AWS Security Maturity Roadmap

Written by Scott Piper <scott@summitroute.com>

SummitRoute.com

January 2021

Introduction

Who am I and who is this for

I've worked as an independent AWS security consultant for the past three and half years for my own company Summit Route[1] helping over thirty companies, ranging from small start-ups with barely a footprint in AWS to Fortune 100s with hundreds of AWS accounts, from companies that did a lift and shift of large numbers of EC2s and have large footprints in datacenters to companies who exclusively use serverless, from companies that have been in AWS for a decade when it was first coming out to those just getting started with AWS.

I've performed security assessments, consulted on best practices, written custom code, performed research, and more. Currently, I focus on giving training to companies. I developed the popular (and free) AWS security CTFs flaws.cloud[2] and flaws2.cloud[3], and developed the open-source tools CloudMapper[4], CloudTracker[5], and Parliament[6] with Duo Security.

I have no affiliation with Amazon, but I only do work related to AWS security. This is my opinionated actionable guide to using AWS securely in 2021.

Keeping up-to-date

Some of what I write here will unfortunately become out-of-date. To stay up with what is happening in AWS security, I recommend the following:

- ☐ Watch the talks from re:Invent, re:Inforce, and the independent non-profit conference fwd:cloudsec.
- ☐ Subscribe to the weekly newsletters Last Week in AWS[7] by Corey Quinn, CloudSecList[8] by Marco Lancini, and tl;dr sec[9] by Clint Gibler.
- ☐ On Twitter, follow @0xdabbad00 (my twitter account), @SummitRoute (my company), @jeffbarr (Chief evangelist for AWS), @AWSSecurityInfo (official AWS security twitter account), @StephenSchmidt (AWS CISO), and @jim_scharf (VP of AWS Identity).
- ☐ Join the Slack "Cloud Security Forum". Ask myself or anyone else doing AWS security things for an invite.

Learning the foundations

This paper is mostly about current best practices, which requires you to know a lot of foundations of AWS and security in general.

The best book for learning about security concepts you'll likely use in your work around AWS is Securing DevOps: Security in the Cloud by Julien Vehent, who managed Firefox Operations Security. For some hands-on training, flaws.cloud and flaws2.cloud will show you some important concepts.

If you want to get a cert in order to give yourself a goal to work toward, and proof that you have some knowledge in the area, the only cert I recommend getting is the AWS Security Specialty[10] certification. To prepare for the exam, you can take the acloud.guru[11] course and read the various white-papers from AWS along with the FAQs for security related AWS services. For companies, I also offer AWS security training.

[1] https://summitroute.com
[2] http://flaws.cloud
[3] http://flaws2.cloud
[4] https://github.com/duo-labs/cloudmapper
[5] https://github.com/duo-labs/cloudtracker
[6] https://github.com/duo-labs/parliament
[7] https://lastweekinaws.com/
[8] https://cloudseclist.com/
[9] https://tldrsec.com/
[10] https://aws.amazon.com/certification/certified-security-specialty/
[11] https://acloud.guru/learn/aws-certified-security-specialty

Mine is not geared toward the certification, but will help. There are other AWS offered certs which you may also wish to get in order to learn the foundations of AWS.

Most common security incidents

In order to discuss the steps to secure AWS environments, you should be aware of what the common security incidents on AWS are so you can plan your defensive strategies accordingly. Generically, the most common AWS related incidents are:

1. Publicly accessible resources such as S3 buckets or ElasticSearch clusters.
2. Leaked access keys. For example, access keys posted to GitHub.
3. Compromised IAM Roles through SSRF or RCE against an EC2, resulting in access to the metadata service at 169.254.169.254.

There are many other issues possible, and many issues that are not AWS specific, but I believe extra effort should be spent in addressing these. My goals are to reduce the likelihood and impact of those events and others, and to enable you to detect and respond to incidents, or misconfigurations before they become incidents.

