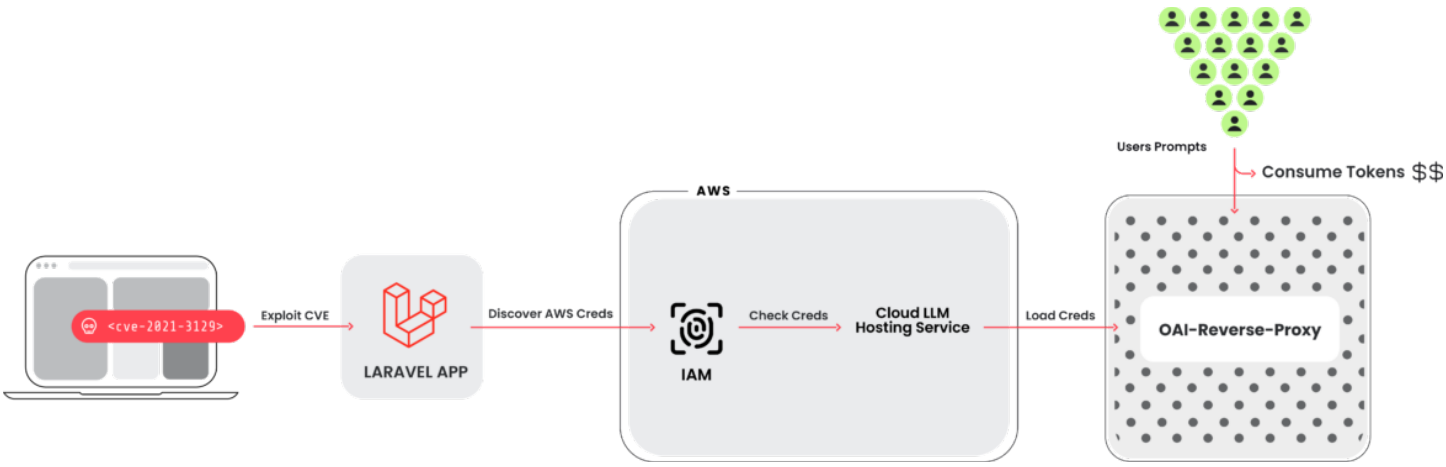
LLMjacking: Stolen Cloud Credentials Used in New AI Attack

Alessandro Brucato

The Sysdig Threat Research Team (TRT) recently observed a new attack that leveraged stolen cloud credentials in order to target ten cloud-hosted large language model (LLM) services, known as LLMjacking. The credentials were obtained from a popular target, a system running a vulnerable version of Laravel (CVE-2021-3129). Attacks against LLM-based Artificial Intelligence (AI) systems have been discussed often, but mostly around prompt abuse and altering training data. In this case, attackers intend to sell LLM access to other cybercriminals while the cloud account owner pays the bill.

Once initial access was obtained, they exfiltrated cloud credentials and gained access to the cloud environment, where they attempted to access local LLM models hosted by cloud providers: in this instance, a local Claude (v2/v3) LLM model from Anthropic was targeted. If undiscovered, this type of attack could result in over $46,000 of LLM consumption costs per day for the victim.

Sysdig researchers discovered evidence of a reverse proxy for LLMs being used to provide access to the compromised accounts, suggesting a financial motivation. However, another possible motivation is to extract LLM training data.



Breadth of Targets

We were able to discover the tools that were generating the requests used to invoke the models during the attack. This revealed a broader script that was able to check credentials for ten different AI services in order to determine which were useful for their purposes. These services include:

