Amazon EC2 Credential Exfiltration: How It Happens and How to Mitigate It

July 31, 2023 [Updated August 4, 2023] :: 9 minute read

@Cloud :: #aws #cloud #security

Table of Contents

An introduction to Amazon EC2 credentials

When you assign an Identity and Access Management (IAM) role to an Amazon Elastic Compute Cloud (EC2) instance, the short-term credentials for the role

are made available via a web service known as the Instance Metadata Service (IMDS). The IMDS provides an HTTP endpoint for retrieving instance metadata such as the instance IP address, AWS Region the instance is running in, the Amazon Machine Image used to launch the instance, and the access key, secret access key, and session token associated with the instance's IAM role. The AWS documentation describes how to retrieve instance role credentials from IMDS. If you've seen or used the `http://169.254.169.254` or `http://fd00:ec2::254` endpoints, then you've seen/used IMDS.

Retrieval of instance role credentials from IMDS is the mechanism by which the AWS CLI and SDKs learn the credentials belonging to the instance's IAM role without you having to configure anything on the instance. Quoting the IAM documentation:

> *The AWS SDKs, AWS CLI, and Tools for Windows PowerShell automatically get the credentials from the EC2 Instance Metadata Service (IMDS) and use them.*

This is great! It means you can start using the AWS CLI, SDKs, or Tools for Windows PowerShell on an EC2 instance without having to configure any credentials.

However, like most nice things, IMDS can be exploited and used in unintended ways. This blog post will explain how EC2 credentials can be retrieved from IMDS, removed from the EC2 instance, and used outside of EC2. This post will also explain some ways to mitigate this activity.

Lifting credentials from IMDS (this is why we can't have nice things)

It's important to understand that IMDS is only reachable from EC2 instances. IMDS is not reachable from the internet, your on-prem network, via AWS Direct Connect, VPN, avian carrier, or any other means. However, when on the instance, the metadata is available to any caller. The EC2 documentation provides the following warning:

> *Although you can only access instance metadata and user data from within the instance itself, the data is not protected by authentication or cryptographic methods. Anyone who has direct access to the instance, and potentially any software running on the instance, can view its metadata.*

IMDS provides the instance role's temporary security credentials in the familiar form of an access key, secret access key, and session token. These are the same pieces of data used by the AWS CLI and SDKs when configuring them to use temporary security credentials.

A well-meaning, but perhaps ill-informed administrator, or worse, an attacker who has gained unauthorized access to an instance can use their access to the instance to talk with IMDS, retrieve the instance credentials, and copy the credentials to either a computer outside of EC2 or to another EC2 instance. The actor can then configure the AWS CLI or SDKs to use those credentials. The external computer/other EC2 instance then gains the same permissions as the EC2 instance the credentials where originally assigned to.

This process of copying the credentials off an instance is called "EC2 credential exfiltration", meaning the EC2 instance credentials have been removed--exfiltrated--from the instance and are being used elsewhere.

The security risk with this activity is that these additional computers/instances can now impersonate the original instance while also using the original instance's credentials to potentially gain additional access in the environment.

Could lifting credentials from IMDS be legit activity?

Update: Jonathan Best made me aware of a commercial product that deliberately moves EC2 instance credentials between EC2 instances. The product's documentation explains this behavior and how to modify it (which begs the question why the default behavior isn't modified to not share instance credentials, but I digress).

The case of the commercial product above notwithstanding, there's no case I can think of where lifting EC2 instance credentials is legitimate activity. The use of exfiltrated EC2 credentials is not sustainable owing to the fact the credentials are short lived; soon after the credentials are issued, they expire and become invalid. A legit use of exfiltrated EC2 credentials would require frequent re-exfiltration and configuration of the other instance(s)/computer(s). It would be more trouble than it's worth.

Sustainable means of getting credentials on another instance/computer are:

1. For another EC2 instance: Assign the instance its own IAM role and grant the role permissions needed by the instance.
2. For computers outside of EC2: Use IAM Roles Anywhere to obtain temporary security credentials in IAM for workloads such as servers, containers, and applications that run outside of AWS.

Detecting EC2 credential exfiltration

Amazon GuardDuty--an intelligent threat detection service that continuously monitors your AWS accounts for malicious activity--will generate a detailed security finding upon detecting EC2 credential exfiltration.

GuardDuty will generate two types of findings based on EC2 credential exfiltration:

1. UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS - GuardDuty has detected credentials issued to a particular EC2 instance being used by an instance in another AWS account.
2. UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS - GuardDuty has detected credentials issued to an EC2 instance are being used from outside of AWS.

GuardDuty is not enabled by default and is not included in the AWS free tier. For more information, see the GuardDuty pricing page.

As with any security finding, understanding root cause is critical. Without that understanding, any remediation or mitigation steps taken can make the situation worse. Be sure to thoroughly investigate the above GuardDuty findings so you can mount an appropriate response.

Mitigating EC2 credential exfiltration

I'm going to mention two methods for mitigating the risk of EC2 credential exfiltration. These methods may not be the only methods, but they are methods I'm familiar with and have experience using.