AWS Security Maturity Roadmap

I've grouped security goals into stages that somewhat build on each other. These are the steps to take your company from having no cloud security program for AWS to having a solid program. You may find you do some of these steps in a different order, partially complete steps for some accounts before others, or have to go back and make slight adjustments, but this should help give you a roadmap for making progress.

Stage 1: Inventory

- ☐ Identify all AWS accounts in the company and their points of contact.
- ☐ Integrate AWS accounts into AWS Organizations.
- ☐ Ensure all account root emails are on distribution lists.
- ☐ Opt-out of AI services using your data.
- ☐ Have an AWS account for Security.
- ☐ Create budget alarms.

In this phase, you will want to create a spreadsheet of all the AWS accounts you have (name and 12-digit account ID), their point of contact, associated payer, etc. You can also use this spreadsheet to keep track of which accounts have met certain goals that will be discussed in this paper by including additional columns. See the post How to inventory AWS accounts[12] for more information on how to inventory your AWS accounts. You will want to ask the different groups at your company what accounts they have, the finance team what accounts are being paid for, talk to your TAM if you have one to figure out what accounts have been registered under your company email, search your company emails, and possibly search network logs.

If your accounts are not already in an AWS Organization, you will want to do so. The root for the AWS Organization should not have any AWS resources in it (ie, no EC2's, S3 buckets, etc.). By putting the accounts in an Organization you will get consolidated billing, an easier ability to create new accounts, and ability to use SCPs and other Organization features (discussed later).

The root email is the primary way AWS communicates with you and is used for recovering the account. You want to ensure the root account emails are associated with email addresses under your company domain, and that these are associated with email distribution lists to ensure communications from AWS are seen.

Your data on AWS generally stays in the regions you put it in and generally AWS does not use that data for their own purposes or send it to affiliates. That is not true if you use the AWS AI services, unless you opt-out of this data usage, which you should do via functionality[13] of Organizations.

Instead of the spreadsheet mentioned earlier, some companies are able to use tags in Organizations to keep track of many of the important datapoints about their accounts. If you have multiple AWS Organizations though, such as after an acquisition, you will likely still want to use a spreadsheet.

You should have an AWS account used by the Security team that will be used for log collection. If you don't already have lots of AWS accounts, recognize that at this point you now have a minimum of 3 AWS accounts (an account that has your business stuff in it, the AWS Organization root account with nothing in it, and the Security account).

For individuals and small businesses, budget alarms can help avoid catastrophic bills of thousands of dollars. For larger enterprises that are already spending millions per month in AWS, these will be less valuable. Larger enterprises may get more value out of Cost Anomaly Detection[14] alerts instead. Commonly when attackers compromise accounts they will spin up EC2s to mine bitcoin. However, mistakes that result in infinite loops,

[12] https://summitroute.com/blog/2018/06/18/how_to_inventory_aws_accounts/
[13] https://summitroute.com/blog/2021/01/06/opting_out_of_aws_ai_data_usage/
[14] https://aws.amazon.com/blogs/aws-cost-management/preview-anomaly-detection-and-alerting-now-available-in-aws-cost-management/

or just having resources that are not removed after use are common problems encountered on AWS. In dev or test accounts, you could apply Budget Actions[15] to apply SCPs to, roughly speaking, disable the account.

Stage 2: Have backups

- ☐ Create regular backups.

One of the most notorious AWS security incidents was the destruction of a business named Code Spaces. An attacker gained access to their account and destroyed all of their data and backups, resulting in the business shutting down within 12 hours of the incident.

As you learn what data you have in AWS, you should ensure that data is backed up to a separate AWS account in a separate region from where the data comes from (or possibly even somewhere off of AWS). You can easily create backups with the AWS Backup[16] service and S3 replication policies[17]. You can ensure WORM backups with S3 Object Lock[18], in addition to using a separate AWS account for them.