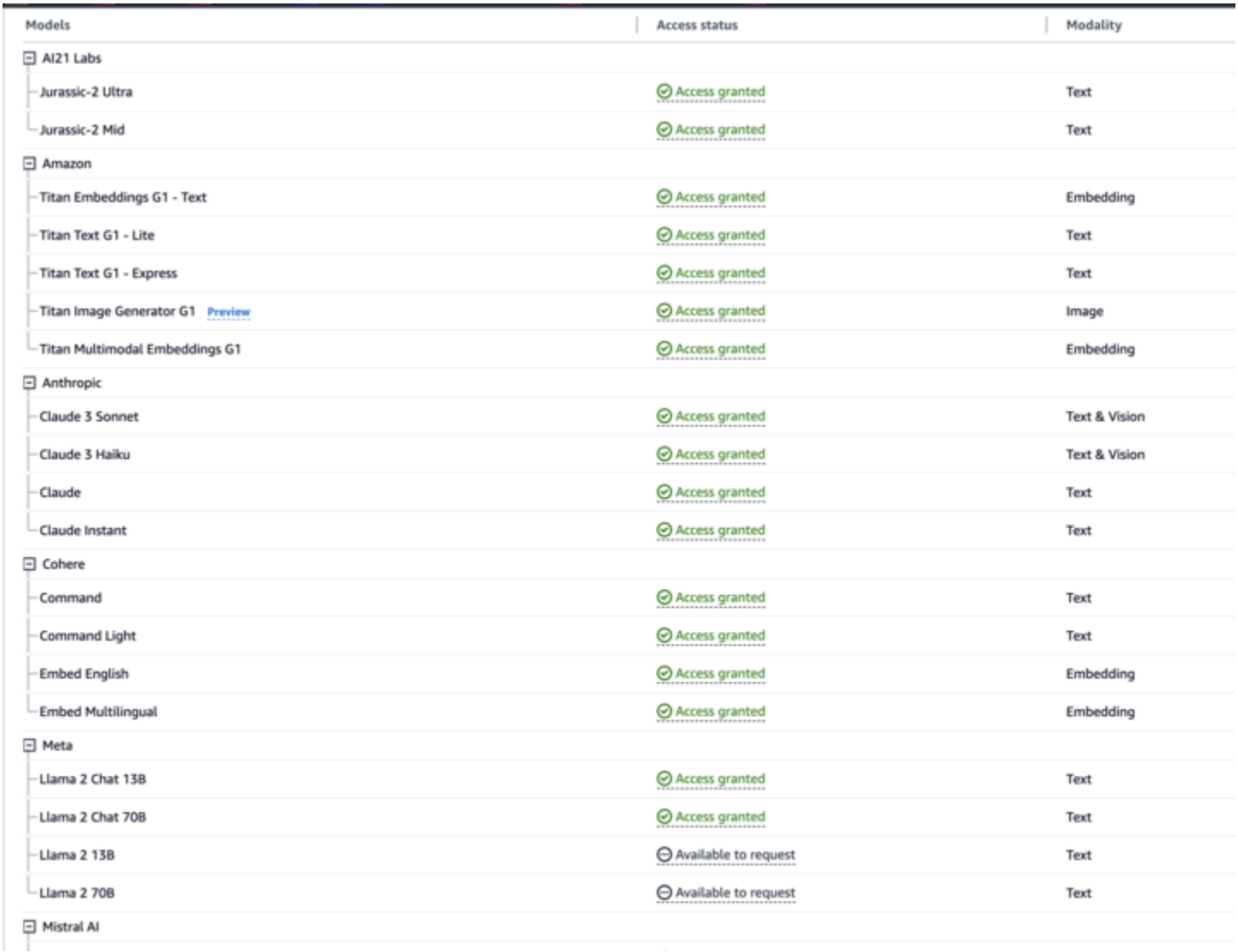
AI21 Labs, Anthropic, AWS Bedrock, Azure, ElevenLabs, MakerSuite, Mistral, OpenAI, OpenRouter, and GCP Vertex AI

The attackers are looking to gain access to a large amount of LLM models across different services. No legitimate LLM queries were actually run during the verification phase. Instead, just enough was done to figure out what the credentials were capable of and any quotas. **In addition, logging settings are also queried where possible. This is done to avoid detection when using the compromised credentials to run their prompts.**

