Head in the Clouds

By Christopher Maddalena

Published: 2018-09-12 · Archived: 2026-04-06 01:20:52 UTC

Meet the Big Three

In the cloud space, there are three major providers at this time: Amazon Web Services (AWS), Microsoft Azure, and Google Compute. Each provider offers relatively low cost services for computing, storage, load balancing, and more.

There are too many services to cover in one article, so this article will focus on the basics of the computing and storage options, how to track down these cloud assets, and what can be done with assets setup with insecure configurations. This will be a primer for anyone looking to take a closer look at cloud resources from the perspective of a security auditor, penetration tester, or bounty hunter. Informational articles for each provider were made to accompany this article. Links are posted at the bottom of the page.

The IP addresses used by these services is a good a place as any to start.

IP Address Ranges

The IP addresses used by the three cloud providers are all published, but the lists are not all simple to fetch.

Amazon Web Services

Amazon provides the simplest option for fetching their IP addresses by offering a webpage with easily digested JSON:

https://ip-ranges.amazonaws.com/ip-ranges.json

That is all there is to it. The JSON makes it simple to do something like create a script that makes a single web request and parses the JSON.

Microsoft Azure

Microsoft's solution is an XML document. Not as easy as JSON, but trivial to parse with a script. Unfortunately, the document is always changing and must be downloaded from the Microsoft Download Center:

The downloaded document will have a name that includes the date it was last updated, e.g. PublicIPs_20180813.xml. The XML is made-up of Region nodes that name the region, such as "australiac2," and includes the IP address ranges in lines like this one:

```
<IpRange Subnet="20.36.64.0/19" />
```

Google Compute

Google has, by far, the most convoluted solution, which is detailed here in the Google Compute Engine documentation:

To get the IP address ranges, one must first fetch the TXT DNS record for _cloud-netblocks.googleusercontent.com, like so:

> nslookup -q=TXT _cloud-netblocks.googleusercontent.com 8.8.8.8
> Server: 8.8.8.8
> Address: 8.8.8.8#53
>
> Non-authoritative answer:
> _cloud-netblocks.googleusercontent.com text = "v=spf1 include:_cloud-netblocks1.googleusercontent.com include:_cloud-netblocks2.googleusercontent.com include:_cloud-netblocks3.googleusercontent.com include:_cloud-netblocks4.googleusercontent.com include:_cloud-netblocks5.googleusercontent.com ?all"

The returned entries in the SPF record for the "_cloud-netblocks#" names contain the information about Google Compute's current IP address ranges. To get the list, additional lookups must be performed one at a time for each _cloud-netblocks hostname:

> nslookup -q=TXT _cloud-netblocks1.googleusercontent.com 8.8.8.8
> Server: 8.8.8.8
> Address: 8.8.8.8#53
>
> Non-authoritative answer:
> _cloud-netblocks1.googleusercontent.com text = "v=spf1 include:_cloud-

> netblocks6.googleusercontent.com ip4:8.34.208.0/20 ip4:8.35.192.0/21 ip4:8.35.200.0/23 ip4:108.59.80.0/20 ip4:108.170.192.0/20 ip4:108.170.208.0/21 ip4:108.170.216.0/22 ip4:108.170.220.0/23 ip4:108.170.222.0/24 ip4:35.224.0.0/13 ?all"

Making Use of the IP Addresses

An up-to-date list of these IP addresses is useful for identifying assets hosted "in the cloud." If an organization has a domain or subdomain that points back to an IP address in the list it will lead to a storage bucket or cloud server.

Maintaining an Updated Master List

These tasks are all automated in the following script:

The script fetches the latests IP address ranges used by each provider and then outputs all of them as one list in a CloudIPs.txt file. Each range is on a new line following a header naming the service, e.g. "# Amazon Web Services IPs."

With a list of IP addresses on hand, it is time to look at two of the major services that use these addresses.

The Storage Services: Buckets

The common term for a cloud storage container is a "bucket." They are terribly useful things for doing everything from basic file storage to offsite backups and web hosting. They have also been at the center of many a recent information leakage blunder due to how easy they are to misconfigure them and make files available to the public internet that never should have been.

