How Adversaries Can Persist with AWS User Federation

By Vaishnav Murthy - Joel Eng

Archived: 2026-04-06 01:18:23 UTC

- CrowdStrike Services identified a novel technique used by threat actors that escapes typical containment practices and permits persistence in victim AWS environments.
- The technique requires that the adversary first obtain valid AWS API credentials with the necessary security token service (STS) and identity and access management (IAM) permissions, and then use the `sts:GetFederationToken` API call to create a federated user session.
- Permissions and access to the federated sign-in session survive the deactivation of the base user's API credentials.
- Responders should attach an explicit deny-all IAM policy to compromised IAM users as a containment measure. If a root API key is compromised, containment is much more challenging and may not always be possible. **Note:** Creating an AWS API key (a prerequisite for this technique) for the root user is discouraged by AWS. Organizations should use the root user sparingly, and only via the AWS console.

In recent incident response investigations, CrowdStrike Services has observed adversaries use the `sts:GetFederationToken` API call to create federated sessions from IAM users. In this scenario, the federated session inherits permissions from the base IAM user.

Perhaps surprising to many incident responders, the privileges and access of the federated session are not revoked when the base IAM user's credentials are deactivated. Many defenders may not realize it, but this is a characteristic of all forms of temporary credentials obtained via the various API calls to AWS STS. The resulting sessions are independent of the long-term credentials of the underlying IAM or root user. The privileges of the federated session are only reduced or revoked when the IAM policies attached to the base IAM user are modified to reduce or revoke those permissions. (Deleting the IAM user will effectively revoke all privileges but CrowdStrike does not recommend doing so, as detailed below.). The federated session remains active until the session token expires.[1]

It is common for incident responders to only deactivate the base IAM user's credentials. However, this will not be sufficient to contain an attack utilizing this technique. CrowdStrike is posting this blog to raise awareness of the use of this technique by threat actors and how threat actors can expand its use to establish persistence and escalate privileges within AWS environments if the underlying IAM user already had those escalation privileges.

What Is a Federated Session?

A federated session in AWS involves exchanging long-lived credentials for a temporary set of credentials: an AWS access key, AWS secret key and AWS session token, issued by AWS STS. AWS has stated that federated sessions were predominantly used within custom access brokers before the advent of modern methods using IAM roles instead.

AWS's best practice is to use temporary credentials instead of long-lived credentials (such as API keys and passwords tied to local IAM users) in most use cases. AWS also recommends using federation-based AWS Identity Center for managing access for all human users. Using temporary security credentials enhances security by eliminating the exposure from static credentials that are vulnerable to theft and ensuring that any stolen session tokens have a limited lifespan.

A method for federating users that, in CrowdStrike's experience, is not well known by cloud practitioners is federating directly from IAM user and root principals. All IAM users that have been granted the permission `sts:GetFederationToken` and the account's root user are capable of acting as local identity providers and issuing temporary credentials for this type of federation. AWS describes a use case for this capability as:

> A typical use is in a proxy application that gets temporary security credentials on behalf of distributed applications inside a corporate network. You must call the GetFederationToken operation using the long-term security credentials of an IAM user. As a result, this call is appropriate in contexts where those credentials can be safely stored, usually in a server-based application.

Because it is older and less commonly used, most organizations are unaware of the user-based federation capability and treat IAM users as local service accounts whose permissions are bound to the long-term credentials of that single entity. However, using the `sts:GetFederationToken` API, a user can pass its privileges for use in a derivative federated session, which operates with distinct credentials.

AWS places certain limits on federation-based privilege inheritance: The temporary credentials associated with the federated session cannot have greater privileges than the original identity. In addition, the federated session is blocked from exercising IAM actions from the AWS CLI or API. However, these IAM limits do not apply to console sessions, which is how an adversary with the compromised credentials of an IAM user with certain IAM privileges can abuse user-based federation to persist despite deactivation of the base identity's compromised credentials.

AWS previously indicated in its documentation that IAM limitations for federated sessions were limited to the CLI/SDK and did not apply when using the AWS console,[2] but after being notified of these concerns by CrowdStrike, AWS agreed the documentation could be made clearer and has done so by stating explicitly that IAM limitations do not apply to console usage. This change was made prior to the publication of this blog post.

Persistence Technique

This persistence technique takes advantage of the following:

- Incident response teams are trained to preserve artifacts and affected resources such as IAM users during an incident. So instead of deleting the affected IAM user, or detaching IAM policies, they often will simply disable an IAM user's API keys as a method of containment.

- Federated sessions can be used to make IAM changes via the AWS console

The steps to utilize this technique are outlined below. Note that the technique works for an account's root user as well as any IAM user with the required permissions, but in the interest of clarity, only the IAM user approach is described.

1. Compromise an IAM user with the required permissions

The technique begins when an adversary compromises a user who has the minimum permissions set of `sts:GetFederationToken`, `iam:ListAccessKeys` and `iam:UpdateAccessKey`. This is unfortunately a common scenario, given that many operations teams attach the highly privileged `AdministratorAccess` IAM policy to IAM users for ease of use, which allows all API actions.