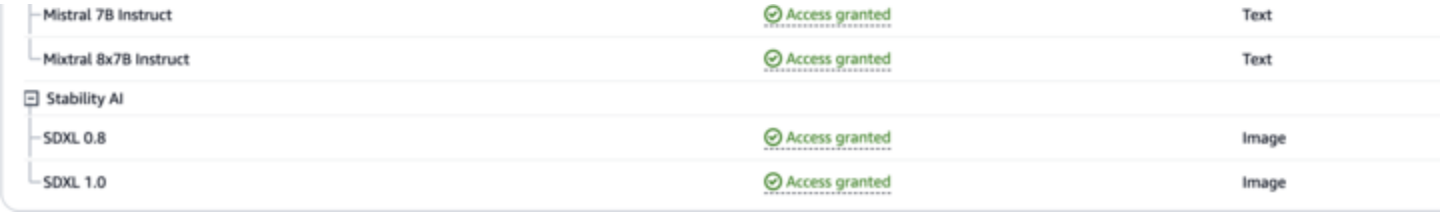
Background

Hosted LLM Models

All major cloud providers, including Azure Machine Learning, GCP's Vertex AI, and AWS Bedrock, now host large language model (LLM) services. These platforms provide developers with easy access to various popular models used in LLM-based AI. As illustrated in the screenshot below, the user interface is designed for simplicity, enabling developers to start building applications quickly.



Models	Access status	Modality
AI21 Labs		
Jurassic-2 Ultra	Access granted	Text
Jurassic-2 Mid	Access granted	Text
Amazon		
Titan Embeddings G1 - Text	Access granted	Embedding
Titan Text G1 - Lite	Access granted	Text
Titan Text G1 - Express	Access granted	Text
Titan Image Generator G1 Preview	Access granted	Image
Titan Multimodal Embeddings G1	Access granted	Embedding
Anthropic		
Claude 3 Sonnet	Access granted	Text & Vision
Claude 3 Haiku	Access granted	Text & Vision
Claude	Access granted	Text
Claude Instant	Access granted	Text
Cohere		
Command	Access granted	Text
Command Light	Access granted	Text
Embed English	Access granted	Embedding
Embed Multilingual	Access granted	Embedding
Meta		
Llama 2 Chat 13B	Access granted	Text
Llama 2 Chat 70B	Access granted	Text
Llama 2 13B	Available to request	Text
Llama 2 70B	Available to request	Text
Mistral AI		



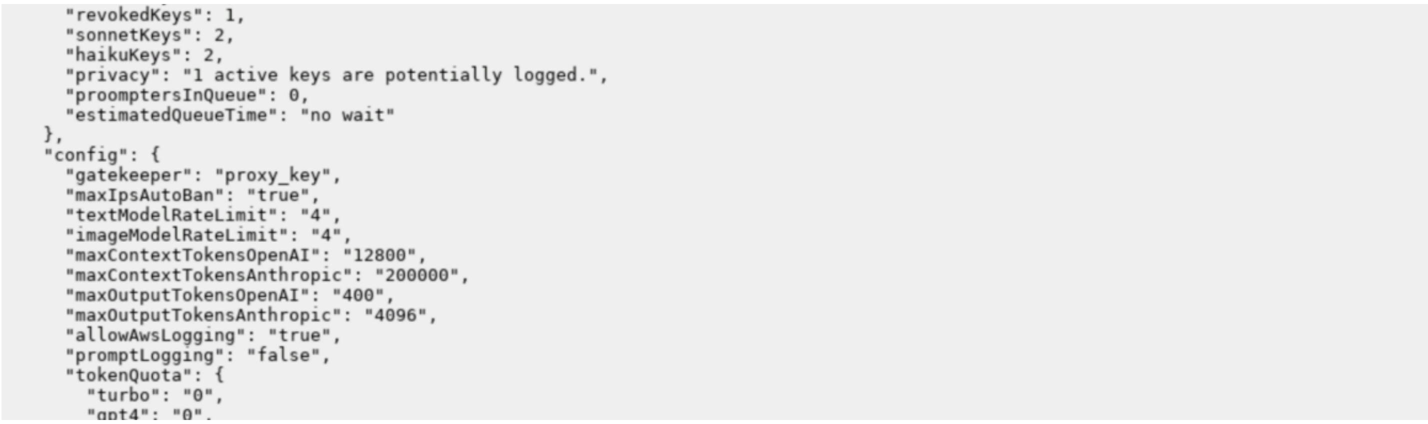
These models, however, are not enabled by default. Instead, a request needs to be submitted to the cloud vendor in order to run them. For some models, it is an automatic approval; for others, like third-party models, a small form must be filled out. Once a request is made, the cloud vendor usually enables access pretty quickly. The requirement to make a request is often more of a speed bump for attackers rather than a blocker, and shouldn't be considered a security mechanism.

