Lesser Known Techniques for Attacking AWS Environments

By Clint Gibler

Published: 2020-01-15 · Archived: 2026-04-05 20:31:21 UTC

This post will discuss lesser known attack techniques that I would use in attacking AWS accounts and conclude with a discussion of defenses. A common theme among many of these concepts is **abusing trust**, whether that is incorrectly trusting attacker controlled resources hosted on AWS, or the trust relationships between accounts or within an account.

I'll discuss a few techniques in gaining initial access, recon, lateral movement between accounts, and data exfiltration.

tl;dr

- **Initial access**: Backdoor community resources (e.g. AMIs, CloudFormation templates, Lambda Layers, etc.) or phish with Stack Sets.
    - **Defense**: Consider using Infrastructure as Code scanning tools to enforce secure defaults and resources that are allowed to be used.
- **Recon** Abuse naming patterns to guess resource IDs (like S3 bucket names) or fingerprint existing roles or services or vendors in use.
    - **Defense**: Follow least privilege so that even resources with known names cannot be accessed unless needed, and consider randomizing resource names.
- **Lateral movement**: Abusing trust and privileges across accounts (IAM, network-level, etc.).
    - **Defense**: Follow least privilege for cross account trust, assess if your cloud security posture has a "soft center," that if an attacker gets inside it's game over.
- **Exfiltration**: Share compromised resources to an account you control to speed exfiltration, or use DNS for stealthy exfiltration.
    - **Defense**: Set up auto-remediation that will automatically unshare resources shared with unknown accounts, and turn on logging for any VPC DNS resolvers. If you want to have an isolated network, consider running your own DNS resolver and disabling the one run by AWS.

About the author

My name is Scott Piper and I do independent AWS security consulting through my company Summit Route, helping in a variety of ways to improve the security of AWS environments for companies, primarily by providing training.

I'm often asked by red teams and pentesting companies what types of AWS attack techniques I would use. There are concepts from actual breaches and public techniques that I might use, but in this post I'll discuss some additional, possibly lesser known, concepts or slight adjustments to known techniques.

Initial access

The idea of backdooring community AMIs was first mentioned at Black Hat 2009 by Nicholas Arvanitis, Marco Slaviero, and Haroon Meer in their talk "Clobbering the Cloud" (slides and video). I investigated a malicious AMI in 2018 that would do Monero mining, and there have been other reports of this issue.

This same concept though can be applied to other resources on AWS, such as CloudFormation templates, Terraform modules, CDK libraries, Lambda Layers, SSM documents, Serverless Application Repository applications, Marketplace resources, container base images, etc.

The concept here is to make one of these resources available for people to use that when used would provide you with access to their account.

As part of AWS's "shared responsibility model" AWS doesn't seem to do any auditing of the resources in any of their "marketplaces", meaning ways in which they host community generated resources.

Getting people to run these might be a little difficult, but an attacker could attempt to target the supply chain by going after the owners of existing popular resources here. We've seen that in the world of Chrome browser extensions where the owners of popular extensions have been phished, or had their extensions purchased and then used to deploy adware. Other marketplaces have encountered similar issues, such as Ruby gems hijacked.