Planning for Disaster Recovery involves a trade-off between how quickly you need to recover (Recovery Time Objective) and how recent that data needs to be (Recovery Point Objective) versus how much money and effort you are willing to spend. You should consider using Glacier Deep Archive[19] for some data which is $1/TB/mo to store data, but will take up to 12 hours to retrieve.

Stage 3: Visibility and initial remediation

- ☐ Turn on CloudTrail, GuardDuty, and Access Analyzer for all accounts to send their logs and alerts to the Security account.
- ☐ Create an IAM role in every account that grants view access into the account from the Security account.
- ☐ Run a one-time scanning tool to identify tactical remediations.
- ☐ Turn on S3 Public Block Access.
- ☐ Develop an account initialization script and new account creation process.

CloudTrail logs should be turned on via the Organization root via an organization trail[20]. By using the organization trail you not only have a centrally configured CloudTrail that is enabled by default for all newly created accounts in the Organization, but also these CloudTrail logs cannot be turned off by the child accounts. You will create an S3 bucket in the Security account, and then from the Organization Management account, enable the CloudTrail logs. You will need to setup permissions on this bucket for your organization to send logs to it.

GuardDuty and Access Analyzer can also now be enabled across an entire AWS Organization with a few clicks in the Organization Management account to connect all accounts to a Security account, via the Delegated Admin[21] concept. For now, just enable this integration so you can manually check it, and we'll set up alerting on this later. Another service, Macie, also allows this concept, so consider turning on this service as well, although the cost/benefits trade-off is less definitive. Macie's primary use case is for identifying your sensitive data in S3. It has S3 monitoring abilities, but GuardDuty can also be configured[22] to perform that service. This needs to be done in all regions you are active in (we'll block off other regions via SCPs later).

[15] https://aws.amazon.com/blogs/aws-cost-management/get-started-with-aws-budgets-actions/
[16] https://aws.amazon.com/backup/
[17] https://docs.aws.amazon.com/AmazonS3/latest/dev/replication.html
[18] https://aws.amazon.com/about-aws/whats-new/2018/11/s3-object-lock/
[19] https://aws.amazon.com/about-aws/whats-new/2019/03/S3-glacier-deep-archive/
[20] https://docs.aws.amazon.com/awscloudtrail/latest/userguide/creating-trail-organization.html
[21] https://summitroute.com/blog/2020/05/04/delegated_admin_with_guardduty_and_access_analyzer/
[22] https://aws.amazon.com/blogs/aws/new-using-amazon-guardduty-to-protect-your-s3-buckets/

Create an IAM role in every account that grants access for the Security team. You can make a copy of the CloudFormation template I use when I perform assessments (be sure to change the account ID!) and you'll need to ask people what the account ID is for the accounts they ran this in. See here[23] for how the URL is formed and the template file that allows people to easily grant you access. Minimally you want SecurityAudit and ViewOnlyAccess privileges, but may want more in order to do auto-remediation or incident response.

Once you have access, you'll want to run a one-time scanning tools such as CloudMapper[24], Prowler[25], or NCC's ScoutSuite[26]. This will give you a feel for how much work lies ahead. These will help point out tactical issues that you can work on fixing now, but later phases in this maturity model will help strategically stop these issues from happening in the first place. For now, the main things you should worry about are any public S3 buckets with sensitive contents, public AWS managed ElasticSearch servers, or any EC2's running publicly accessible services that should be private, such as ElasticSearch again, databases, or other "soft" targets, which can largely be identified by the port numbers that have been made public.

The feature S3 Public Block Access[27] should be turned on in accounts which disables buckets and their objects from being made public. For cases where you do still need some S3 buckets public, this can be configured to ensure no new buckets are made public.

As you granted access to the accounts for the Security account, you should use this as an initialization script for future accounts so that you have a common, secure baseline, that new accounts you create, or old accounts that are newly acquired, can conform to. You should document a process for creating new accounts. What team should do that and for what needs? Where should the root password and MFA (Multi-Factor Authentication) be kept? What email address pattern should be used for the account? Who should be on the distribution list of that email address? What phone number and billing information do these accounts tie back into it?