Method one: restrict where EC2 instance credentials can be used from

IAM has two global condition context keys that can help you write policies to restrict the use of EC2 instance credentials to only the instance to which the credentials were issued. Using an instance's credentials on another instance or from outside of AWS will be denied by such a policy. The AWS blog post *How to use policies to restrict where EC2 instance credentials can be used from* by Liam Wadman and Josh Levinson does a great job explaining how to use these global condition context keys.

I want to highlight a couple of important points from the AWS blog post:

1. When using a policy such as the one shown in the AWS blog post, you must either a/ ensure you have private VPC endpoints for all AWS services you use from your EC2 instance, or b/ exempt services in the policy for which you do not have a VPC endpoint (an example of which is shown in the blog post). If you do neither of these things and your instance reaches an AWS service via its public (i.e., internet) endpoint, this can be detected as an exfiltration, depending on the network path taken to reach the public endpoint. Additionally, the API call will be denied by the policy. If the path to the internet includes traversing a NAT device on-prem, or elsewhere outside of AWS, GuardDuty will see the EC2 instance credentials being used from an IP address that doesn't belong to EC2 and could generate an `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS` finding.
2. The instance's identity role--a special role with ARN `arn:aws:iam::*:role/aws:ec2-infrastructure` used to identify the instance to AWS services--must be exempt from the policy. This ensures the instance can still do things like decrypt its Elastic Block Storage (EBS) volume(s). You should not exempt your instance role(s) in this condition of the policy; leave the `ArnNotLike` condition as-is with just the instance identity role.

Method two: use version 2 of the Instance Metadata Service (IMDSv2)

IMDSv2 uses session-oriented requests. With session-oriented requests, software running on an EC2 instance must first request a token from IMDS before issuing one or more calls to retrieve instance metadata. When using IMDSv2, modern versions of the AWS CLI and SDKs will automatically handle retrieval and use of session tokens.

AWS have designed IMDSv2 to mitigate a number of types of attacks which affect request/response systems such as IMDSv1: misconfigured-open website application firewalls, misconfigured-open reverse proxies, unpatched server-side request forgery vulnerabilities, and misconfigured-open layer-3 firewalls and network address translation. You can read more about IMDSv2 in the AWS blog post *Add defense in depth against open firewalls, reverse proxies, and SSRF vulnerabilities with enhancements to the EC2 Instance Metadata Service*.

The EC2 documentation describes how to enable IMDSv2 and, optionally, how to enforce use of v2.

Can EC2 credential exfiltration be mitigated at the network layer?

Network layer controls such as Security Groups and Network ACLs can not directly mitigate EC2 credential exfiltration, but they can help provide defense-in-depth.

Security Groups and Network ACLs can be used to control inbound communications to EC2 instances. For example, allowing incoming management connections (i.e., SSH, RDP, etc) from expected sources only. Think of this like adopting the principle of least privilege at the network layer.

Neither Security Groups nor Network ACLs can be used to restrict traffic to/from the IMDS endpoint addresses.

You can filter traffic to/from the IMDS using packet filtering software inside EC2 instances. The EC2 documentation describes how to use iptable on Linux, ipfw on FreeBSD, and pf on FreeBSD/OpenBSD. Using ipfw or pf, you can limit access to IMDS to specific Unix users or groups on the EC2 instance. For example, you can allow user `root` to access IMDS and deny all other users.

Another option I ran into while researching this blog post is Colin Percival's imds-filtered for FreeBSD. This software filters communication to the IMDS against a ruleset which defines the path(s) within the IMDS that Unix users/groups are allowed to access. You can think of `imds-filtered` as a layer-7 filter for IMDS whereas iptables/ipfw/pf are layer-3 filters.

Takeaways

1. IMDS plays an integral role in enabling the AWS CLI and SDKs to use EC2 instance role credentials by providing those credentials on demand. Like all credential stores, controls must be put in place to prevent misuse.
2. Consider implementing policy to restrict where EC2 instance credentials can be used from. Ensure you create all necessary VPC private endpoints ahead of time to avoid denying legitimate credential use.
3. Consider enforcing use of IMDSv2. Use the IMDS packet analyzer to understand what, if any, software on your EC2 instance is using IMDSv1 and then take steps to update that software to use v2.

Disclaimer: The opinions and information expressed in this blog article are my own and not necessarily those of Amazon Web Services or Amazon, Inc.

Related posts

- Careful Control of Keys: How I Use MFA with the AWS CLI
  @cloud :: #aws #security
- IAM is the Perimeter
  @cloud :: #aws #security
- AWS ABCs: Granting A Third-Party Access to Your Account
  @Cloud :: #aws #awsabc #security
- 3 Tools for Getting VMs From Your Datacenter to the AWS Cloud

@Cloud :: #aws #cloud #migration

- AWS ABCs -- Can I Firewall My Compute Instances?

@Cloud :: #aws #awsabc #security

:: Theme made by panr