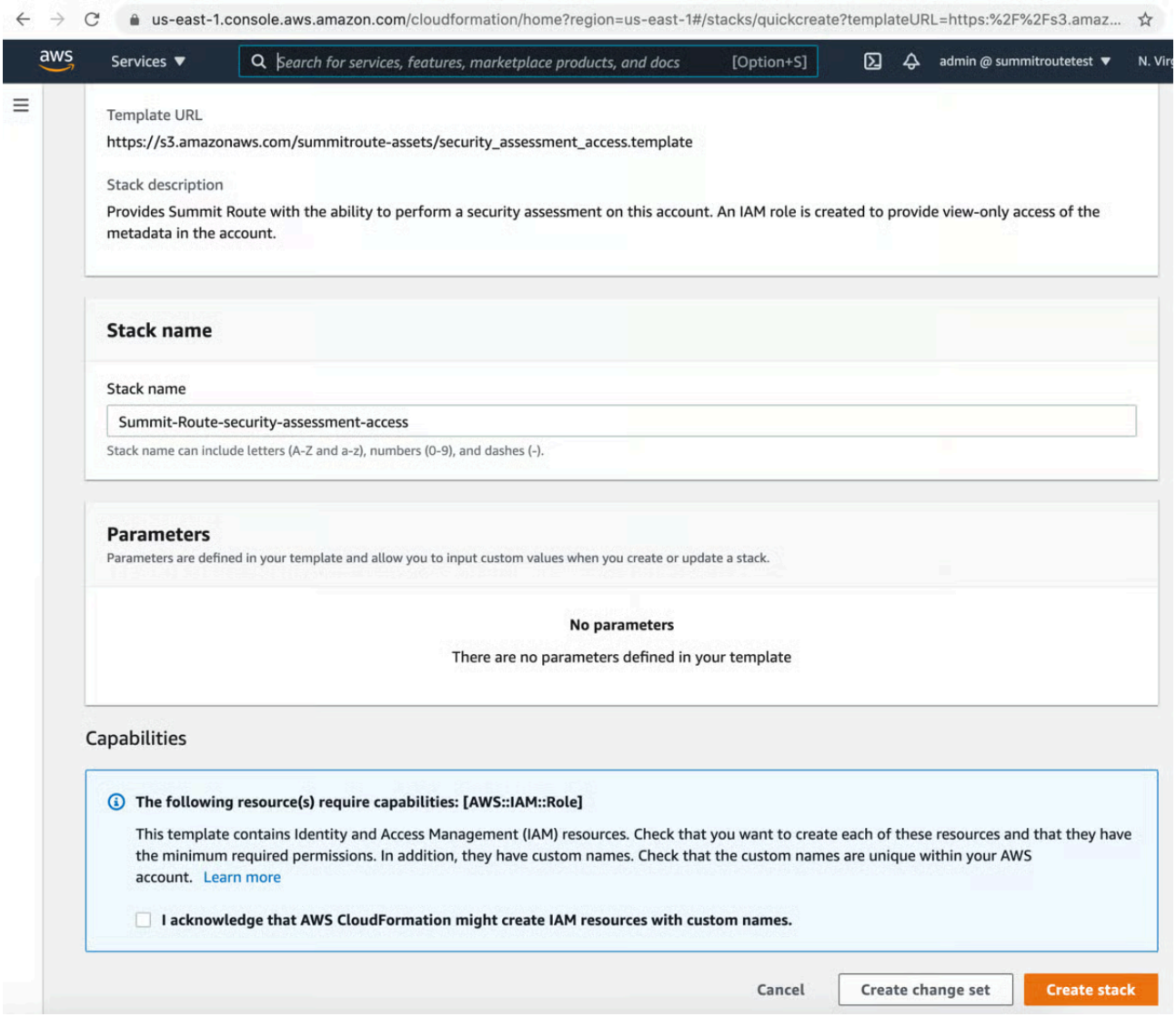
They may also be able to abuse aspects of how search and access works in the marketplace. For example, the Monero backdoored AMI I had investigated was able to find its way into a number of AWS accounts because the AWS CLI and other tooling did not require an `owners` flag be passed, which resulted in AWS randomly picking AMIs that matched the other attributes people specified. This flaw was registered as CVE-2018-15869.

Phishing with Stack Sets

When I do AWS security assessments for clients, I send them a link to a CloudFormation Stack Set. When they click this link, they'll be asked to log into their AWS account if they haven't already. Then they'll be prompted asking them if they want to create resources in their environment. This will create an IAM role that I can assume into that has privileges for me to investigate their account. This makes it very easy for me to gain access to accounts to perform the work I need to do.

An attacker could use this same technique to trick someone into deploying a more privileged role, a backdoored EC2, or other badness. The URL is on the the domain `us-east-1.console.aws.amazon.com` and the referenced Stack Set is hosted in an S3 bucket (here), so everything is on AWS domains and thus for an attacker's goals, this appears trustworthy.

But again, due to the shared responsibility model, AWS doesn't audit any of these.



Stack set creation

Recon

Abusing naming patterns

Ian Mckay wrote a post titled S3 Bucket Namesquatting - Abusing predictable S3 bucket names where he talks about how AWS and vendors frequently use naming patterns for their S3 bucket and how an attacker could abuse this predictably by creating an S3 bucket that uses this naming pattern. This attack is also referred to as bucket sniping.