You should use CloudFormation StackSets with Organizations[28] in order to establish this baseline. This functionality ensures that a CloudFormation template is deployed whenever a new account is created.

Stage 4: Detection

- ☐ Send alerts to a ticketing system.
- ☐ Enable investigations to logs.
- ☐ Perform regular scanning of the accounts for security issues.
- ☐ Document your security guidelines for your company.
- ☐ Consider turning on other logging sources.

With CloudTrail logs going to an S3 bucket in the Security account, and GuardDuty and Access Analyzer events arriving in that account as well, you should integrate all these things into a monitoring and alerting system. As a first step, most companies will create a CloudWatch Event rule that sends the alerts directly to an SNS that goes to email or Slack messages. They then quickly realize they need to filter this through a Lambda or something with logic in it in order to ignore certain types of alerts. As their alert response processes mature, they also find they need to send these to a ticketing system. This enables them to track metrics, have an audit history, and perform escalations.

When an alert happens, you will want to dig in to understand other relevant events from the logs. Minimally, you can setup Athena to be able to query your logs in S3. This can be done using cloudtrail-parquet-glue[29].

[23] https://summitroute.com/aws_security_assessments/
[24] https://github.com/duo-labs/cloudmapper
[25] https://github.com/toniblyx/prowler
[26] https://github.com/nccgroup/ScoutSuite
[27] https://aws.amazon.com/blogs/aws/amazon-s3-block-public-access-another-layer-of-protection-for-your-accounts-and-buckets/
[28] https://aws.amazon.com/blogs/aws/new-use-aws-cloudformation-stacksets-for-multiple-accounts-in-an-aws-organization/
[29] https://github.com/alsmola/cloudtrail-parquet-glue

Another minimal solution is to send the logs to CloudWatch Logs. However, many use vendor solutions for their log investigations. Almost all developers that use AWS will need access to CloudTrail logs for the accounts they work in, so you may have a mix of these solutions.

You want to ensure you can minimally do the following:

- Receive a notification about a GuardDuty alert from any of your accounts and any region you enabled it in.
- Be able to alert on Access Denied errors from CloudTrail logs, specifically for your production accounts.
- Be able to search through CloudTrail logs to see all of the actions performed by a principal during a time period.

Some best practices can't be detected based purely off of a log event, for example detecting an IAM role that has been inactive for over 90 days. Your event detection solution may also break down at some point, or you'll want to add new rules that accounts might already have violated at some point in the past. For these reasons, you'll want something to perform regular scanning of your environments. Minimally, this can be done with CloudMapper[30], but AWS Config and the associated rules should also be considered, as well as vendor solutions. AWS Config now supports[31] a delegated admin concept.

Now that you are detecting various issues, you'll want to document what those issues are so you are not just telling your team they did something wrong without them knowing the rules you want them to play by. Many of the vendor and open-source solutions in this space detect things that you will not care about. For example, you might have accepted the risk of things like a bastion host that allows SSH access globally (although at some point you should try to switch that to use Systems Manager Session Manager). There are also things that out of the box these tools don't detect that are specific to your environment that you want done. For example, you might want all public network resources to only be in a few specific accounts and subnets, or that all S3 buckets that contain certain types of data should be replicated to another specific account and region to ensure Disaster Recovery and fail-over. You should work toward building these additional checks into your detection solution.

Consider turning on more logging sources, such as the following, plus any others you might be using (full list of known log sources on AWS is here[32]):

- VPC DNS queries[33]
- VPC Flow Logs (These are not as useful as many assume)
- CloudTrail S3 access and Lambda invokes
- S3 access logs
- Load balancers
- CloudFront

Stage 5: Secure IAM access

- ☐ Use SSO for access.
- ☐ Remove all IAM users.
- ☐ Remove all unused IAM roles.
- ☐ Reduce the privileges of service roles to necessary services.
- ☐ Implement pre-commit hooks for secret detection.
- ☐ Plan how accounts will be connected.

