JavaGhost’s Persistent Phishing Attacks From the Cloud

By Margaret Kelley

Published: 2025-02-28 · Archived: 2026-04-05 16:51:48 UTC

Executive Summary

Unit 42 researchers have observed phishing activity that we track as TGR-UNK-0011. We assess with high confidence that this cluster overlaps with the threat actor group JavaGhost. The threat actor group JavaGhost has been active for over five years and continues to target cloud environments to send out phishing campaigns to unsuspecting targets.

According to website defacement lists such as DefacerID, the group focused historically on defacing websites. However, according to our telemetry, in 2022, they pivoted to sending out phishing emails for financial gain.

Between 2022-24, Unit 42 has performed multiple investigations relating to the group JavaGhost, which targeted organizations’ AWS environments. The group focuses on sending phishing campaigns and has not been seen stealing data for extortion during their time in organizations’ AWS environments.

These attacks are not due to a vulnerability in AWS. This group takes advantage of misconfigurations in the victim organizations' environments that expose AWS credentials in the form of long-term access keys. They use these leaked keys to initiate all the actions discussed in this report.

This article covers common methodologies that JavaGhost uses to create their phishing infrastructure. We also cover other tactics employed within compromised cloud environments to establish long-term persistence.

We have recently observed JavaGhost using advanced evasion methods to cover their tracks. These methods have typically only been used by Scattered Spider, which shows the level of sophistication of this threat actor group.

All JavaGhost activities have resulted in a detectable logging footprint, which forms the basis of the alerts at the end of the article.

Palo Alto Networks customers are better protected through Cortex Cloud and Cortex XSIAM.

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

JavaGhost History

Historically, JavaGhost participated in the website defacement of numerous entities, starting in 2019. Figure 1 shows some of the websites the group defaced.

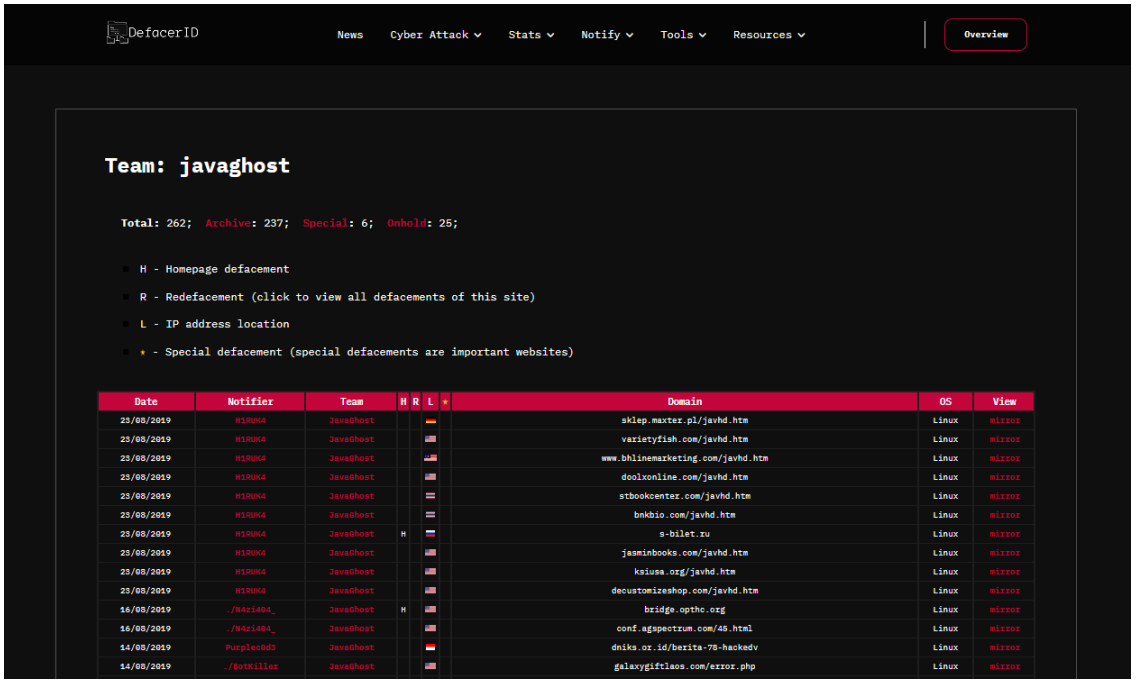

Figure 1. Websites defaced by JavaGhost. Source: DefacerID.

