Add defense in depth against open firewalls, reverse proxies, and SSRF vulnerabilities with enhancements to the EC2 Instance Metadata Service | Amazon Web Services

Published: 2019-11-19 · Archived: 2026-04-06 03:14:38 UTC

> **July 27, 2021**: We've updated the link to the 2019 re:Invent session on this topic.

Since it first launched over 10 years ago, the Amazon EC2 Instance Metadata Service (IMDS) has helped customers build secure and scalable applications. The IMDS solved a big security headache for cloud users by providing access to temporary, frequently rotated credentials, removing the need to hardcode or distribute sensitive credentials to instances manually or programatically. Attached locally to every EC2 instance, the IMDS runs on a special "link local" IP address of 169.254.169.254 that means only software running on the instance can access it. For applications with access to IMDS, it makes available metadata about the instance, its network, and its storage. The IMDS also makes the AWS credentials available for any IAM role that is attached to the instance.

When you run applications in the cloud, application security is as critical as instance security; if the applications running on an instance have vulnerabilities or misconfigurations, there can be serious consequences. While application security plays an important role in a layered defense, AWS also constantly evaluates where to add layers, even within the instance, to minimize the damage that can occur when these situations occur.

Today, AWS is making v2 of the EC2 Instance Metadata Service (IMDSv2) available. The existing instance metadata service (IMDSv1) is fully secure, and AWS will continue to support it. But IMDSv2 adds new "belt and suspenders" protections for four types of vulnerabilities that could be used to try to access the IMDS. These new protections go well beyond other types of mitigations, while working seamlessly with existing mitigations such as restricting IAM roles and using local firewall rules to restrict access to the IMDS. AWS is also making new versions of the AWS SDKs and CLIs available that support IMDSv2.

What's new in IMDSv2

With IMDSv2, every request is now protected by session authentication. A session begins and ends a series of requests that software running on an EC2 instance uses to access the locally-stored EC2 instance metadata and credentials. The software starts a session with a simple HTTP PUT request to IMDSv2. IMDSv2 returns a secret token to the software running on the EC2 instance, which will use the token as a password to make requests to IMDSv2 for metadata and credentials. Unlike traditional passwords, you don't need to worry about getting the token to the software, because the software gets it for itself with the PUT request. The token is never stored by IMDSv2 and can never be retrieved by subsequent calls, so a session and its token are effectively destroyed when the process using the token terminates. There's no limit on the number of requests within a single session, and there's no limit on the number of IMDSv2 sessions. Sessions can last up to six hours and, for added security, a session token can only be used directly from the EC2 instance where that session began.

For example, this curl recipe retrieves a session token that's valid for the full six hours (21600 seconds) and then uses that token to access the EC2 instance's profile metadata:

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`

curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadata-token: $TOKEN"
```

Plain text

If you need to write code against the IMDSv2 directly, you can get more detail on the new scheme in the EC2 User Guide.

How these changes add defense in depth

IMDSv2's changes are easy to use, and you'll start using it automatically if you're using the updated AWS SDKs and CLIs. These changes go beyond other types of mitigations to protect against misconfigured-open website application firewalls, misconfigured-open reverse proxies, unpatched SSRF vulnerabilities, and misconfigured-open layer-3 firewalls and network address translation.

Protecting against open Website Application Firewalls

Some Web Application Firewall (WAF) services, such as AWS WAF, can't be configured to act as open WAFs. However, some third-party WAFs can be misconfigured to allow attackers unauthorized access to the network behind the WAF, including the EC2 IMDS.

Many WAFs are designed to act invisibly, so that they can protect websites and applications without administrators having to change or reconfigure the applications that are behind the WAF. To be transparent, WAFs usually pass on all of the headers that come with a request, and do not add their own headers, such as the standard X-Forwarded-For header that other kinds of proxies add. In other words, applications behind a WAF get requests just as the requester sent them.

The AWS approach is to block open WAFs by using a type of request that open WAFs very rarely support, HTTP PUT requests. Although web services such as Amazon S3 use PUT requests for object storage, they're an uncommon type of request for websites and browsers to use. Our analysis of third-party WAF products and open WAF misconfigurations found that the vast majority do not permit HTTP PUT requests. We're using this PUT request to provide a new layer of defense that goes beyond any existing capabilities – we've architected the IMDSv2 service to require a PUT request at the beginning of a session, which will prevent open WAFs from being abused to access the IMDS in the vast majority of cases.

Protecting against open reverse proxies

As it happens, it's also very rare for open reverse proxies to allow PUT requests, but IMDSv2 has another layer of defense against open reverse proxies. Reverse proxies, such as Apache httpd or Squid, can also be misconfigured

to allow external requests that reach internal resources, but it's still normal for these proxies to send an X-Forwarded-For HTTP header. That header itself is used to pass on the IP address of the original caller. IMDSv2 will also not issue session tokens to any caller with an X-Forwarded-For header, which is effective at blocking unauthorized access due to misconfigurations like an open reverse proxy.

Protecting against SSRF vulnerabilities

SSRF vulnerabilities allow attackers to make unauthorized requests from web applications. Since these requests come from the application itself, they can be used to access internal resources that the application has access to but that were not intended to be accessible to outsiders. SSRF vulnerabilities vary in their severity, and some are immune to other types of mitigations. For instance, blocking SSRFs through static headers in instance metadata requests is effective only when the vulnerability merely allows the attacker to control the URL that is being requested; however, AWS analysis found many SSRF vulnerabilities that allow attackers to set arbitrary headers because the SSRF vulnerability impacts the application's own header processing.

IMDSv2's combination of beginning a session with a PUT request, and then requiring the secret session token in other requests, is always strictly more effective than requiring only a static header. AWS analysis of real-world vulnerabilities found that this combination protects against the vast majority of SSRF vulnerabilities.

Protecting against open layer 3 firewalls and NATs

Last, there is a final layer of defense in IMDSv2 that is designed to protect EC2 instances that have been misconfigured as open routers, layer 3 firewalls, VPNs, tunnels, or NAT devices. With IMDSv2, the PUT response containing the secret token will, by default, not be able to travel outside the instance. This is accomplished by having the default Time To Live (TTL) on the low-level IP packets containing the secret token set to "1," much lower than a typical value, such as "64." Hardware and software that handle packets, including EC2 instances, subtract 1 from each packet's TTL field whenever they pass it on. If the TTL gets to 0, the packet is discarded, and an error message is sent back to the sender. A packet with a TTL of "64" can therefore make sixty-four "hops" in a network before giving up, while a packet with a TTL of "1" can exist in just one. This feature allows legitimate traffic to get to an intended destination, but is designed to stop packets from endlessly running around in circles if there's a loop in a network.

With IMDSv2, setting the TTL value to "1" means that requests from the EC2 instance itself will work because they're returned to the caller (on the instance) before the subtraction occurs. But if the EC2 instance has been misconfigured as an open router, layer 3 firewall, VPN, tunnel, or NAT device, the response containing the token will have its TTL reduced to zero before leaving the instance, and the packet containing the response will be discarded on its way out of the instance, preventing transport to the attacker. The information simply won't make it further than the EC2 instance itself, which means that an attacker won't get the response back with the token, and with it the ability to access instance metadata, even if they've been successful at getting past all other defenses.

Making the transition

Both IMDSv1 and IMDSv2 will be available and enabled by default, and customers can choose which they will use. The IMDS can now be restricted to v2 only, or IMDS (v1 and v2) can also be disabled entirely. AWS recommends adopting v2 and restricting access to v2 only for added security. IMDSv1 remains available for customers who have tools and scripts using v1, and who are comfortable with the existing security posture of their instances.

A number of tools are available to make transitioning to v2 and disabling v1 seamless. Starting today, a new CloudWatch metric is available that provides visibility into the number of v1 calls that are being made on any given instance. Customers can use this metric to monitor how often v1 is still being accessed as Amazon Machine Images, the AWS SDKs, CLIs, cloud-init, and other software accessing the IMDS is updated, released, and upgraded. When you can see that an instance can be launched, activated, used for service, and the metric is zero, it is safe to require v2 of the IMDS, disabling v1. For more information on transitioning to IMDSv2, see the user guide.

Security can also be further enhanced while this transition is happening. AWS credentials provided by the IMDS now include an ec2:RoleDelivery IAM context key. Credentials provided by the older IMDSv1 have an ec2:RoleDelivery value of "1.0," and credentials using the new scheme will have an ec2:RoleDelivery value of "2.0." This context key makes it easy to enforce use of the new scheme on a service-by-service or resource-by-resource basis by using those context keys as conditions in IAM policies, resource policies, or AWS Organizations service control policies. For example, if all of the software accessing an S3 bucket has been upgraded to use IMDSv2, then that S3 bucket can be safely restricted to only allow access to role-account credentials that have the "2.0" value (or greater) for the context key. The effect is that credentials retrieved using IMDSv1 will be prevented from accessing the bucket. AWS CloudTrail is also being updated to record the new ec2:RoleDelivery parameters.

Hear about IMDSv2 at re:Invent

Mark Ryland spoke in more detail about IMDSv2, and the transition to it, at AWS re:Invent in December 2019. To watch a recording of the session, see Security best practices for the Amazon EC2 instance metadata service on YouTube.

Want more AWS Security how-to content, news, and feature announcements? Follow us on Twitter.
