CLOUD THREAT REPORT 2H 2020

Never trust, always verify.

Table of Contents

Foreword

Historically, defense in depth has mostly been performed through network-layer controls. While network security controls remain an important component of cloud security, an additional layer of identity and access management (IAM) governance is now needed as organizations continue to scale their cloud presence. Similar to scanning applications for vulnerabilities, IAM policies across all cloud accounts must be constantly monitored and evaluated to determine the risk impact to the business.

Research findings from two large-scale projects, a Red Team exercise and a GitHub® reconnaissance operation, clarified this need. In the Red Team exercise alone, **one simple IAM misconfiguration allowed our Unit 42 researchers to compromise an entire, massively scaled cloud environment and bypass just about every security control**.

Given the severity of this finding, Unit 42 researchers quickly pivoted to measure just how pervasive IAM misconfigurations were across other cloud environments. They found several common identity-related flaws across thousands of cloud accounts, many of which could seemingly be attributed to a lack of IAM governance and standards.

Consider figure 1, where the user account has access across three different cloud providers, each with its own unique roles and permissions. The governance challenge here comes from the sheer volume of user and machine roles combined with permissions and services that are created in each cloud account.

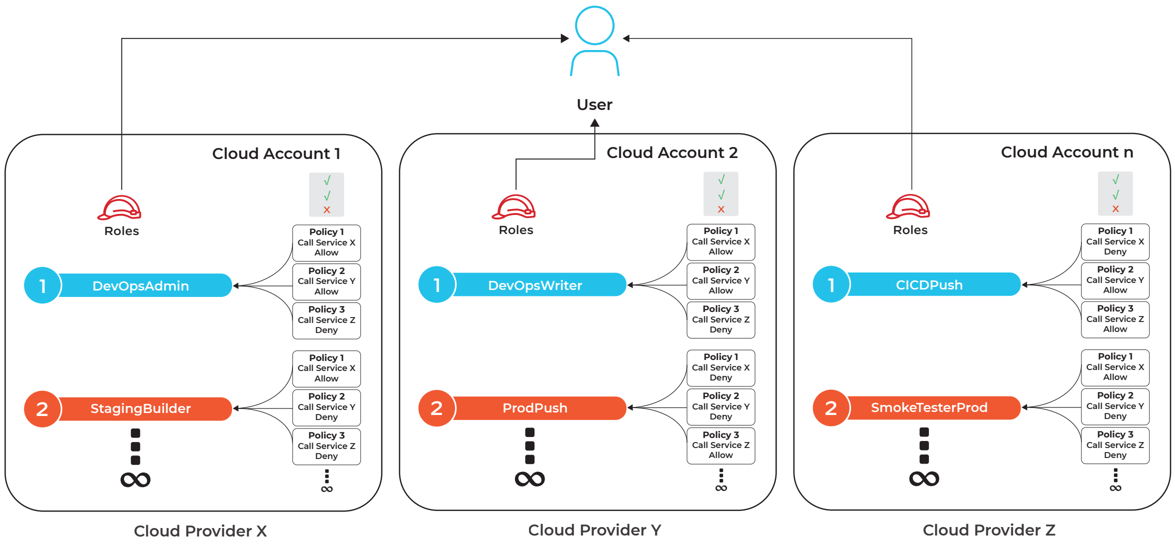


Figure 1: Cloud user account access diagram

Humans are good at many things, but understanding effective permissions and identifying risky policies across hundreds of roles and different cloud service providers are tasks best left to algorithms and automation. Read the report and find out how to take action!

Matthew Chiodi
CSO, Public Cloud at Palo Alto Networks

Executive Summary

Cloud is poised to become the dominant way that organizations store their data and manage applications. Our own data shows 46% of organizational workloads are already there, with the figure likely to grow to 64% in the next 24 months. To better understand the threat landscape associated with this rapid shift, Unit 42 researchers focused deeply on identity in the cloud. They analyzed methods that attackers use to silently perform reconnaissance operations, as well as common threat actors. Researchers also carefully identified steps organizations can take to build a cloud security program based upon identity best practices. The research took place between May and August 2020 and was global in scope—spanning terabytes of data, thousands of cloud accounts, and more than 100,000 GitHub code repositories. **Overall, the findings indicate that identity misconfigurations are prevalent across cloud accounts and represent a significant security risk to organizations, which can lead to costly data breaches**.

Cloud Identity Flaws Are Difficult to Detect

During a Red Team exercise, Unit 42 researchers were able to use a customer misconfiguration to compromise an entire Amazon Web Services (AWS®) environment, with thousands of workloads, in less than one week. They were able to do this by exploiting a single misconfigured IAM trust policy. With this flaw, an attacker could launch any number of attacks against an organization, including denial-of-service (DoS) and ransomware, or even open a door for an advanced persistent threat (APT) adversary. Because identity defects are difficult to detect, especially at scale, many go unnoticed by organizations until it's too late.