Cloud vendors have simplified the process of interacting with hosted cloud-based language models by using straightforward CLI commands. Once the necessary configurations and permissions are in place, you can easily engage with the model using a command similar to this:

> *aws bedrock-runtime invoke-model –model-id anthropic.claude-v2 –body '{"prompt": "\n\nHuman: story of two dogs\n\nAssistant:", "max_tokens_to_sample" : 300}' –cli-binary-format raw-in-base64-out invoke-model-output.txt*

LLM Reverse Proxy

The key checking code that verifies if credentials are able to use targeted LLMs also makes reference to another project: [OAI Reverse Proxy](). This open source project acts as a reverse proxy for LLM services. Using software such as this would allow an attacker to centrally manage access to multiple LLM accounts while not exposing the underlying credentials, or in this case, the underlying pool of compromised credentials. During the attack using the compromised cloud credentials, a user-agent that matches OAI Reverse Proxy was seen attempting to use LLM models.



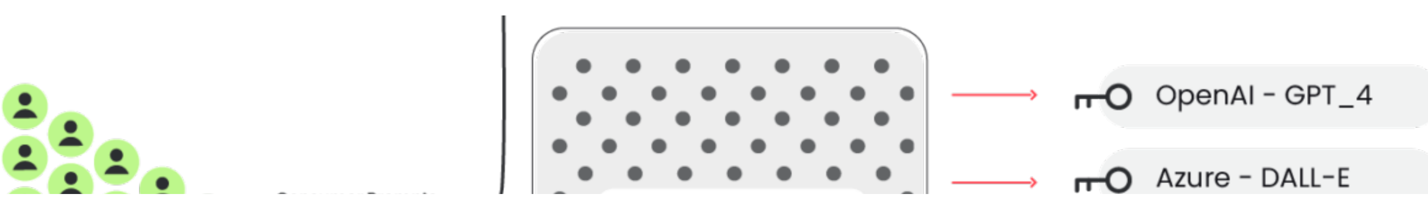
```
    "revokedKeys": 1,
    "sonnetKeys": 2,
    "haikuKeys": 2,
    "privacy": "1 active keys are potentially logged.",
    "proomptersInQueue": 0,
    "estimatedQueueTime": "no wait"
  },
  "config": {
    "gatekeeper": "proxy_key",
    "maxIpsAutoBan": "true",
    "textModelRateLimit": "4",
    "imageModelRateLimit": "4",
    "maxContextTokensOpenAI": "12800",
    "maxContextTokensAnthropic": "200000",
    "maxOutputTokensOpenAI": "400",
    "maxOutputTokensAnthropic": "4096",
    "allowAwsLogging": "true",
    "promptLogging": "false",
    "tokenQuota": {
      "turbo": "0",
      "gpt4": "0",
```