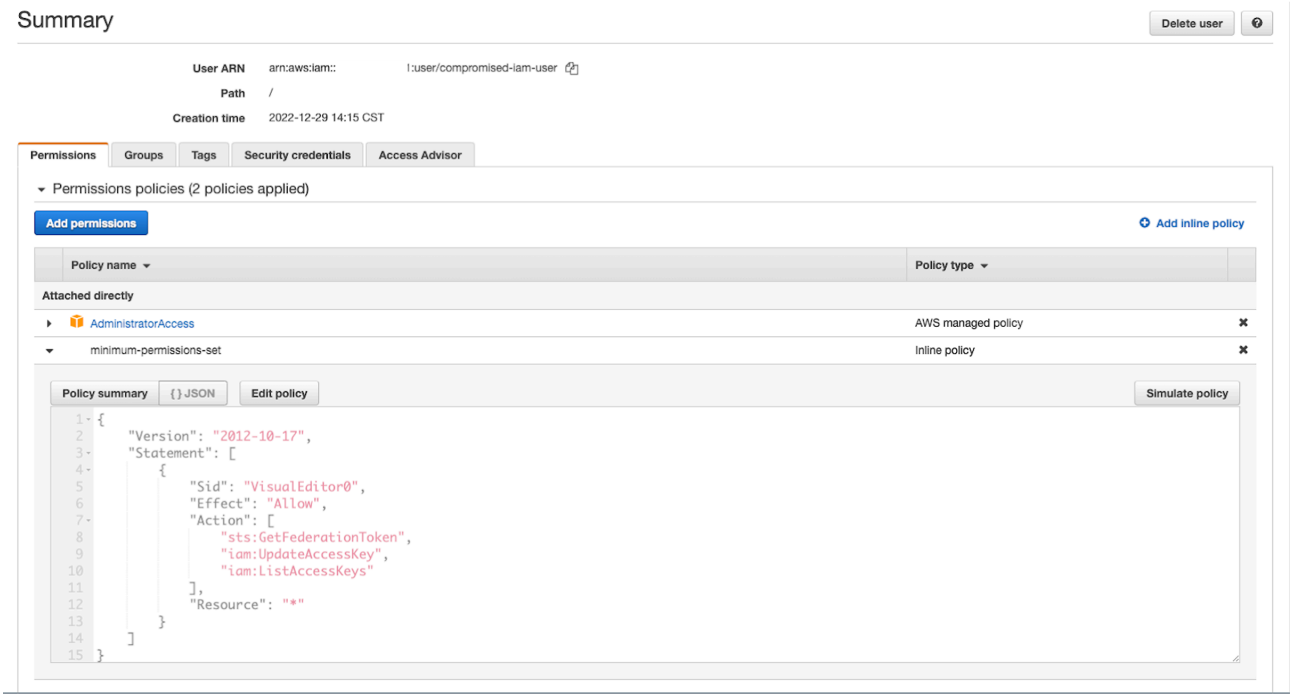

Figure 1. IAM policy granting permission to call the `sts:GetFederationToken` API call (click to enlarge)

This IAM user **does not need to have a password (console access) configured.** The technique will work as long as the adversary has access to API keys associated with the base IAM user.

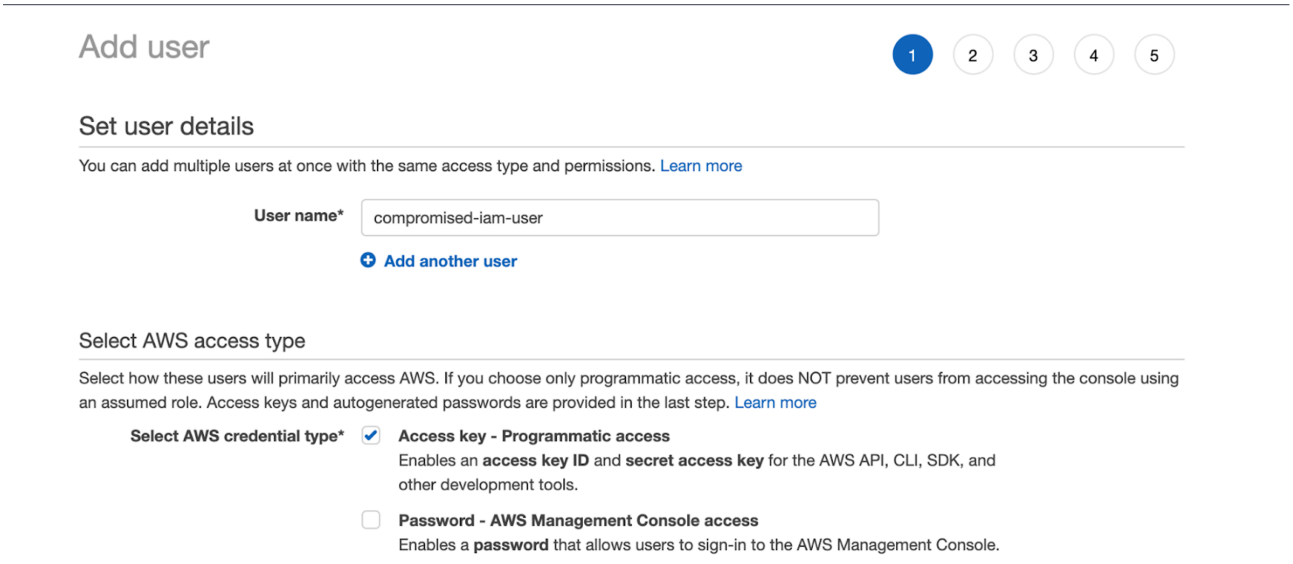

Figure 2. IAM user has no password, only an access key (clock to enlarge)