The JavaGhost group also had two websites (shown in Figures 2 and 3). One contains the group's slogan, "we are there but not visible." The other contains text in Indonesian that translates to "stop blaming everything," which matches with the language used to name some of the resources in their attacks. The site also lists the various group member handles (shown in Figure 3).

Figure 2. Historic JavaGhost website. Source: Wayback Machine.

Figure 3. Historic JavaGhost website. Source: Wayback Machine.

Based on our investigations, the group shifted in 2022 from website defacement to sending out phishing campaigns to unsuspecting targets. Datadog reported on this activity shift back in 2023, but the group continues their work, which Unit 42 has seen as recently as December 2024.

Attack Overview

Unit 42 has handled numerous cases in 2022-24 associated with JavaGhost. The attack leveraged overly permissive IAM permissions allowing the victim's Amazon Simple Email Service (SES) and WorkMail services to send out phishing messages. JavaGhost benefits from using other organizations' AWS environments because they do not have to pay for any of the created resources. They can also use preexisting SES infrastructure to send out phishing emails.

Using preexisting SES infrastructure allows the threat actor's phishing emails to bypass email protections since the emails originate from a known entity from which the target organization has previously received emails.

Initial Access with Defense Evasion

Between 2022-24, the group evolved their tactics to more advanced defense evasion techniques that attempt to obfuscate identities in the CloudTrail logs. This tactic has historically been exploited by Scattered Spider. AWS CloudTrail records all management events occurring within an AWS account.

JavaGhost obtained exposed long-term access keys associated with identity and access management (IAM) users that allowed them to gain initial access to an AWS environment via the command-line interface (CLI). These long-term access keys come from various exposures, as discussed in a prior Unit 42 research article.

Upon entry to an organization's AWS environment with the compromised access key, the threat actors do not perform the application programming interface (API) call GetCallerIdentity. Other threat actors often use GetCallerIdentity as their first API call after compromising AWS credentials to enumerate basic information about the compromised account, such as the account ID and user ID.

Because defenders frequently anticipate attackers using GetCallerIdentity during initial compromise, JavaGhost evades detection by *not* using this API call, thereby bypassing any alerts configured to trigger on its execution. Instead, the group performs different first API calls such as GetServiceQuota, GetSendQuota and GetAccount for their initial interaction with a compromised AWS account.

GetServiceQuota returns the current quota limit for a specified AWS service while GetSendQuota returns the max number of emails that can be sent in 24 hours for SES. GetAccount returns information about the email-sending status of SES and other SES attributes.

After confirming their access to an organization's AWS account with the long-term access key from the CLI, the threat actors behind JavaGhost generate temporary credentials and a login URL to allow themselves console access. Accessing the console via this methodology obfuscates their identity and allows them easier visibility into the resources within an AWS account.

Since attackers rarely create temporary credentials to access the AWS console URL, these methods often bypass detection. The following section details these techniques, emphasizing how using the console allows attackers to sidestep the restrictions IAM imposes on temporary access keys generated through the CLI.

GetFederationToken and GetSigninToken

Generating an AWS console login page from long-term access keys takes multiple steps but the entire process can be scripted. NetSPI has created a GitHub repo with an example of how to perform this process and AWS has instructions as well.

The first step in this process requires the creation of temporary credentials from the compromised long-term access key. Long-term access keys start with the four letters AKIA, while temporary access keys begin with ASIA.

To acquire temporary AWS credentials, JavaGhost uses the GetFederationToken API within the AWS Security Token Service (STS). This API call requires the following parameters:

- A name for the federated user
- An inline or managed session policy defining the desired permissions (as illustrated in Figure 4). JavaGhost purposefully utilizes an “allow all” inline policy to take advantage of the maximum permissions allowed to the underlying IAM user.
- The duration for which the temporary credentials should be valid (specified in seconds)

Figure 4. Example of an inline policy from the GetFederationToken event.

While the AssumeRole API call can also retrieve temporary credentials for this process, JavaGhost opts to use the GetFederationToken option instead. Of note, the inline policy provided in the request does not override the

permissions associated with the long-term access key.

The permissions granted to the short-term access key result in an intersection of the IAM user permissions associated with the access key and the policies included in the GetFederationToken request. If the GetFederationToken permissions contain broader privileges than the IAM user, then the more limited permissions from the IAM user take effect. Therefore, the provided policy can only reduce and never increase the permissions already granted to the principal represented by the compromised access key and secret key.

Once the GetFederationToken request returns the temporary credentials (i.e., sessionId, sessionKey and sessionToken), an encoded URL is required before generating the sign-in token. To generate the encoded URL, the threat actor uses the Python urllib3 library.