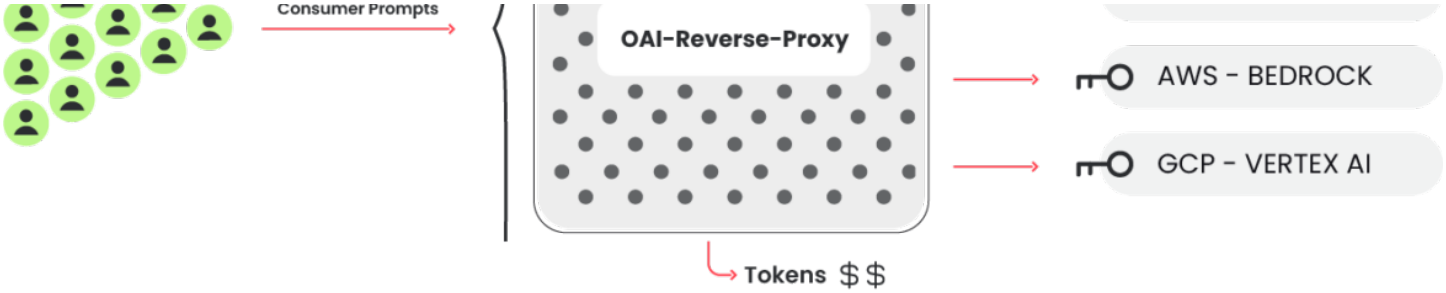

The image above is an example of an OAI Reverse Proxy we found running on the Internet. There is no evidence that this instance is tied to this attack in any way, but it does show the kind of information it collects and displays. Of special note are the token counts (“tookens”), costs, and keys which are potentially logging.


```
{
  "uptime": 1485257,
  "endpoints": {
    "openai": "htt[redacted]/proxy/openai",
    "openai2": "htt[redacted]/proxy/openai/turbo-instruct",
    "anthropic": "htt[redacted]/proxy/anthropic",
    "anthropic-sonnet (⚠Temporary: for Claude 3 Sonnet)": "htt[redacted]/proxy/anthropic/sonnet",
    "anthropic-opus (⚠Temporary: for Claude 3 Opus)": "htt[redacted]/proxy/anthropic/opus",
    "aws": "htt[redacted]/proxy/aws/claude",
    "aws-sonnet (⚠Temporary: for AWS Claude 3 Sonnet)": "htt[redacted]/proxy/aws/claude/sonnet"
  },
  "proompts": 6141,
  "tookens": "69.34m ($1046.80)",
  "proomptersNow": 1,
  "openaiKeys": 3,
  "openaiOrgs": 3,
  "anthropicKeys": 2,
  "awsKeys": 1,
  "turbo": {
    "usage": "0 tokens ($0.00)",
    "activeKeys": 2,
    "revokedKeys": 0,
    "overQuotaKeys": 1,
    "trialKeys": 0,
    "proomptersInQueue": 0,
    "estimatedQueueTime": "no wait"
  },
  "gpt4-turbo": {
    "usage": "165.4k tokens ($1.65)",
    "activeKeys": 2,
    "overQuotaKeys": 1,
    "proomptersInQueue": 0,
    "estimatedQueueTime": "no wait"
  },
  "gpt4": {
    "usage": "0 tokens ($0.00)",
```


This example shows an OAI reverse proxy instance, which is setup to use multiple types of LLMs. There is no evidence that this instance is involved with the attack.

If the attackers were gathering an inventory of useful credentials and wanted to sell access to the available LLM models, a reverse proxy like this could allow them to monetize their efforts.





Technical Analysis

In this technical breakdown, we explore how the attackers navigated a cloud environment to carry out their intrusion. By employing seemingly legitimate API requests within the cloud environment, they cleverly tested the boundaries of their access without immediately triggering alarms. The example below demonstrates a strategic use of the InvokeModel API call logged by CloudTrail. Although the attackers issued a valid request, they intentionally set the max_tokens_to_sample parameter to -1. This unusual parameter, typically expected to trigger an error, instead served a dual purpose. It confirmed not only the existence of access to the LLMs but also that these services were active, as indicated by the resulting ValidationException. A different outcome, such as an AccessDenied error, would have suggested restricted access. This subtle probing reveals a calculated approach to uncover what actions their stolen credentials permitted within the cloud account.

InvokeModel

The InvokeModel call is logged by CloudTrail and an example malicious event can be seen below. They sent a legitimate request but specified "`max_tokens_to_sample`" to be -1. This is an invalid error which causes the "ValidationException" error, but it is useful information for the attacker to have because it tells them the credentials have access to the LLMs and they have been enabled. Otherwise, they would have received an "AccessDenied" error.

```
{

    "eventVersion": "1.09",

    "userIdentity": {

        "type": "IAMUser",

        "principalId": "[REDACTED]",

        "arn": "[REDACTED]",
```

```
        "accountId": "[REDACTED]",
        "accessKeyId": "[REDACTED]",
        "userName": "[REDACTED]"
    },
    "eventTime": "[REDACTED]",
    "eventSource": "bedrock.amazonaws.com",
    "eventName": "InvokeModel",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "83.7.139.184",
    "userAgent": "Boto3/1.29.7 md/Botocore#1.32.7 ua/2.0 os/windows#10 md/
arch#amd64 lang/python#3.12.1 md/pyimpl#CPython cfg/retry-mode#legacy
Botocore/1.32.7",
    "errorCode": "ValidationException",
    "errorMessage": "max_tokens_to_sample: range: 1..1,000,000",
    "requestParameters": {
        "modelId": "anthropic.claude-v2"
    },
    "responseElements": null,
    "requestID": "d4dced7e-25c8-4e8e-a893-38c61e888d91",
```

```
    "eventID": "419e15ca-2097-4190-a233-678415ed9a4f",

    "readOnly": true,

    "eventType": "AwsApiCall",

    "managementEvent": true,

    "recipientAccountId": "[REDACTED]",

    "eventCategory": "Management",

    "tlsDetails": {

        "tlsVersion": "TLSv1.3",

        "cipherSuite": "TLS_AES_128_GCM_SHA256",

        "clientProvidedHostHeader": "bedrock-runtime.us-
east-1.amazonaws.com"

    }

}
```