2. Create a federated session from this IAM identity

The next step is to use an API key associated with the compromised IAM user to get a federation token derived from the identity of that user. The API call requires the caller to specify a name for the session (in this case, “UnauthorizedUser”) and one or more in-line policies or policy ARNs to limit the privileges that will be granted to the session. The effective permissions of the resulting temporary credentials are the least-privileged subset of the requested permissions and the calling user’s permissions. By specifying `AdministratorAccess`, which allows all API actions, the adversary ensures that the federated session will have all of the permissions of the calling user.

```
jeng02@ML-C02GH2BVMD6T CRTBT % aws sts get-federation-token \
  --name UnauthorizedUser \
  --policy-arns arn=arn:aws:iam::aws:policy/AdministratorAccess \
  --duration-seconds 43200
{
    "Credentials": {
        "AccessKeyId": "ASIAY7KV3F6F45TZKMWS",
        "SecretAccessKey": "bnO9zBjoFnhxQfgrvNKvdvVW149WUj5U4458FtWs",
        "SessionToken": "IQoJb3JpZ2luX2VjEJD//////////wEaCXVzLWVhc3QtMSJHMEUCIADmR5zSEp79I6SSL4Q6XoqxrMt8NjLhm15WDjJPcAkcAiEA5PVBNF4cnVtgc1ZeSt0QjaMRhS8MjqObaWq5lfzZDGEqrwII2P
//////////ARADGgw2MTcwNDQ3MTc0NTEiDC3E5dMeDKZ7owtp6SqDAqTPZDYcWt72eBbgePBCd6sFcEYmrd7VsYY3RTfHWILNg5D2+KzwquDtMETx60onZVjruLFi0ULzneN2yb/80kmIZRFHmh27Rjzbc4Klxvh6BAebXFazUtzT4
jKDUIx3GLJVBcPotonV48Dz/CxLVSDhDlz1C4rnXR1m8h33qoiQyuruWpPxu6I3XXISpKEYcztg+wHqYZdEnz/BAZvLWHDXUQvptpmV6+kGkRUC1PMNJArSdaUT9f+QgW+ge+lv9rApC4sY2zFYFYU2bP1FoYDcC2lR6OLeVmF5XHtL
ZubRrASJZZRu74TjTR3G2y8hpAmGebNMBZ6l2g/PTYZyAPOcDUYwno7RnQY62QFPDaptbM3wrIQ2TvoELuMy2Du/201caKEn9H8plpiNv4i+LlX/9uiT+60hYTjxQGqmV+AASjrZXauu9z5M9Yf83MDklXpQRP1oTeAHHbYnYtfcnn6
Q6drnrTo4Mki+AA/bnfymMQGl0YK4R41TVe8LVvNMovlouHiMraJ2rM6cmo9wDR2Qv/9uPwagM8uyjT0bOhyZbCgyN9Z6Jtffimjc957GzxuQL7CMA6Uv4OkZuyr4VZkOTi0tscE1lyqMY40oUAXGVZbzVRMR6opHWoF6gcavXEzAzY
ZV",
        "Expiration": "2023-01-04T03:17:50+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "[redacted]:UnauthorizedUser",
        "Arn": "arn:aws:sts::[redacted]:federated-user/UnauthorizedUser"
    },
    "PackedPolicySize": 7
}
jeng02@ML-C02GH2BVMD6T CRTBT %
```

Figure 3. API call creating the federated session from the base IAM user (click to enlarge)

3. The federated session permissions allow the adversary to perform all actions that the IAM user can perform.

In the example in Figure 4, the adversary uses the new federated session tokens to exercise the `ec2:DescribeRegions` permission inherited from the federating IAM user.

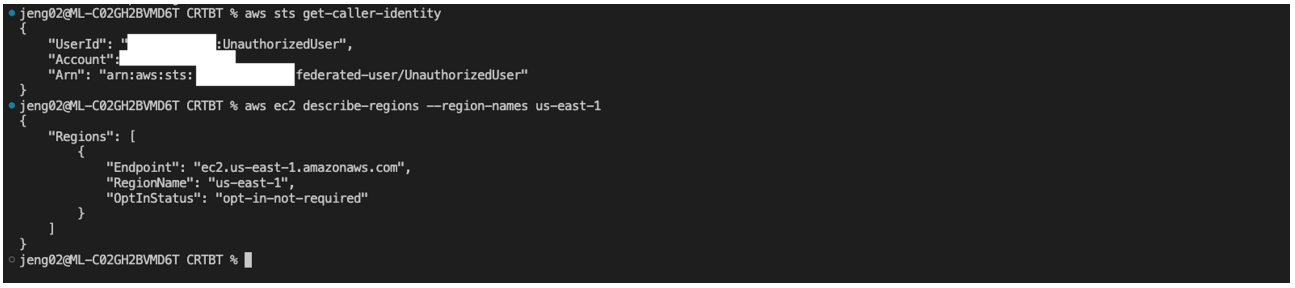

Figure 4. API calls made using the federated session (click to enlarge)

4. The adversary can use the federated session to log into the console, even if the IAM user has no password.

Converting the AWS CLI session to a console session is accomplished by using the federated session token to request a sign-in token from `https://signin.aws.amazon<.>com/federation` and then using that token to authenticate to the AWS console endpoint `https://console.aws.amazon<.>com/console/home`. NetSPI has an easy-to-use tool for this on GitHub. CrowdStrike has responded to incidents in which adversaries used the `aws_consoler` tool to generate a sign-in token to access the AWS console using AWS API keys.