[30] https://duo.com/blog/continuous-auditing-with-cloudmapper
[31] https://aws.amazon.com/about-aws/whats-new/2020/11/aws-config-supports-organization-wide-resource-data-aggregation-delegated-administrator-account/
[32] https://matthewdf10.medium.com/how-to-enable-logging-on-every-aws-service-in-existence-circa-2021-5b9105b87c9
[33] https://aws.amazon.com/blogs/aws/log-your-vpc-dns-queries-with-route-53-resolver-query-logs/

IAM User Access Keys never expire. They regularly wind up in source code that finds its way onto public GitHub repos and when configured to be used by the AWS CLI, they exist as plain-text in `~/.aws/credentials`, which poses risks. For these reasons, you should avoid IAM Users entirely and instead use IAM Roles.

For human users, you should use SSO for access. This gives you a central location for creating and removing users, or rolling their credentials if they are compromised. You can either have a single "Identity" AWS account that users log into via SSO and then assume into roles from there to the different accounts, or you can have them SSO directly into the different AWS accounts they have access to[34]. Solutions exist for different providers for working from the command-line with SSO, including the AWS CLI v2.

If you do have requirements to use IAM access keys, for any human users, they should also be using aws-vault[35]. If possible, for all access keys, you should create a single "Identity" account that they will then assume roles from in order to access other accounts. This ensures you have only a single account with IAM access keys and allows you to setup alerts if the access keys are used without assuming roles into other accounts.

You should audit your IAM Users and Roles to identify ones that have not been used for a certain amount of time. For the remaining ones, use Access Advisor to reduce the privileges of the IAM roles to only those services they use. Ideally, you should reduce this further to only the necessary Actions and Resources, but that is a harder problem, we'll aim for later. For now, the main thing you want to do is find roles with admin (or near admin) privileges, especially when these are associated with EC2s.

Using Access Advisor data, there are some additional S3 privileges you can get information about. You can identify who has used `s3:ListAllMyBuckets` and you should remove that from any applications that are not using it. Very few applications should need that privilege.

You should implement a solution to detect secrets being committed to source-code repos to avoid having IAM User access keys added to public, or even private, repos. One solution is Yelp's detect-secrets[36] project.

As you deploy SSO, you should give thought to how accounts will be accessed and how they will be connected. In addition to humans accessing your accounts, you also have applications that need access to resources across different accounts. This includes resources such as IAM roles and S3 buckets that allow you to share access with other accounts, and network resources that you'll need to connect using VPC peering, transit gateways, etc. You should give thought to the ways in which you want accounts to be connected, especially as you accumulate more and more accounts. Separate accounts are a strong security boundary, but this can become blurred and complicated as trust relationships are created between accounts. You should document a plan for when and how new accounts will be created and connected.

Stage 6: Reduce attack surface and mitigate compromises

- ☐ Apply SCPs.
- ☐ Have no publicly facing EC2s or S3 buckets.
- ☐ Enforce IMDSv2 on all EC2s.

SCPs (Service Control Policies) can be used to restrict actions an account can perform. See examples in my post AWS SCP Best Practices[37]. You can make exceptions so only a special IAM role, such as that used for Stack Set deployments, can bypass the restrictions. You can apply different SCPs to different accounts and these can ensure that no one, not even the root user, can perform certain actions.

The SCPs to apply should include:

- ☐ Deny root user access.