Code language: Perl (perl)

Example Cloudtrail log

AWS Bedrock is not supported in all regions so the attackers called "*InvokeModel*" only in the supported regions. At this time, Bedrock is supported in us-east-1, us-west-2, ap-southeast-1, ap-northeast-1, eu-central-1, eu-west-3, and us-gov-west-1, as shown here. Different models are available depending on the region; here is the list of models supported by AWS Region.

GetModelInvocationLoggingConfiguration

Interestingly, the attackers showed interest in how the service was configured. This can be done by calling *"GetModelInvocationLoggingConfiguration,"* which returns S3 and Cloudwatch logging configuration if enabled. In our setup, we used both S3 and Cloudwatch to gather as much data about the attack as possible.

```
{
    "loggingConfig": {
        "cloudWatchConfig": {
            "logGroupName": "[REDACTED]",
            "roleArn": "[REDACTED]",
            "largeDataDeliveryS3Config": {
                "bucketName": "[REDACTED]",
                "keyPrefix": "[REDACTED]"
            }
        },
        "s3Config": {
            "bucketName": "[REDACTED]",
            "keyPrefix": ""
        },
        "textDataDeliveryEnabled": true,
        "imageDataDeliveryEnabled": true,
        "embeddingDataDeliveryEnabled": true
    }
}
```

Code language: Perl (perl)

Example GetModelInvocationLoggingConfiguration response

Information about the prompts being run and their results are not stored in Cloudtrail. Instead, additional configuration needs to be done to send that information to Cloudwatch and S3. This check is done to hide the details of their activities from any detailed observations. OAI Reverse Proxy states it will not use any AWS key that has logging enabled for the sake of "privacy." This makes it impossible to inspect the prompts and responses if they are using the AWS Bedrock vector.

Impact

In an LLMjacking attack, the damage comes in the form of increased costs to the victim. It shouldn't be surprising to learn that using an LLM isn't cheap and that cost can add up very quickly. Considering the worst-case scenario where an attacker abuses Anthropic Claude 2.x and reaches the quota limit in multiple regions, **the cost to the victim can be over $46,000 per day**.

According to the pricing and the initial quota limit for Claude 2:

1000 input tokens cost $0.008, 1000 output tokens cost $0.024.

Max 500,000 input and output tokens can be processed per minute according to AWS Bedrock. We can consider the average cost between input and output tokens, which is $0.016 for 1000 tokens.

Leading to the total cost: (500K tokens/1000 * $0.016) * 60 minutes * 24 hours * 4 regions = $46,080 / day

By maximizing the quota limits, attackers can also block the compromised organization from using models legitimately, disrupting business operations.

Detection

The ability to detect and respond swiftly to potential threats can make all the difference in maintaining a robust defense. Drawing insights from recent feedback and industry best practices, we've distilled key strategies to elevate your detection capabilities:

Cloud Logs Detections: Tools like Falco, Sysdig Secure, and CloudWatch Alerts are indispensable allies. Organizations can proactively identify suspicious behavior by monitoring runtime activity and analyzing cloud logs, including reconnaissance tactics such as those employed within AWS Bedrock.

Detailed Logging: Comprehensive logging, including verbose logging, offers invaluable visibility into the inner workings of your cloud environment. Verbose information about model invocations and other critical activities gives organizations a nuanced understanding about activity in their cloud

environments.

Cloud Log Detections

Monitoring cloud logs can reveal suspicious or unauthorized activity. Using Falco or Sysdig Secure, the reconnaissance methods used during the attack can be detected, and a response can be started. **For Sysdig Secure customers, this rule can be found in the Sysdig AWS Notable Events policy.**

Falco rule:

```
- rule: Bedrock Model Recon Activity

  desc: Detect reconaissance attempts to check if Amazon Bedrock is
enabled, based on the error code. Attackers can leverage this to discover
the status of Bedrock, and then abuse it if enabled.

    condition: jevt.value[/eventSource]="bedrock.amazonaws.com" and
jevt.value[/eventName]="InvokeModel" and jevt.value[/
errorCode]="ValidationException"

    output: A reconaissance attempt on Amazon Bedrock has been made
(requesting user=%aws.user, requesting IP=%aws.sourceIP, AWS
region=%aws.region, arn=%jevt.value[/userIdentity/arn],
userAgent=%jevt.value[/userAgent], modelId=%jevt.value[/requestParameters/
modelId])

    priority: WARNING
```

Code language: Perl (perl)

In addition, CloudWatch alerts can be configured to handle suspicious behaviors. Several runtime metrics for Bedrock can be monitored to trigger alerts.