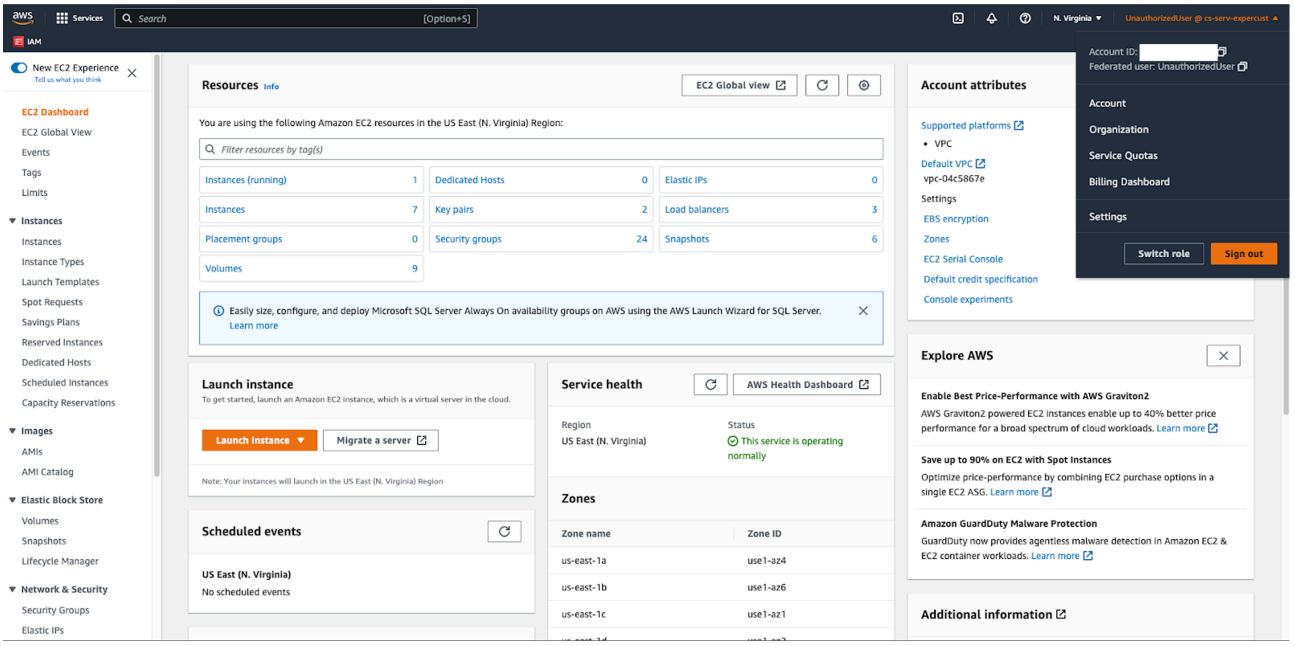

Figure 5. Console login using the federated session (click to enlarge)

5. Changing permissions of the originating IAM user immediately affects the federated session, but disabling the access key that was used to issue the session does not.

While changes to the permissions of the base IAM user are propagated to the federated session, CrowdStrike found that disabling the access key that produced the federation session leaves the session intact for the remainder of the duration specified when it was created. In an incident response situation involving a compromised IAM user, many incident response teams leave the IAM user in place, but disable all access keys as part of containment. Based on CrowdStrike's observations, this commonly leveraged remediation action is insufficient. Defenders must instead revoke or override all permissions for that user. CrowdStrike recommends attaching an explicit deny-all

IAM policy to the IAM user, since explicit deny statements take precedence over allow statements. This would let incident responders preserve the IAM user's existing permissions instead of having to record them separately and recreate them later.

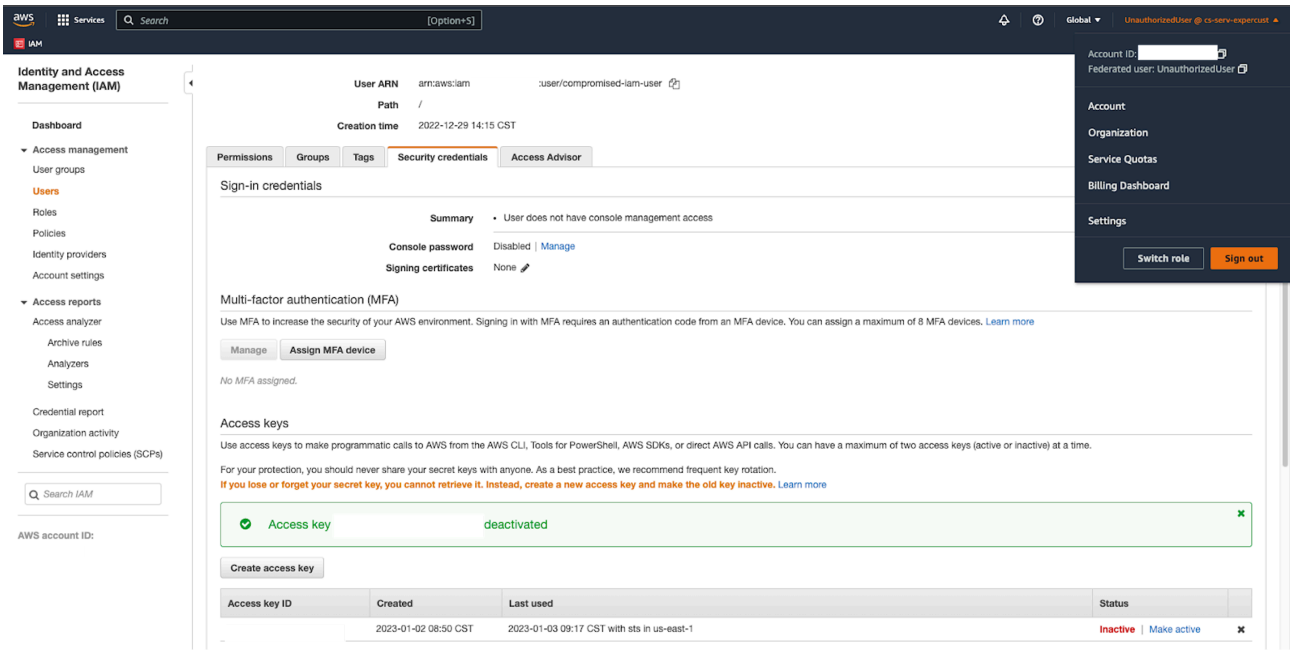

Figure 6. Disabling the base user's access key does not invalidate the federated session (click to enlarge)

6. If the base IAM user has the UpdateAccessKey IAM permissions assigned to it, the adversary can reactivate the deactivated credentials, establishing persistence.

Adversaries also have a way to reestablish persistence using a lingering federated session. As noted earlier, AWS documentation on the `GetFederationToken` API stated prior to the release of this blog post:

> You can use temporary credentials generated by `GetFederationToken` in any AWS service except the following: You cannot call any IAM operations using the AWS CLI or the AWS API.

