noting the controls available in S3 which can add to the overall ransomware resiliency strategy.

- **Object Lock**: A configuration applied to an S3 bucket which prevents the deletion of an object or its version.
- **Object Versioning**: This setting will force the creation of new objects, instead of overwriting previously uploaded objects. When combined with 'Object Lock', these settings could prevent objects from being overwritten with maliciously encrypted versions. Whenever versioning is enabled, be sure to set policies to manage the lifecycle of your versioned objects.
- **MFA Delete**: Ensures that the MFA bit is set on the callers' session token when attempting to delete an object from S3.

Again, these controls are likely to be incompatible with highly transactional data but could be feasible when looking to secure your sensitive backups. Knowing what data you have and where it lives is a prerequisite for enforcing any of these Bucket-level mitigations.

A central Key Management account allows a security organization to enforce the principal of least privilege in the strictest sense, establish a baseline for normal key usage patterns, and monitor for abnormal transactions.

Detecting Cloud-Native Ransomware

This paper outlines two possibilities a threat actor might use to impact the availability of data hosted on S3. Our fictional threat actor used a key policy lock-out method to impact the availability of a new KMS key. Also recognized is the possibility of using the same modus operandi on an existing key. Or maliciously encrypting S3 data with an external KMS key, one housed in the attacker-controlled account. Let us look at each of the scenarios to discuss what events in your CloudTrail might warrant an investigation from your responders.

Key-policy lockout with new key – If a threat actor employs this technique, there are two critical points that could be alerted on and responded to during the exploitation phase, when observing the use of a new KMS key and ingesting the events surrounding the update to the key policy. If your organization has a clear view of which keys should be used for encryption, the use of non-approved KMS keys should raise alarms. Additionally, updating key policy to include one of the global condition keys mentioned in this paper should also warrant a follow through.

Key-policy lockout with existing key – A threat actor may choose to focus their efforts on impacting an existing KMS key. This leaves limited opportunity for detection during the exploitation phase. Still, custom alerts can be crafted to notify when the key policy is updated to include one of the global condition keys mentioned in this paper which might signify a cloud-native ransomware attack has occurred.

Encryption with External KMS Key – While this scenario is not detailed in this paper it remains a viable mechanism for the malicious encryption of data. Security operations teams would want to be notified when encryption operations are performed with KMS keys which are not housed in an account under their control.

In Conclusion

Cloud Service Providers make available cryptographic tools which, if not properly secured, can be leveraged by ransomware gangs to affect the availability of data in the cloud. A successful ransomware campaign in the cloud uses cloud-native services to encrypt data with speed and built-in access control mechanisms to lock out the victim from critical business data.

Drafting centralized architectural patterns for key management is the first step to preventing cloud-native ransomware. Centralized key management allows for both more effective prevention of ransomware and a more consistent picture of what a normal cryptographic pattern in your cloud estate looks like.

While a preventive posture is ideal, it is unlikely the majority of organizations have these architectural controls in place on day one. Additionally, it is incumbent on every organization, even those with highly restrictive environments, to plan for the day when their guardrails fail. As such, detecting if cloud-hosted data is impacted by ransomware should be the overarching priority. These detection strategies will subtlety vary depending on the technique used by the threat actor but are rooted in concept of knowing what external means to your cloud estate, whether that be in monitoring for external access granting, external key usage or conditional statements in policy.

References:
[1] Ransomware encryption techniques: https://medium.com/@tarcisioma/ransomware-encryption-techniques-696531d07bb9
[2] Ransom in the Cloud - Spencer Gietzen (DEF CON Cloud Village): https://www.youtube.com/watch?v=8QdZ2-sAQFs&list=PL5944c_fOMYn2cQQuQe23gtqZfHWzyrPn&index=3
[3] Grants in AWS KMS: https://docs.aws.amazon.com/kms/latest/developerguide/grants.html
S3 Ransomware Part 1: Attack Vector: https://rhinosecuritylabs.com/aws/s3-ransomware-part-1-attack-vector/
S3 Ransomware Part 2: Prevention and Defense: https://rhinosecuritylabs.com/aws/s3-ransomware-part-2-prevention-and-defense/

Email info@vectra.ai   vectra.ai