Ian avoided diving into the specifics of the attack, so to describe this issue further with an example, Athena by default uses the pattern `aws-athena-query-results-ACCOUNTID-REGION`. If an attacker knew your account ID, and you hadn't yet started using Athena, but planned on it, they could create an S3 bucket in their own account that matched that pattern before you did, and either block you from using Athena by denying read/write access to that bucket, or they could open up read/write access to you on that bucket so you'd unknowingly be writing your Athena query results into the attacker's bucket.

Luckily Athena encrypted the results, and around November 2019 they required you to create the bucket as opposed to just attempting to use it. It's likely now with the new s3:ResourceAccount policy condition and the –expected-bucket-owner API condition that opportunities to abuse this concept by creating a public bucket would be reduced further.

However, instead of abusing this concept by creating an S3 bucket, an attacker could abuse it for recon.

For example, let's say the attacker compromises an EC2 that has an IAM role that allows `s3:ListBucket` and `s3:GetObject` on `*`. This would allow them to read any S3 buckets, but they would have to guess the names of the buckets. This is an excellent place for someone to abuse these naming patterns.

As an example, AWS Config by default uses the S3 bucket `config-bucket-ACCOUNTID`. That data then records the names of all the S3 buckets in the account. Therefore by knowing that one S3 bucket name and reading the contents, the attacker could find out the names of all other S3 buckets in the account and read those, along with knowing all the other metadata in the account.

There are other S3 buckets with naming patterns that may be valuable, and there may also be other types of resources.

Service usage recon

In order to determine if AWS Config is being used, you can check for the role associated with it, `AWSServiceRoleForConfig`. If this has been setup from the Organization level, that role will be `AWSServiceRoleForConfigMultiAccountSetup`. If they run GuardDuty, there will be a role named `AWSServiceRoleForAmazonGuardDuty`.

It is possible to identify the existence of IAM roles in one account from another account by creating an IAM role trust policy and attempting to reference a role in another account and seeing if it errors. This concept was first discovered by Daniel Grzelak and presented at Kiwicon in 2016 (video, slides, code and wordlist). That video is hilariously bad filming as it was done via a cellphone from someone in the audience and broken up into 6-minute clips. This concept is more thoroughly explained in Spencer Gietzen's post Unauthenticated AWS Role Enumeration (IAM Revisited) and is incorporated into pacu.

Once you identify that certain AWS services, vendors, or maybe just popular CloudFormation templates have been deployed in an account, it may help you narrow in on resources associated with those that use default names or naming patterns. By knowing those services are used in advance, you could avoid having Access Denied errors recorded in CloudTrail, which are more likely to generate alarms.

Lateral movement between accounts

Companies are creating more and more AWS accounts for themselves, in large part due to separate accounts being a strong security boundary. However, they then interconnect those accounts heavily, which can blur or erase those security boundaries.

If an attacker had admin access inside one account, they could look to see what accounts it knows about and if they can access them. They could use CloudMapper's weboftrust command to try to figure some of this out. That

command is mostly for finding what accounts the account being assessed trusts to access it though, and not which external accounts trust it.

Knowing who trusts your account can be valuable though. CloudMapper's `weboftrust` and the AWS service Access Analyzer can show which resources in my account are trusted by other accounts and I would like to extend this further, looking at things like shared AMIs etc.

For an attacker, this could enable RCE into the other accounts. For example, I've seen one company where they had an S3 bucket that hosted a bash script that every EC2 at the company was supposed to run at boot. So if you modified that bash script you'd get RCE on every EC2 across every account at the company.

An attacker could look at the IAM privileges in an account to see if they are granted access to other accounts. They could also look in CloudTrail logs, specifically CloudTrail Event History (since that can be accessed even if the S3 bucket where logs might be sent can't be), and look to see where things in the compromised account are assuming roles to. This is discussed further in my post Lateral movement between AWS accounts - Abusing trust relationships.

An attacker could look at what resources are referenced from one account, but aren't part of that account. For example, companies frequently have CloudTrail logs all sent to a single S3 bucket. Sometimes those companies then make those logs easily accessible to all accounts, by having a bucket policy that includes a condition such as:

```
 "Condition" : { "ForAnyValue:StringLike" : {
    "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/*"]
}}
```

This will likely allow you to read the CloudTrail logs for not only your own account, but all AWS accounts that are part of the organization. Using these logs, an attacker could start to understand the other AWS accounts.

An attacker could look at networking trusts, such as transit gateway, VPC peering, etc. to see what networks trusts the compromised account to again move laterally.