While most IAM actions are denied when using the AWS CLI or the AWS API, these actions are **allowed** when done using the AWS console. A lingering federated session allows the adversary to avoid the additional limitations AWS places on CLI/SDK-based IAM operations by converting to a console session. An adversary using a compromised access key with the sufficient IAM permissions can simply re-enable the key after incident responders have performed containment and then issue itself another session.

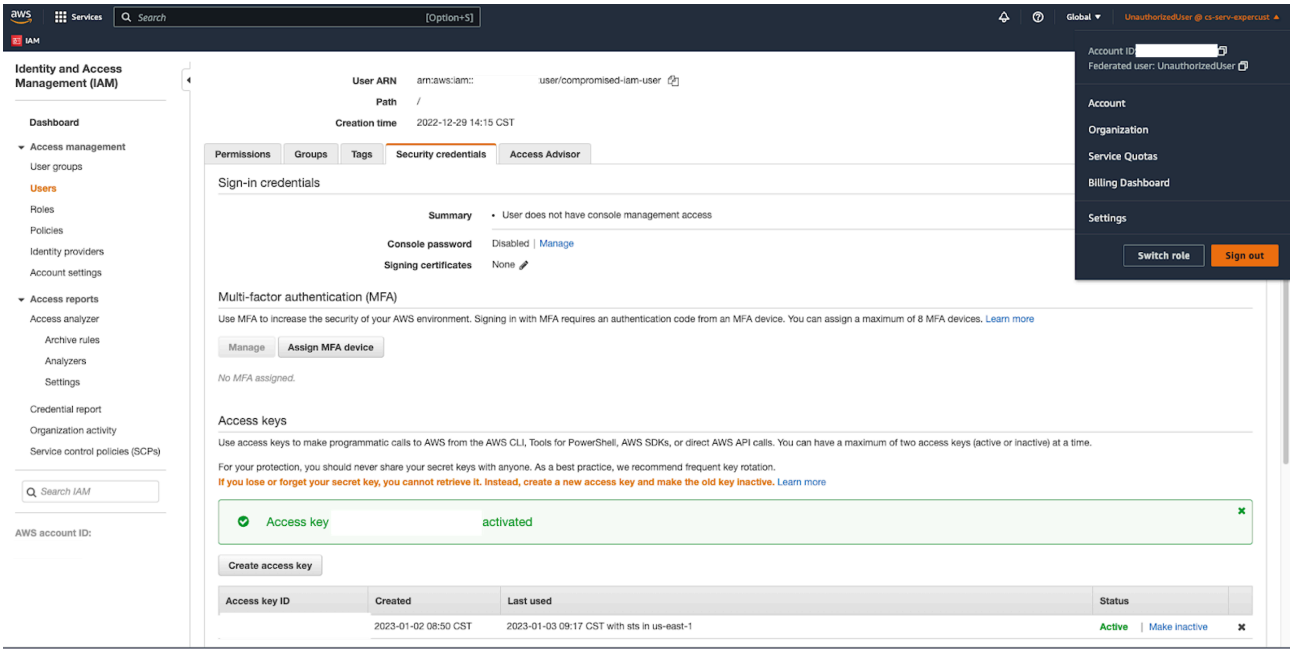

Figure 7. The federated session is able to re-enable the disabled access key of the base user (click to enlarge)

In comparison, the IAM actions are denied through the CLI regardless of permissions.

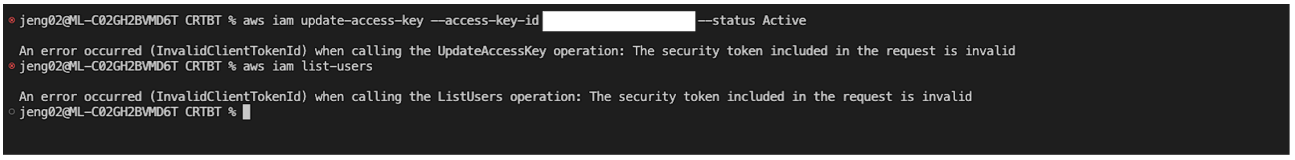

Figure 8. The same IAM actions are not allowed using the CLI (click to enlarge)