[34] "Identity federation with multiple AWS accounts" by Alex Smolen - https://medium.com/@alsmola/identity-federation-with-multiple-aws-accounts-61065d00e461
[35] https://github.com/99designs/aws-vault
[36] https://github.com/Yelp/detect-secrets
[37] https://summitroute.com/blog/2020/03/25/aws_scp_best_practices/

- ☐ Allow only approved regions.
- ☐ Allow only approved services.
- ☐ Deny ability to create IAM access keys.
- ☐ Require the use of IMDSv2.
- ☐ Deny ability to leave Organization.
- ☐ Deny ability to make a VPC accessible from the Internet that isn't already for specific accounts.
- ☐ Deny ability to disrupt GuardDuty, Access Analyzer, CloudTrail, S3 Public Block Access, and other security services.
- ☐ Deny ability to disrupt CloudWatch Event collection or other aspects of your monitoring and alerting pipeline.
- ☐ Deny ability to modify important IAM roles, such as one used for Stack Sets, incident response, or vendors performing monitoring.

Resources such as EC2s and S3 buckets should not be publicly accessible, but instead should have an ELB or CloudFront in front them. This has a number of benefits including being able to make use of AWS Shield (DDoS protection), AWS WAF, and a more standardized means of logging access. This has operations improvements such as an ability to scale and reduced latency. CloudMapper's `report` and `public` commands will show you what public resources you have.

You may also at this point consider deploying AWS WAF and paying for Shield Advanced.

The way in which EC2s obtain their credentials for their IAM roles is through the Instance MetaData Service (IMDS). If an attacker can get the EC2 to access this service and return the results (such as through SSRF or a proxy service), they can take-over the IAM role. You should ensure your EC2s enforce the use of the newer IMDSv2[38]. This can be done incrementally, until ultimately you set an SCP to enforce this.

Stage 7: Reproducibility and ownership

- ☐ Use Infrastructure as Code.
- ☐ Control AMI and package sourcing.
- ☐ Apply tagging strategy

You should control how changes to your AWS environment happen. Changes should only be made through Infrastructure as Code (IaC), such as Terraform, CloudFormation, or the CDK. This has a number of benefits such as ensuring you have a reproducible environment and can better audit changes. This is often done gradually, beginning with the VPC, Security Group, and IAM configurations, and eventually encompassing all resources. The tool former2[39] can assist with transitioning your environment to IaC.

Once you control infrastructure changes through code, you can also use two-person rule deployments, where one person proposes the changes and another signs off in order for the change to be implemented. You can also use tools to scan these proposed changes before they are deployed. Options for doing this are described here[40].

Not all of the public AMIs (Amazon Machine Images) used to create EC2s are vetted by AWS, and there is not an easy way to identify the trust-worthiness of the source account ID that an AMI comes from. There have been issues[41] in the past of public AMIs with malware in them. You should identify what AMIs you want to be allowed to run in your environment. You should also document guidelines around the instance types to be used and the OS's, so you don't end up with a a dozen flavors of Linux running in your accounts without good reason. Similarly, you should do this for any lambda layers, container images, and other resources.

[38] https://aws.amazon.com/blogs/security/defense-in-depth-open-firewalls-reverse-proxies-ssrf-vulnerabilities-ec2-instance-metadata-service/
[39] https://aws.amazon.com/blogs/opensource/accelerate-infrastructure-as-code-development-with-open-source-former2/
[40] https://blog.christophetd.fr/shifting-cloud-security-left-scanning-infrastructure-as-code-for-security-issues/
[41] https://summitroute.com/blog/2018/09/24/investigating_malicious_amis/

One strategy you should consider is building all of your AMIs yourself, possibly with all of their code and packages pre-configured. This would mean that every code release would be a pre-built AMI, which is the strategy used by Netflix in their post How We Build Code at Netflix. Alternatively, you can deploy code and updates using something like salt, ansible, puppet, chef, or another solution.

You should control the packages (code and software) that go into your code bases. Your production servers should not be reaching out to various GitHub repos for updates. You should have a CI/CD pipeline that builds things for you in a reproducible way and consider having mirrors of package repos or libraries. The primary benefit of doing this is it helps ensure you know which libraries are being used to build your services.