When it comes to seeking out vulnerabilities in the cloud, public buckets take the cake for ease of discovery and potential for high impact.

Common Pitfalls

Today, buckets are private by default and a user must view some warnings before a bucket can be made public. The problem is public buckets are not inherently bad. They are useful for web hosting and file sharing, so a user may legitimately want a bucket to be public. The issue is the access controls can be misunderstood and sensitive documents might be placed inside a bucket where the uploader does not realize they will be made available to the internet at large.

Each service enables users to specify access control lists with users like "allUsers" or "Everyone" to allow public access. The slightly more restrictive options include AWS' "Any Authenticated AWS User" and Google's "allAuthenticated Users." These may sound like they mean authenticated users associated with the account, but they actually mean *any user authenticated with AWS or Google*. These misunderstandings have led to numerous documents made public that never should have been.

Common Bucket HTTP Responses

Each provider returns XML in response to a web request for a bucket and most use the same XML schema. These standardized responses make it possible to do something as simple as brute force web requests to enumerate bucket names and record whether or not they contain publicly accessibly files.

The services return this response when a non-existent bucket is requested:

```
<Error>
<Code>NoSuchBucket</Code>
<Message>The specified bucket does not exist.</Message>
</Error>
```

If the bucket exists, the XML will start with:

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Name>cmaddy</Name>
<Prefix/>
<Marker/>
<MaxKeys>1000</MaxKeys>
<IsTruncated>false</IsTruncated>
<Contents>
…
```

The XML will then display data for any files that are accessible. If the bucket exists but no access whatsoever is allowed, this response is returned:

```
<Error>
<Code>AccessDenied</Code>
<Message>Access denied.</Message>
<Details>
Anonymous caller does not have storage.objects.list access to test.
</Details>
</Error>
```

Note: This also works for enumerating Digital Ocean "Spaces," their service offering for buckets and cloud storage.

The exception to this is Azure. Azure will return XML responses, but requires some additional information in the web request before it will return a resource. This is covered in-depth in the Azure article/cheat sheet linked below.

The Computing Services: Virtual Private Servers

Each provider also offers users the ability to run virtual machines on their infrastructure. The uses range from short-lived VMs stood up for testing to semi-permanent machines used for hosting applications over a longer period of time.

AWS refers to their offering as Elastic Compute Cloud, or EC2. It is common to hear someone refer to “an EC2” when they mean a virtual machine hosted by AWS’ EC2 service.

Google Compute calls their offering Compute Engine.

Azure just calls the service “virtual machines” on the dashboard. Microsoft keeps it really simple.

Common Pitfalls

A common issue with these VMs is they are too often not given the respect they deserve and can be easily forgotten. First, someone may stand up a VM to test a deployment of a new application with some production data to use for QA or user acceptance testing. This VM will be short-lived, so this someone does not bother with securing the application or hardening the VM image. They may not even bother documenting its existence.

They make these decisions, perhaps, because they assume no one will ever find the machine or the application before the machine is shutdown. The problems begin when that assumption is wrong or they forget to kill the machine.

As discussed earlier, the IP addresses of these services are documented and it is best to assume someone is always watching for an opportunity to swoop in and investigate a cloud server running a default configuration.

For the purposes of this article, it suffices to say these VMs may be just like an organization’s internal hosts. They can contain some of the same data and may even be linked to internal networks, except they are less likely to be hardened or monitored like an internal asset.

There are a couple of interesting items that make these VMs different from an ordinary web server in a datacenter.

The Metadata Service

Each cloud service uses a special IP address, 169.254.169.254, with their virtual machines. This address is used for the metadata service, which is documented in these locations:

Azure Instance Metadata Service

The Azure Instance Metadata Service provides information about running virtual machine instances that can be used to…

docs.microsoft.com

The service offers an array of information about the VM instance. This is everything from the hostname to much more sensitive data about the host and the account to which it is attached. For example, executing this curl command on an EC2 host will return security credentials associated with the host, if any:

```
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
```

If the metadata service is somehow exposed, such as via a Server Side Request Forgery vulnerability in an application, an attacker could learn quite a bit about the host and even gain some level of access to a management account by swiping security credentials. Security credentials are covered in the next section.

Snapshots

Users can create snapshots of virtual machines to save as backups. By default, snapshots are private. However, numerous snapshots have been, and continue to be, made public for some reason. Many of these snapshots likely contain nothing of interest, but with each public snapshot there is always the chance someone setup a VM just the way they like it, created a snapshot, and then mistakenly made that snapshot public along with the machine's config files and scripts containing passwords and other sensitive data.

To demonstrate the point, running this command using `awscli` will return all public snapshots available in AWS' us-west-2 region:

```
aws ec2 describe-snapshots --region us-west-2
```

At the time of this writing that command returned 16,126 public snapshots in this one region. These snapshots can be copied, mounted, and then browsed for interesting data. Some useful commands, like the above, are covered in the cheat sheet articles linked below.

Identity Management: Accounts, Keys, and Access

Each service has a root account, the account that has full control over the cloud services. These accounts can then delegate access by issuing access keys and tokens or adding new users to the account.

Each service handles this a little bit differently. The differences between each service are detailed in the individual information articles linked below, but the general idea is to avoid using the root account and make use of new users created for specific purposes.

Compromised Accounts

A compromised account or set of access keys/tokens is a potentially serious issue. This occasionally happens completely by accident. Access keys are often used in scripts and may be accidentally committed to code repositories. Even when that slip-up is caught, the keys may not be revoked or completely removed from the repository (removing the keys and committing that change does not remove them from the git history).

Even if the compromised keys have very limited or read-only access to services, they can still be a boon to an attacker. For example, even though the keys may not be used to access a particular storage bucket, there is nothing to stop someone from listing all available buckets on the account. Basic read-only enumeration can provide a great

deal of information, like the name of a misconfigured bucket containing sensitive data that might have otherwise been nearly impossible to discover or tie to the organization.

Accessing Services: Using the CLI Tools

Each service offers a web user interface and dashboard, but they also all offer their own command line tools. These tools can be used to easily interact with the services, assuming a valid username and password combination or set of keys are available.

These tools are used for managing and interacting with the various available services, like uploading files to a bucket or creating a new virtual machine. A potential downside of using these tools is anyone else with access to an authenticated user's workstation could abuse them to enumerate a cloud environment.

CLI Authentication Abuses

Once a user authenticates, their credentials are locally stored in config and credential files. In some cases, these files can be copied and reused on another machine or the contents can be read and then used to authenticate without needing to copy any files. The specifics of how each service handles authentication are detailed in the individual information articles linked below.

As a proof of concept, this tool was created to search for and collect these files:

chrismaddalena/SharpCloud

SharpCloud – Simple C# for checking for the existence of credential files related to AWS, Microsoft Azure, and Google…

github.com

SharpCloud is a simple, basic C# console application that checks for the credential and config files associated with each cloud provider. If found, the contents of each found file is dumped for collection and reuse.

Conclusion

There is no doubt that cloud services are an amazing resource for individuals and enterprises. The ability to spin up a virtual machine for pennies or make use of a bucket as a cheap off-site backup solution for a home or small office is fantastic and the services have only gotten easier to use over time.

The downside is cloud services seem to have caused everyone to take a step backwards in some aspects when it comes to security. The old idea of “security through obscurity” has made a resurgence with cloud services because it is so easy to assume a cloud server or bucket cannot be linked to a specific organization because they exist within huge net blocks owned by three of the major technology companies.

That is why cloud services demand attention when it comes to security audits, penetration tests, and other security reviews. Hopefully this article has helped shed some light on why cloud servers and storage buckets are not the needles in a haystack they may appear to be at first.

The following links lead to three articles written as companion pieces for this article. There is one for each provider and each one contains provider-specific information and a deeper look at the command line tools and identity management.

Amazon Web Services

Google Compute

Microsoft Azure

Continued Education

Also, the flAWS.cloud website is a great resource for walking through many of the common issues outlined above. It has six stages that each deal with insecure buckets, virtual servers, and snapshots. As the name suggests, flAWS deals with AWS services and misconfigurations, but these lessons easily translate to Compute and Azure.

Post Views: 1,258