Once the encoded URL is obtained, a GetSigninToken request returns the information needed to create the URL that allows federated users to access the AWS console. Within the CloudTrail logs associated with the GetSigninToken events, the user agent shows Python-urllib/3.10, which is how Unit 42 inferred the Python library used by JavaGhost to perform these operations.

The generated URL grants access to the console for a default of 15 minutes, which is what the threat actor chose to do. After that, a threat actor must repeat this process to generate a new URL or specify a longer session duration during the GetSigninToken request. The temporary access key generated by the GetFederationToken actions does not need to be regenerated unless the session duration has expired.

To revoke the session associated with the compromised credentials, an IAM policy has to be attached directly to the user. The process discussed above does not require the usage of any roles, so there is no built-in way to revoke the session like AWS provides with an IAM role.

To stop an active threat actor using this console access method, attaching the AWS managed AWSDenyAll policy invalidates all the permissions for the user. It does not stop an active session in the console, but all attempted actions are blocked.

Setting Up the Phishing Infrastructure

Regarding the SES logging configuration, none of the customer AWS environments from our engagements had SES data events enabled. Therefore, the following analysis focuses solely on CloudTrail Management Events.

JavaGhost uses SES and WorkMail to configure their phishing infrastructure. The group starts by creating various SES email identities, followed by updating DomainKeys Identified Mail (DKIM) settings. DKIM uses public key cryptography to verify the authenticity of emails.

The threat actor group also modified the SES Virtual Delivery Manager (VDM) and Mail-from attributes.

To send emails, an SES email or domain identity must exist. The creation of new SES identities appears as CreateEmailIdentity events in the CloudTrail logs and the response elements provide additional details around whether the identity type was a domain or an email address.

JavaGhost creates multiple email and domain identities as well as modifying the following attributes. The DKIM settings are configured during the user creation and generate the PutEmailIdentityDkimAttributes event in

CloudTrail logs.

While DKIM settings can be configured separately from the identity creation process, this group usually updates them during the identity creation itself. The attackers also update the custom Mail-From domain configuration for the email identities. This resulted in the PutEmailIdentityMailFromAttributes event showing the attribute update in the request parameters field within the CloudTrail logs.

The group makes various changes to the SES Virtual Delivery Manager (VDM) feature, which also results in the PutAccountVdmAttributes event appearing in the CloudTrail logs.

In addition to setting up various email identities, JavaGhost configures an AWS WorkMail Organization and adds WorkMail users. Creating a WorkMail Organization results in numerous SES and AWS Directory Service (DS) events within the CloudTrail logs.

Upon the creation of the WorkMail Organization seen as CreateOrganization in the CloudTrail logs, the following events appear in the CloudTrail logs associated with SES:

- CreateReceiptRule
- CreateReceiptRuleSet
- PutIdentityPolicy
- SetActiveReceiptRuleSet
- VerifyDomainDkim
- VerifyDomainIdentity

In the console, within the advanced configuration of the WorkMail Organization creation, user directories can either be created from scratch or an existing directory can be used (shown in Figure 5).