The CloudTrail log generated from this activity can be found below:[3]


```
{
   "eventVersion": "1.08",
   "userIdentity": {
       "type": "FederatedUser",
       "principalId": "XXXXXXXXXXX:UnauthorizedUser",
       "arn": "arn:aws:sts::XXXXXXXXXXX:federated-user/UnauthorizedUser",
       "accountId": "XXXXXXXXXXX",
       "accessKeyId": "ASIAXXXXXXXXXXX",
       "sessionContext": {
           "sessionIssuer": {
               "type": "IAMUser",
               "principalId": "AIDAY7KV3F6F6NXQB6KUH",
               "arn": "arn:aws:iam::XXXXXXXXXXX:user/compromised-iam-user",
               "accountId": "XXXXXXXXXXX",
               "userName": "compromised-iam-user"
           },
           "webIdFederationData": {},
           "attributes": {
               "creationDate": "2023-01-04T19:43:13Z",
```

```
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-04T19:53:35Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "UpdateAccessKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "userName": "compromised-iam-user",
        "accessKeyId": "AKIAXXXXXXXXX",
        "status": "Active"
    },
    "responseElements": null,
    "requestID": "eea9ee11-edb3-4de2-aa00-57a8ec37c9e1",
    "eventID": "beec4fb1-3947-485b-ba2b-12a2cf4353ac",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "XXXXXXXXXXXX",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```


7. If the IAM user has the AttachUserPolicy or PutUpdatePolicy IAM permissions assigned to it (essentially permitting privilege elevation by policy), the federated session can create, modify or attach additional IAM policies to the base IAM user, potentially escalating their privileges — something that is not possible when using the CLI or API.

Adversaries could similarly use the federated session to escalate their privileges if the base user has the requisite IAM permissions (`AttachUserPolicy` or `UpdateUserPolicy`) assigned to it. (It should be noted that any principal with those unrestricted privileges is de facto a full administrator since they can add administrative privileges to themselves at any time.). Once logged in via the console, the federated session could be used to attach a privileged policy (such as `AdministratorAccess`) to the base IAM user, which would then elevate their own permissions to include `AdministratorAccess` .

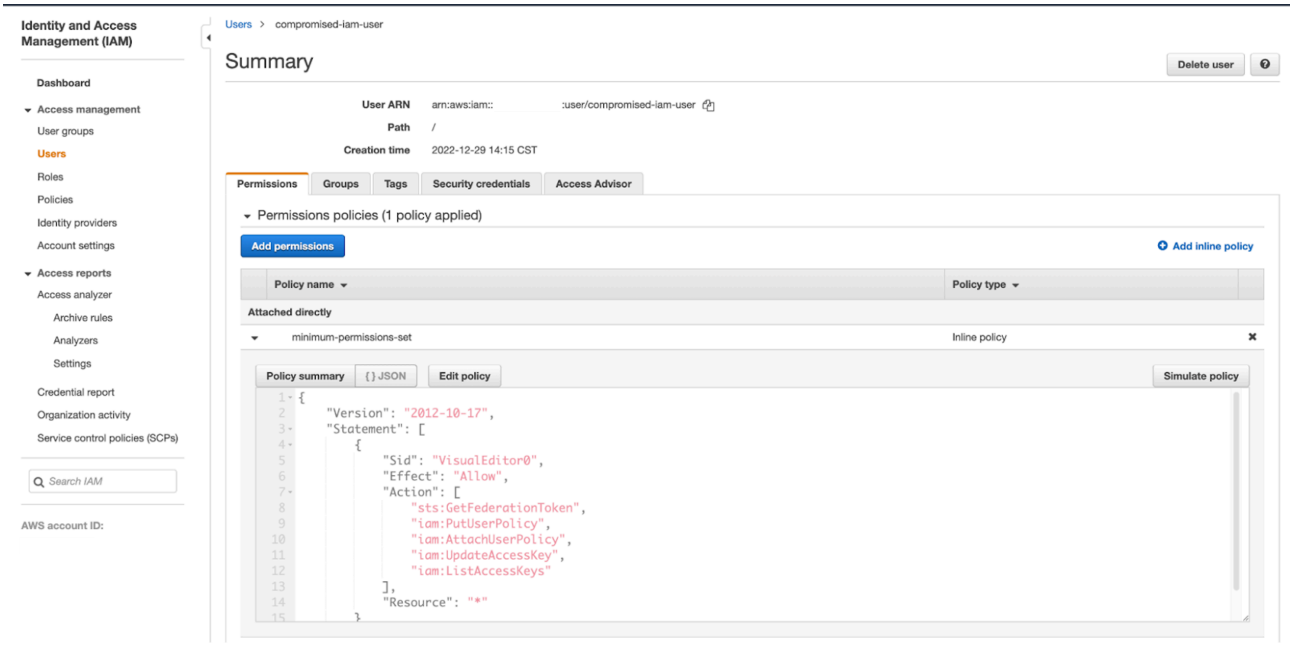

Figure 9. The minimum permissions set necessary for privilege escalation (click to enlarge)

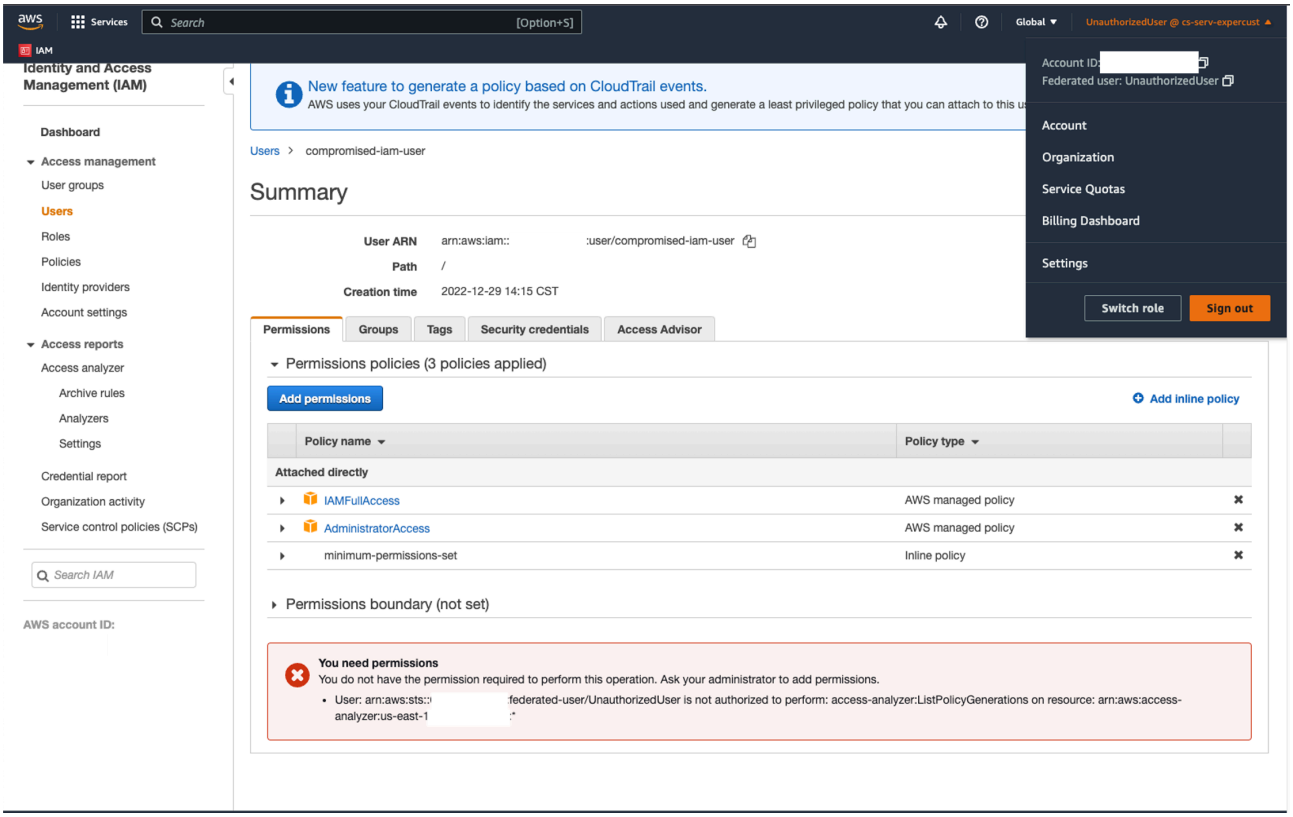

Figure 10. The federated session has added `AdministratorAccess` to the base IAM user (click to enlarge)

Note that just as in the previous example, these actions **cannot** be performed via the AWS CLI.

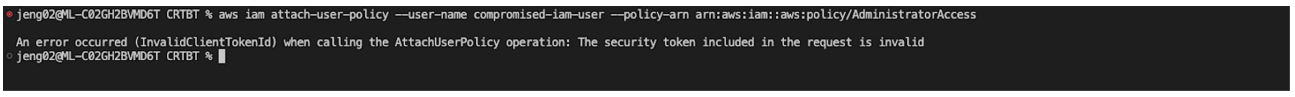

Figure 11. The same API call is blocked using the CLI (click to enlarge)

8. The effective privileges of the federated session are only revoked when the base user's permissions are detached, or an explicit deny-all IAM policy is applied.

The only ways to fully contain an adversary using an IAM user federated session are to fully remove or override the permissions of the IAM user. CrowdStrike recommends overriding the base IAM user's permissions with an explicit deny-all IAM policy, or if the AWS account is a member of an AWS Organization, an SCP rather than removing attached IAM policies. This would allow incident responders to more easily review the full scope of capabilities an adversary had during an incident and makes it easier for operations teams to restore functionality post-remediation. The example below shows an explicit deny-all IAM policy attached to the compromised IAM user, that results in the lingering federated session losing all permissions, even in the console.

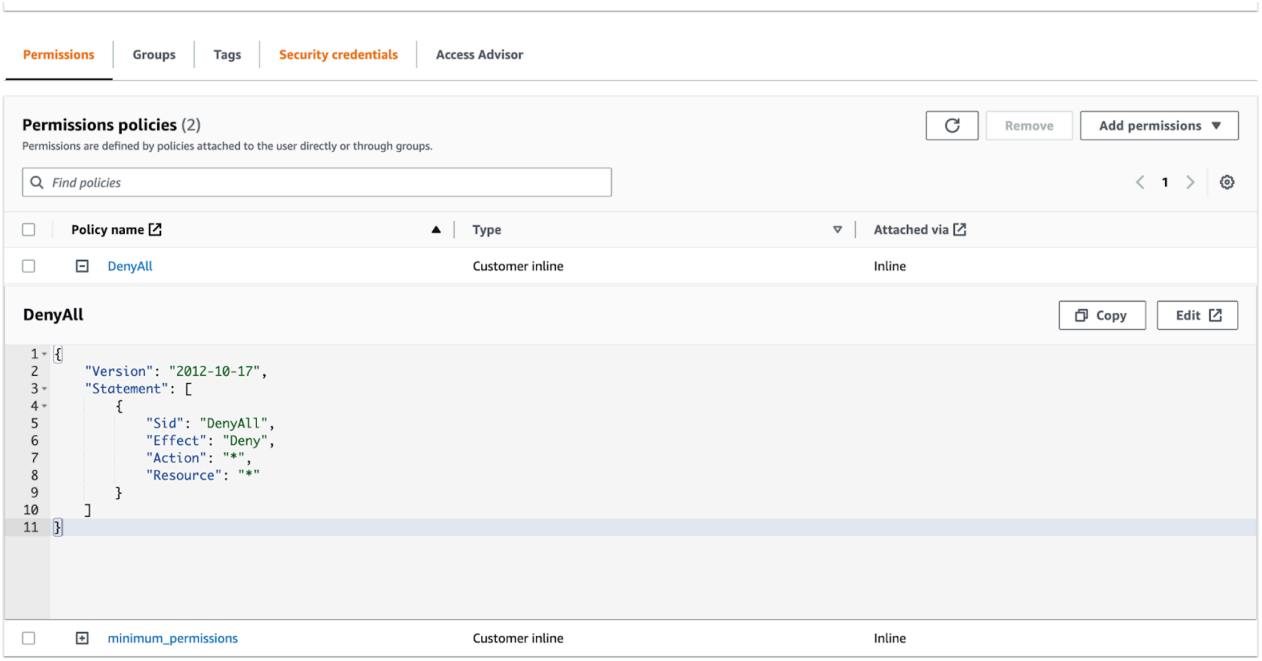

Figure 12. An explicit deny-all IAM policy is attached to the base IAM user (click to enlarge)

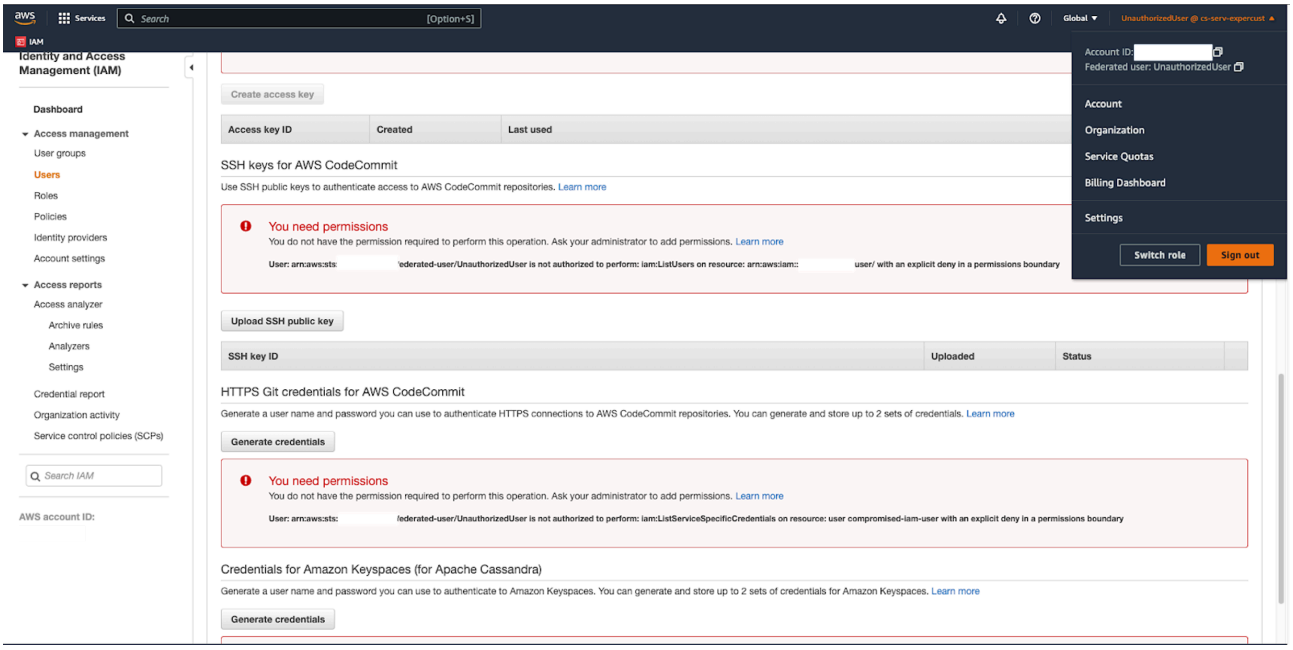

Figure 13. Once the explicit deny-all IAM policy is attached, access is denied (click to enlarge)

Recommendations

CrowdStrike recommends that organizations take the following actions for prevention, detection, response and remediation to protect against this technique.

Prevention

- Create an SCP preventing the use of `sts:GetFederationToken` for all IAM users:

```
{
 "Version": "2012-10-17",
 "Statement": <
   {
     "Sid": "DenyGetFederationToken",
     "Effect": "Deny",
     "Principal": "*",
     "Action": "sts:GetFederationToken",
     "Resource": "*"
   }
 >
}
```

- Exercise good root user hygiene by never configuring API keys for the root user, which is a prerequisite for creating federated sessions from the root user.
- To the extent possible, CrowdStrike recommends avoiding the use of IAM users. Organizations should favor using temporary credentials over long-lived credentials associated with IAM users, by instead using AWS Identity Center to manage access for all human users, and IAM instance profiles for EC2 instance-

backed workloads. This will avoid the use of long-lived AWS API keys within the AWS environment, which is a prerequisite for creating federated sessions from IAM users.

Detection

- Detect and alert on users calling the `GetFederationToken` API call. Here's an example CloudTrail search: `eventName="GetFederationToken" AND eventSource="sts.amazonaws.com"`
- If federated sessions are not used widely in the AWS account or organization, the following CloudTrail search returning all actions taken using federated sessions will return high fidelity indicators of compromise: `userIdentity.type="FederatedUser"`

Response and Remediation

- As shown in the above attack description, deactivating the access key of the compromised base IAM user will not stop abuse using the federated session. Instead, CrowdStrike recommends using an SCP or explicit deny-all IAM policy (described below) to override the permissions assigned to the compromised IAM user. AWS also recently released a detailed blog post on revoking role session tokens generated via an external identity provider that can be referenced for additional remediation strategies.