From the concepts above, an attacker should have identified other AWS accounts in the organization. Often organizations will grant trusts between other AWS accounts within the organization such that an attacker could attempt to brute force assume into IAM roles in the other accounts.

Spencer Gietzen discussed the idea of brute-forcing IAM roles that were publicly accessible in his post Assume the Worst: Enumerating AWS Roles through 'AssumeRole', but this technique would be more effective for attempting to move laterally between the accounts in an organization.

Data exfiltration

Resource sharing

Once an attacker identifies resources of interest, it is often in their best interest to get that data out of the account as quickly as possible because remaining inside the account increases the likelihood of their detection.

In order to move the data out of the account as quickly as possible, sharing the resources to another account is a technique that can be used.

For example, if an attacker finds a lot of RDS snapshots, instead of attempting to spin up an RDS inside the account, they could share those snapshots to another account. It is advantageous to an attacker to have other victim AWS accounts for this purpose, which could potentially be obained using the concepts of “Backdooring community resources” discussed earlier.

I’ve heard the idea of making the resources public for this purpose, but I believe that increases the likelihood of the activity being detected, because it is more common to identify resources being made public than being shared with another account. Finding a resource is public is likely to be a high severity alert, but finding it shared with an unknown account is likely to be lower severity.

DNS

Companies that try to control their network traffic often overlook the VPC DNS resolver. In my post on Isolated networks on AWS I discussed how this can be abused to exfiltrate data in environments that have restricted network traffic.

Since I wrote that post, AWS has now made those logs accessible, but I believe it is still fairly rare for companies to be recording those.

Defenses

Initial Access

The initial access section discusses ways of abusing community resources. Having Infrastructure as Code processes where code can be reviewed by another person can help. That code can also be scanned by static analysis tools.

The Stack Set phishing concept could be mitigated by blocking all CloudFormation scripts except for ones that come from buckets you own, by using a policy such as the following:

```
{
 "Version":"2012-10-17",
 "Statement":[
   {
     "Effect" : "Deny",
     "Action" : [ "cloudformation:CreateStack", "cloudformation:UpdateStack" ],
     "Resource" : "*",
     "Condition" : {
       "ForAllValues:StringNotLike" : {
         "cloudformation:TemplateUrl" : [ "https://s3.amazonaws.com/mybucket/*" ]
       }
     }
   }
```

```
  ]
}
```

Recon

The idea of abusing naming patterns is best mitigated by practicing a least privilege strategy to ensure that even resources with known names cannot be accessed. There may also be some benefit in vendors and those creating popular resources to take steps to randomize some of their names. AWS especially should consider this.

Lateral movement

The concepts for moving laterally are also best mitigated through least privilege strategies for the cross account trusts. You should also consider the threat of what happens when an attacker compromises one of your AWS accounts and whether you have a "soft center" security model where onces something gets inside, it becomes inherently trusted.

Data exfiltration

For data exfiltration, the idea of sharing resources is best mitigated by having auto-remediation solutions that can unshare resources that are being shared with unknown accounts, and followed up quickly by incident responders. This autoremediation solution and incident response roles should be protected by Service Control Policies (SCPs) so that an attacker could not disable those.

AWS should expand Access Advisor's coverage to all resources that can be shared or made public so that people would be more motivated to use that service and integrate it into things. Currently that service is limited, with coverage for only about 6 resource types when I believe there are at least 24 (see my aws_exposable_resources repo).

For data exfiltration via DNS, you should turn on logging for any VPC DNS resolvers. This may help you detect technique being used, but if you want to have an isolated network, you should look into running your own DNS resolver and disabling the one run by AWS.

Tooling we need

I believe there is still a lot that can and should be done to understand the interconnections between accounts and I'd love to see more tooling there.

Primarily, I'd like to better visualize these trust relationships, and do so for all the various resource types that can be configured for these trusts. If Access Advisor gets better resource coverage, people could build on the data from that service.

I'll close with one of my favorite quotes about AWS security, which comes from Marco Slaviero in 2017😀

"The number of known attacks against AWS is small, which is at odds with the huge number (and complexity) of services available. It's not a deep insight to argue that the number of classes of cloud specific attacks will rise."

Marco Slaviero

Editor's note: Scott is one of my favorite people to follow for the latest and greatest in AWS security. I *highly* recommend checking out his blog or training if you want to level up your AWS skills. He also created flaws.cloud, an interactive tutorial/CTF for learning common AWS security mistakes.