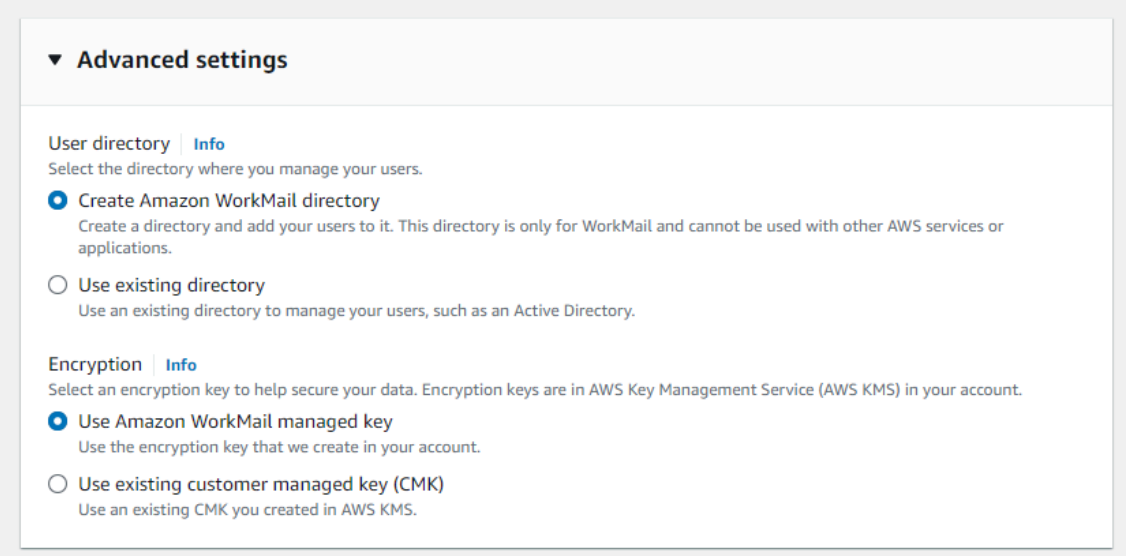

Figure 5. Selecting Create Amazon WorkMail directory generates three Directory Services events automatically in the CloudTrail logs.

Selecting the creation of a new WorkMail directory automatically generates the following DS CloudTrail events:

- AuthorizeApplication

- CreateAlias
- CreateIdentityPoolDirectory

After completing the WorkMail Organization creation, the threat actors create various WorkMail users. Creating a WorkMail user generates a CreateUser event (with workmail.amazonaws[.]com as the event source) and the user automatically gets registered to WorkMail with the event RegisterToWorkMail appearing in the CloudTrail logs. The WorkMail registration requires no input from the user when performed through the console.

Figure 6 shows how to create a new WorkMail user.

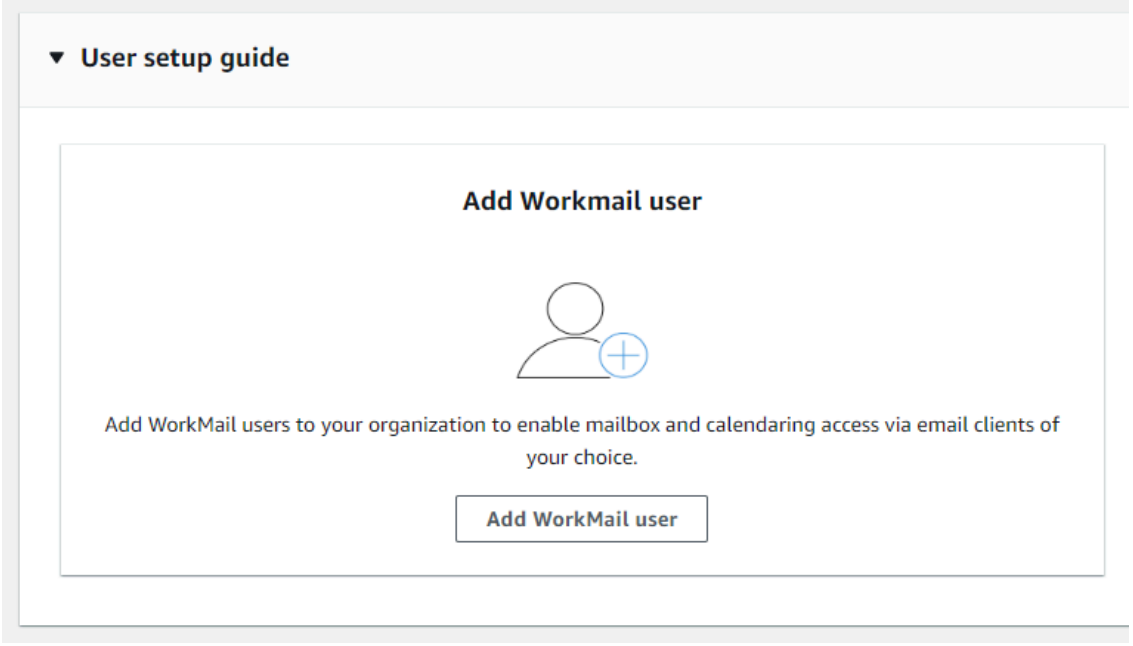

Figure 6. Creating a new WorkMail user.

Before sending out the phishing emails, JavaGhosts creates new SMTP credentials. When creating the new SMTP credentials, the threat actors do not change the default username so all the new SMTP usernames start with ses-smtp-user.* Figure 7 shows an example of this.