```
{
 "Version": "2012-10-17",
 "Statement": <
   {
     "Sid": "DenyAll",
     "Effect": "Deny",
     "Action": "*",
     "Resource": "*"
   }
 >
}
```

Summary

CrowdStrike has observed threat actors leveraging federated users in recent investigations, and has identified how they can be used to establish persistence within AWS environments. The use of federated sessions is a method by which an adversary possessing persistent API credentials can generate temporary credentials that survive revocation of the original persistent credentials. CrowdStrike found that these permissions endure until the pre-established session expiration time, even if the credentials associated with the base user are deactivated. The only way to revoke the permissions of an IAM user's federated session is to reduce the scope of permissions assigned to the base user. This can be done by either detaching all policies assigned to the base user, or attaching an explicit deny-all IAM policy or SCP to the base IAM user. Furthermore, federated sessions derived from the root user cannot be contained except through an SCP, and only for AWS accounts that are members of an AWS organization.

Endnotes

1. The default session duration when acting as an IAM user is 12 hours but can be set as high as 36 hours. When acting as a root user, the maximum duration is 1 hour. Learn more.
2. The documentation pages in question are here and here. Both pages state "You cannot call any IAM operations using the AWS CLI or the AWS API." Based on input from CrowdStrike, both pages have been updated to add "This limitation does not apply to console sessions."
3. The CloudTrail events generated from this technique contain the `userIdentity.type` field of `FederatedUser` but it is important to understand that this is not a federated user in the sense of a user who was authenticated in an external directory.

Additional Resources

- *Learn more about how CrowdStrike Services can help your organization prepare to defend against sophisticated threats, respond and recover from incidents with speed and precision, and fortify your cybersecurity practices.*
- *Request a free CrowdStrike Intelligence threat briefing and learn how to stop adversaries targeting your organization.*
- *Watch an introductory video on the CrowdStrike Falcon console and see an on-demand demo of the CrowdStrike Falcon platform in action.*
- *Request a free trial of the industry-leading CrowdStrike Falcon platform.*