With more people working in an AWS environment, you can lose track of what resources are associated with which applications or teams. When a problem is found with a resource, you want to be able to quickly identify who the owner is, without looking through the CloudTrail logs. This becomes especially important when a CI/CD system is solely responsible for creating resources, as you become less able to identify the owner. Therefore, you should have a tagging strategy. This can be enforced via SCPs or scanning or monitoring can be used to detect when required tags are not used.

Stage 8: Enhance detection and least privilege refinement

- ☐ Implement real-time monitoring.
- ☐ Implement automated remediation.
- ☐ Refine IAM policies.
- ☐ Deploy honey tokens.

CloudTrail logs take 15 minutes to appear in S3. You can use CloudWatch Events to monitor events in near real-time. This requires setting up a CloudWatch Event Rule in every account, in every region you wish to monitor, and then aggregating these events. Note that CloudWatch Events do not record List, Describe, and Get calls so it is not a complete replacement for some rules you might have on the monitoring of the CloudTrail logs going to S3.

In cases where SCPs or other features cannot enforce policies, you can have automated remediations take action on your behalf. This can include removing unused IAM roles, changing security groups that are too open, terminating EC2s in a test environment every evening, unsharing resources that are shared with unknown accounts, and more.

You should also take the time to review and refine your IAM policies to better restrict them. Your policies should have specific Actions, Resources, and include Conditions. Stars ("*") should be avoided unless required by the Action. You should avoid using AWS provided Managed IAM policies because they are not restricted enough. To help reduce the scope of policies to only specific actions, you can use Client Side Monitoring (CSM) as described in Record AWS API calls to improve IAM Policies[42].

Place "fake" access keys as honey tokens for an attacker to find and use so you can more easily detect them. I describe how to do this in Guidance on deploying honey tokens[43].

Stage 9: Secure network communications

- ☐ Move all non-public network resources into private subnets and proxy outbound requests so you can filter and block them.

[42] https://cloudonaut.io/record-aws-api-calls-to-improve-iam-policies/
[43] https://summitroute.com/blog/2018/06/22/guidance_on_deploying_honey_tokens/

- ☐ Restrict egress network traffic.

Some companies create subnets for accounts as part of their account initialization process. These are peered to other accounts in some way and then SCPs are used to ensure no new subnets, Internet Gateways, or other networking components can be created. By doing this you can ensure that some accounts only have private subnets. You can even have isolated networks[44] that cannot make any outbound requests to the Internet.

If you have publicly facing load-balancers in front of your EC2s in some accounts, you can move the EC2s to private subnets. You can then restrict the network communication they initialize, and can run that outbound traffic through proxies in order to better monitor and restrict it.

As an additional network precaution, you can use AWS PrivateLink to restrict access to resources to only those coming from certain VPCs by setting their resource policies to only allow access from VPC endpoints. You may need to have a proxy to access third-party vendors, but you should be able to restrict egress network traffic so nothing calls out to the Internet, or does so only through a proxy. This is not only applicable to EC2s, but also Lambdas and containers. Services such as the AWS Network Firewall can assist with monitoring and protecting your networks.

Stage 10: Incident preparation

- ☐ Limit the blast radius of incidents.
- ☐ Practice responding to incidents.

Consider if it makes sense to break up accounts further. As opposed to direct access to resources, you may create a microservice where better monitoring, restrictions, and rate limiting could be performed. Consider breaking up services into pieces so that if one is compromised the blast radius is more limited.

Ensure there is a separation of duties so the security team can monitor the production accounts without being able to cause incidents there, and those with production access, cannot impact the security team.

If you are lucky enough to not have any security incidents, you should practice what happens when systems or IAM roles are compromised, either via a planned exercise, or red team engagement. How long does it take you to identify, understand, and react? Use the lessons learned to improve.

Conclusion

I work as an independent AWS security consultant. I provide training on AWS security. Reach out to me at scott@summitroute.com or read more at summitroute.com

[44]https://summitroute.com/blog/2020/03/31/isolated_networks_on_aws/