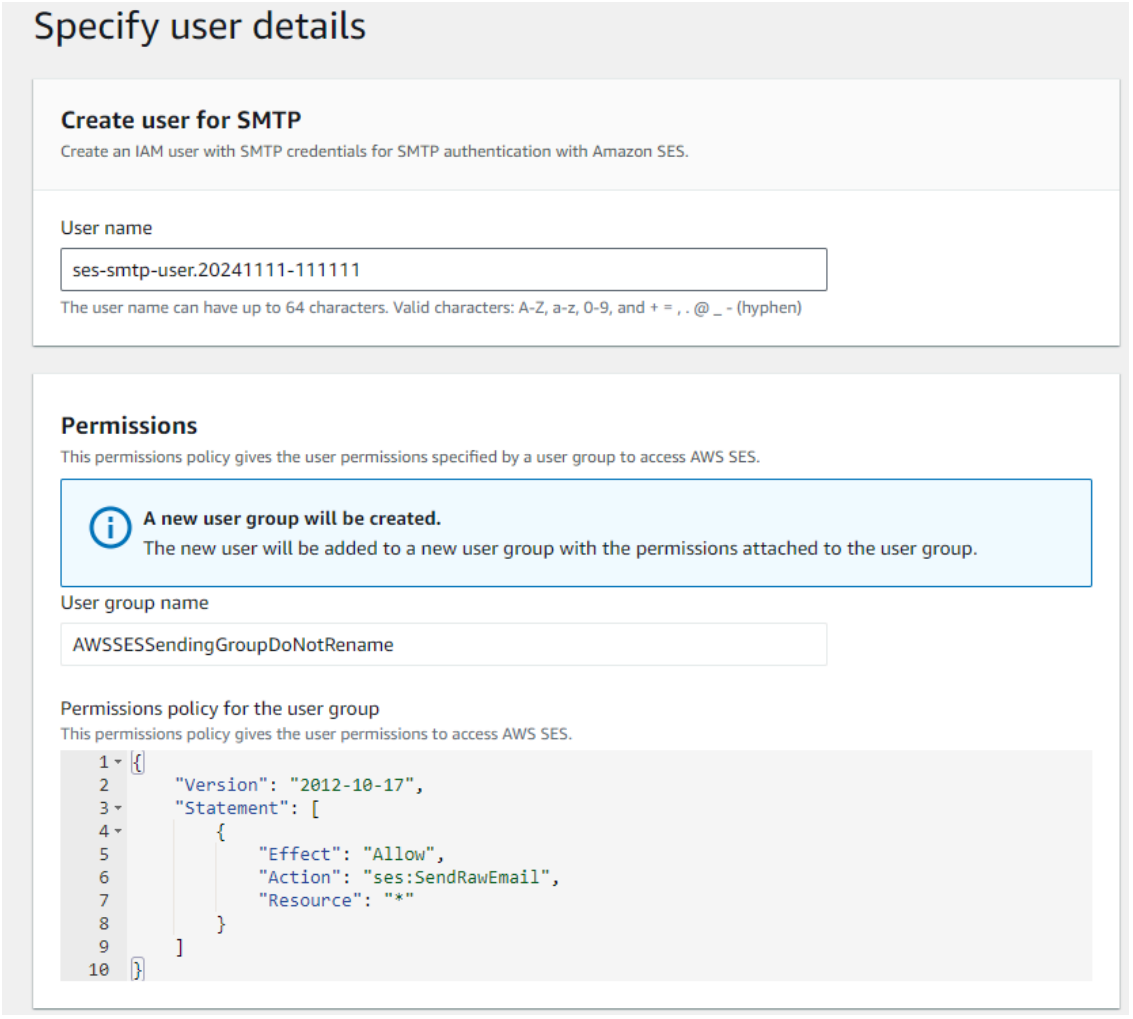

Figure 7. Creation of default named SMTP user with default IAM group and permissions.

Creating new SMTP credentials results in the generation of a new IAM user with the user's name matching the SMTP username and not the SMTP display name. If the AWS account has not used SES historically, the SMTP creator is prompted that a new IAM user group will be created.

This new IAM user group is called AWSSESSendingGroupDoNotRename by default, which also attaches an inline policy to the group allowing ses:SendRawEmail only. These operations appear as CreateGroup and PutGroupPolicy in the CloudTrail logs.

If the AWS account has used SMTP credentials historically, the IAM group will most likely already exist and appear in the Permissions list. Figure 8 shows an example of this.