Detailed Logging

Monitoring your organization's use of language model (LLM) services is crucial, and various cloud vendors provide facilities to streamline this process. This typically involves setting up mechanisms to log and store data about model invocations.

For AWS Bedrock specifically, users can leverage CloudWatch and S3 for enhanced monitoring

capabilities. CloudWatch can be set up by creating a log group and assigning a role with the necessary permissions. Similarly, to log into S3, a designated bucket is required as a destination. It is important to note that the CloudTrail log of the InvokeModel command does not capture details about the prompt input and output. However, Bedrock settings allow for easy activation of model invocation logging. Additionally, for model input or output data larger than 100kb or in binary format, users must explicitly specify an S3 destination to handle large data delivery. This includes input and output images, which are stored in the logs as Base64 strings. Such comprehensive logging mechanisms ensure that all aspects of model usage are monitored and archived for further analysis and compliance.

The logs contain additional information about the tokens processed, as shown in the following example:


```
{

    "schemaType": "ModelInvocationLog",

    "schemaVersion": "1.0",

    "timestamp": "[REDACTED]",

    "accountId": "[REDACTED]",

    "identity": {

        "arn": "[REDACTED]"

    },

    "region": "us-east-1",

    "requestId": "bea9d003-f7df-4558-8823-367349de75f2",

    "operation": "InvokeModel",

    "modelId": "anthropic.claude-v2",

    "input": {
```

```
        "inputContentType": "application/json",
        "inputBodyJson": {
            "prompt": "\n\nHuman: Write a story of a young
wizard\n\nAssistant:",
            "max_tokens_to_sample": 300
        },
        "inputTokenCount": 16
    },
    "output": {
        "outputContentType": "application/json",
        "outputBodyJson": {
            "completion": " Here is a story about a young wizard:
\n\nMartin was an ordinary boy living in a small village. He helped his
parents around their modest farm, tending to the animals and working in
the fields. [...] Martin's favorite subject was transfiguration, the art
of transforming objects from one thing to another. He mastered the subject
quickly, amazing his professors by turning mice into goblets and stones
into fluttering birds.\n\nMartin",
            "stop_reason": "max_tokens",
            "stop": null
        },
        "outputTokenCount": 300
```

```
    }

}
```

Code language: Perl (perl)

Example S3 log

Recommendations

This attack could have been prevented in a number of ways, including:

Vulnerability management to prevent initial access.

Secrets management to ensure credentials are not stored in the clear where they can be stolen.

CSPM/CIEM to ensure the abused account had the least amount of permissions it needed.

As highlighted by recent research, cloud vendors offer a range of tools and best practices designed to mitigate the risks of cloud attacks. These tools help organizations build and maintain a secure cloud environment from the outset.

For instance, AWS provides several robust security measures. The AWS Security Reference Architecture outlines best practices for securely constructing your cloud environment. Additionally, AWS recommends using Service Control Policies (SCP) to centrally manage permissions, which helps minimize the risk associated with over-permissioned accounts that could potentially be abused. These guidelines and tools are part of AWS's commitment to enhancing security and providing customers with the resources to protect their cloud infrastructure effectively. Other cloud vendors offer similar frameworks and tools, ensuring that users have access to essential security measures to safeguard their data and services regardless of the platform.

Conclusion

Stolen cloud and SaaS credentials continue to be a common attack vector. This trend will only increase in popularity as attackers learn all of the ways they can leverage their new access for financial gain. The use of LLM services can be expensive, depending on the model and the amount of tokens being fed to it. Normally, this would cause a developer to try and be efficient — sadly, attackers do not have the same incentive. Detection and response is critical to deal with any issues quickly.

IoCs

IP Addresses

83.7.139.184

83.7.157.76

73.105.135.228

83.7.135.97