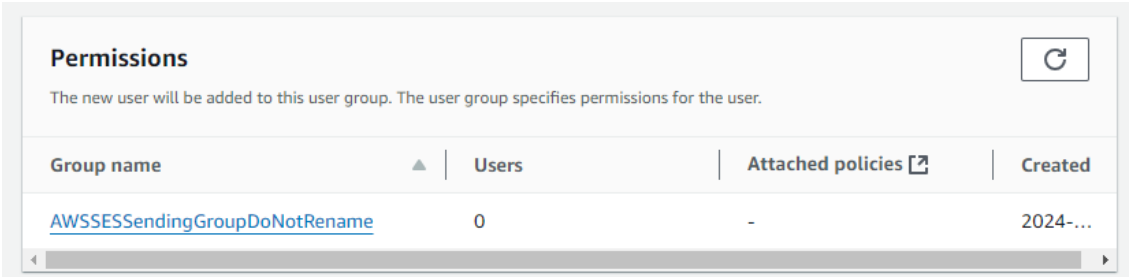

Figure 8. IAM group details once IAM group already exists.

After finishing user creation, the system displays the new IAM username, along with the SMTP username and SMTP password.

The SMTP username displays an access key ID. When reviewing the user in IAM, the SMTP username appears as an access key there as well. Figure 9 shows an example of this.

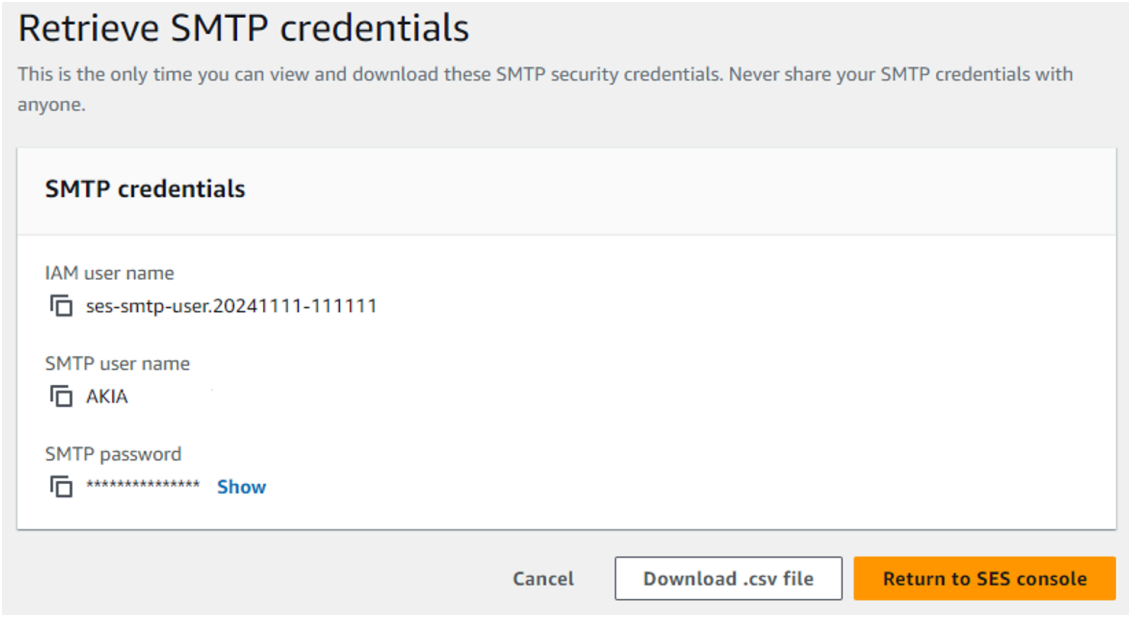

Figure 9. Example retrieval of SMTP credentials.

The SMTP username still resolves to the AWS account ID if decrypted. All these events appear in the CloudTrail logs as IAM CreateUser, CreateAccessKey and AddUserToGroup events.

When organizations already have SES infrastructure in their AWS environment, JavaGhost uses the preexisting resources to send phishing attacks. Unless dataplane logging is enabled, there are few to no events to review in the CloudTrail logs. The cost for the additional emails sent will appear in the Cost and Usage Reports, but otherwise, only various reconnaissance events result in CloudTrail logs.

Identity and Access Management (IAM)

Throughout the time frame of the attacks, JavaGhost creates various IAM users, some they use during their attacks and others that they never use. The unused IAM users seem to serve as long-term persistence mechanisms.

After their creation, the threat actor only confirms access via console logins and performs no other actions. The IAM users have a variety of names. Some are meant to blend in with other IAM users that would be typical within an AWS account and others are more obviously named. The IoC section provides a full list of IAM usernames.

All the new IAM users have the AWS managed AdministratorAccess policy attached as well as access to the console. The AdministratorAccess policy allows any action against any resource within an AWS account.

Figure 10 shows the permissions associated with this policy. All of these IAM events appear in the CloudTrail logs as CreateUser, AttachUserPolicy and CreateLoginProfile.

Figure 10. AWS managed AdministratorAccess policy.

The creation of IAM users is a common cloud technique commonly seen within many of our other investigations. JavaGhost sets themselves apart by evolving to use unique methods to access an AWS account.

In the initial attacks, this group used the original compromised access key for most of their activity. In 2024, they transitioned to using an IAM role to access the organization's AWS account from a threat actor-compromised AWS account before proceeding with the attack.

To accomplish this, the threat actors created a new IAM role with a trust policy attached, allowing access from a threat actor-controlled AWS account. A trust policy specifies what entities can assume the role.

This role creation appears in the CloudTrail logs as CreateRole with the trust policy written in the request parameters field. Figure 11 shows an example of a trust policy.

Figure 11. Example of an inline trust policy from the CreateRole CloudTrail event.

In the case of JavaGhost, the trusted entity belongs to an AWS account. The new role also has unlimited permissions within the environment, with the attachment of the AdministratorAccess policy as seen by the CloudTrail event AttachRolePolicy.

With the successful creation of the new administrative role, the threat actor can log into the AWS account from the trusted threat actor-owned AWS account. When the threat actor assumes their role that they created to access the compromised AWS account, CloudTrail records this event as two separate events, AssumeRole and SwitchRole,

which occur simultaneously. Unlike the creation of new IAM users, the role creation does not appear suspicious until the trust policy reveals the external access and the role is uncovered as a backdoor.

Security Group

The group continues to leave the same calling card in the middle of their attack by creating new Amazon Elastic Cloud Compute (EC2) security groups named Java_Ghost, with the group description "We Are There But Not Visible." These security groups do not contain any security rules and the group typically makes no attempt to attach these security groups to any resources. The creation of the security groups appear in the CloudTrail logs in the CreateSecurityGroup events.

This group description matches the group's slogan on their old website, shown in Figure 12.

Figure 12. JavaGhost website. Source: Wayback Machine.

Additional Suspicious Activity

In addition to the main components of its phishing attacks, the group attempts two other unique tactics within attacks:

- The group attempts to leave an Organization Unit with the event LeaveOrganization. AWS Organizations help with the management of multiple AWS accounts. They consist of features such as Service Control Policies (SCPs), which help manage IAM permissions at scale, and Organizational Units.
    - Organization Units help administrators manage multiple AWS accounts by grouping them together, and they allow for the application of SCPs at the Organization Unit level. Leaving an AWS Organization Unit removes any SCPs that apply to the AWS account and changes the security guardrails that limit activities within an AWS account.
- The group enables all AWS regions not enabled by default.
    - After March 20, 2019, AWS no longer enables new regions by default and JavaGhost enables those 13 disabled regions as part of their attacks to evade security controls. Enabling regions results in the EnableRegion event in the CloudTrail logs with the region name present in the request parameters field.

Conclusion

Unit 42 has investigated multiple JavaGhost cases over the past few years and has observed the group continuously evolving its tactics. Initially, JavaGhost performed attacks using only a compromised access key, but has now advanced to employing sophisticated evasion techniques. Luckily, all of the group's activity results in detectable events within the CloudTrail logs that organizations can hunt for and create new alerts to detect.

Palo Alto Networks Protection and Mitigation

Palo Alto Networks customers are better protected from the threats discussed above through the following products:

Cortex Cloud and Cortex XSIAM alert on the following activities related to AWS resources:

- IAM actions such as new user creations, attaching of AdministratorAccess Policy, getfederatedtoken, and getsignintoken
- Suspicious sending of emails through Simple Email Service (SES)
- Use of getgroup and putgroup in CloudTrail

XSIAM also detects behavioral actions from cloud and on-premises endpoints that suggest the collection of AWS IAM credentials.

To mitigate opportunities for attackers to use techniques discussed above, we recommend:

- Limiting access to administrative rights
- Rotating IAM credentials regularly
- Using short term/just-in-time (JIT) access tokens
- Enabling multi-factor authentication (MFA)

Cloud security posture management (CSPM) capabilities in Cortex Cloud can assist users with creating appropriate rules.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

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. 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.

Cortex XQL Queries

Authentication

```
dataset = amazon_aws_raw

| filter (eventSource = "sts.amazonaws.com" and eventName = "GetFederationToken") or (eventSource = "signin.amazonaws.com" and eventName = "GetSigninToken")

dataset = amazon_aws_raw

| filter (eventSource = "sts.amazonaws.com" and eventName = "AssumeRole") or (eventSource = "signin.amazonaws.com" and eventName = "SwitchRole")
```

SES

```
dataset = amazon_aws_raw

| filter (eventSource = "ses.amazonaws.com" and eventName = "CreateEmailIdentity") or (eventSource = "iam.amazonaws.com" and eventName in ("CreateUser", "CreateAccessKey", "AddUserToGroup"))
```

WorkMail

```
dataset = amazon_aws_raw

| filter eventSource = "workmail.amazonaws.com" and eventName in ("CreateUser", "CreateOrganization")
```

EC2 Security Group

```
dataset = amazon_aws_raw

| alter groupName = json_extract_scalar(requestParameters, "$.groupName")

| alter groupDescription = json_extract_scalar(requestParameters, "$.groupDescription")

| filter eventSource = "ec2.amazonaws.com" and eventName = "CreateSecurityGroup"
```

```
| filter groupName = "Java_Ghost" and groupDescription = "We Are There But Not Visible"
```

IoCs

IP Addresses

Unit 42 has consolidated the IP addresses of the referenced group in this report and stored them in our GitHub repository.

User Agents

- aws-cli/1.18.69 Python/3.8.10 Linux/5.4.0-113-generic botocore/1.16.19
- aws-cli/1.19.112 Python/2.7.18 Linux/5.4.0-42-generic botocore/1.20.112
- aws-cli/1.22.23 Python/3.6.0 Windows/10 botocore/1.23.23
- aws-cli/1.22.97 Python/3.6.0 Windows/10 botocore/1.24.42
- aws-cli/1.25.62 Python/3.8.13 Linux/5.15.0-46-generic botocore/1.27.61
- aws-cli/1.34.14 md/Botocore#1.35.14 ua/2.0 os/windows#10 md/arch#amd64 lang/python#3.10.8 md/pyimpl#CPython cfg/retry-mode#legacy botocore/1.35.14
- aws-cli/1.34.28 md/Botocore#1.35.28 ua/2.0 os/linux#5.15.153.1-microsoft-standard-WSL2 md/arch#x86_64 lang/python#3.12.3 md/pyimpl#CPython cfg/retry-mode#legacy botocore/1.35.28
- aws-cli/2.13.18 Python/3.11.5 Linux/5.4.0-163-generic exe/x86_64.ubuntu.20 prompt/off command/*
- aws-cli/2.17.18 md/awscrt#0.20.11 ua/2.0 os/linux#6.8.0-36-generic md/arch#x86_64 lang/python#3.11.9 md/pyimpl#CPython cfg/retry-mode#standard md/installer#exe md/distrib#ubuntu.24 md/prompt#off md/command#*
- aws-cli/2.22.2 md/awscrt#0.22.0 ua/2.0 os/windows#2019Server md/arch#amd64 lang/python#3.12.6 md/pyimpl#CPython cfg/retry-mode#standard md/installer#exe md/prompt#off md/command#*
- aws-cli/2.2.16 Python/3.8.8 Linux/3.10.0-1160.31.1.el7.x86_64 exe/x86_64.centos.7 prompt/off command/*
- aws-internal/3 aws-sdk-java/1.12.769 Linux/5.10.224-190.876.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.12+8-LTS java/1.8.0_422 vendor/N/A cfg/retry-mode/standard
- aws-internal/3 aws-sdk-java/1.12.769 Linux/5.10.225-191.878.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.12+8-LTS java/1.8.0_422 vendor/N/A cfg/retry-mode/standard
- Boto3/1.24.61 Python/3.8.10 Linux/5.4.0-42-generic Botocore/1.27.61
- Boto3/1.35.28 md/Botocore#1.35.28 ua/2.0 os/linux#5.15.153.1-microsoft-standard-WSL2 md/arch#x86_64 lang/python#3.12.3 md/pyimpl#CPython cfg/retry-mode#legacy Botocore/1.35.28
- Boto3/1.35.3 md/Botocore#1.35.14 ua/2.0 os/windows#10 md/arch#amd64 lang/python#3.10.8 md/pyimpl#CPython cfg/retry-mode#legacy Botocore/1.35.14
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:130.0) Gecko/20100101 Firefox/130.0
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36
- Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36

- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 OPR/113.0.0.0
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36 Edg/128.0.0.0
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36 Edg/129.0.0.0
- Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
- Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
- Python-urllib/3.10

IAM Usernames

- adminuserdevs
- develops
- Gh0st_808
- Gh0st_365
- rootdev
- ses2
- warkopi