Identity Misconfigurations Lead to High-Impact Failures

In the same Red Team exercise, Unit 42 researchers identified an IAM role used by hundreds of users, which they were able to compromise. This allowed them to achieve administrative access outside of the development area. Once outside of development, the misconfigured IAM role allowed researchers to identify and hijack a legitimate administrator account and establish full administrative control over the entire cloud environment. With the "keys to the kingdom," attackers could launch any number of attacks against an organization, such as stealing sensitive data or wiping out the entire infrastructure.

JAPAC and EMEA Organizations Display Poor Cloud Identity Hygiene

Unit 42 researchers found that 75% of organizations in Japan and Asia-Pacific (JAPAC) as well as 74% of organizations in Europe, the Middle East, and Africa (EMEA) are using Google Cloud to run workloads with admin privileges. By contrast, only 54% of organizations in the Americas run with the same type of privileges. It is a best practice to run workloads with the principle of least privilege—limiting permissions for users to the bare minimum they need. If an attacker is able to compromise a workload with admin privileges, they would gain the same level of elevated access. This provides an easy path for attackers to use cloud resources to perform attacks, like cryptojacking operations, at the expense of the organization.

Cryptojacking Remains a Persistent Threat for Organizations

Unit 42 research shows cryptojacking affects at least 23% of organizations globally that maintain cloud infrastructure—a sharp rise from the 8% that researchers observed in February 2018. The mining pools connected to by cloud organizations are more often located in the United States, with 69% of all public mining traffic being directed to US systems. This is due to the majority of the Monero nodes being hosted on US systems. Monero is a popular cryptocurrency used in cryptojacking operations.

01

Identity and Access Management

Identity Flaws Worth Millions

In the spring of 2020, the Unit 42 cloud threat intelligence team was approached by a customer who wanted the group to test the defenses of the customer's AWS infrastructure. While offensive operations are not something Unit 42 typically carries out, they were intrigued by the size and complexity of this particular environment. The customer ran thousands of workloads, hundreds of Amazon Simple Storage Service (S3) buckets, and cloud native databases with more than 500 active development users and nearly 1,000 roles across four AWS accounts. Unit 42 researchers would classify this exercise as a "gray box penetration test" since they were provided limited information about the internal architecture and given limited access to the environment itself. **It took Unit 42 researchers less than a week to discover two critical IAM misconfigurations, which rippled through all the customer's AWS accounts, either of which could be crippling for any organization.**

Attackers are actively attempting to scan for credentials within cloud environments. This is most notably evident from the cryptojacking operations of `Kinsing` and TeamTnT, when, upon compromise of a cloud instance, malware will scan for any configured AWS credentials and attempt to send those credentials to a command and control (C2) node. Access to these credentials allows the attackers to expand the attackable surface and increase their chances of identifying a misconfigured IAM role. For more information regarding the operations of Kinsing and TeamTnT, see Cryptojacking Tool Spotlight.

The Red Team exercise involved two high-level techniques. The first, titled "Misconfigured IAM Trust Policy," details an "outside in" style of penetration. The attacker is unauthorized and has no access to internal credentials, but is able to gain access to internal resources. The second technique, titled "Lateral Movement and Privilege Escalation," details an "inside up" style of attack. Here, the attacker starts with limited access (non-administrative access) and is able to escalate privileges to gain administrative access to the entire cloud environment.

Unit 42 researchers also attempted other IAM penetration techniques, such as checking policies across S3 buckets, Amazon Elasticsearch Service and Amazon EBS snapshot. **All these techniques are based on real-world incidents**, highlighting how vulnerable IAM misconfigurations can be.

The first IAM misconfiguration allowed the Unit 42 team to access sensitive data within internal, non-public S3 buckets. These S3 buckets contained sensitive internal information, such as certificate keys, database credentials, and a source code repository. If an attacker had obtained access to these S3 buckets, they could have launched any number of attacks against an organization, including DoS and ransomware, or even open a door for an APT adversary. An APT adversary would likely gain a beachhead by implanting backdoor code into the source code, leading to a significant disruption in business operations and revenue.

The second misconfiguration was over-privileged IAM roles assigned to non-administrator user accounts. Unit 42 researchers identified non-administrator user accounts that were granted access to an administrative IAM policy. This allowed the researchers to escalate their privileges and gain administrative access to the entire cloud environment. With the "keys to the kingdom," attackers can launch any number of attacks against an organization, such as stealing sensitive data, wiping out the entire infrastructure, or locking down the operation with ransomware.

Unit 42 researchers note that AWS has tried its best to detect and alert users when an IAM trust policy is misconfigured. When a new IAM role is created in the console, its trusted principle is always limited to specific AWS services, accounts, or identity providers. However, while IAM trust policies are secure by default, users can still override the policies and introduce insecure configurations. Multiple warnings are alerted in the policy editor and policy viewer in these instances. AWS also offers their free IAM Access Analyzer to help identify unintended access to resources and data that are shared with an external entity.

Unit 42 researchers estimated the potential financial impact of a data breach for this customer could have easily been in the tens of millions, given the customer's scale and business model. **Fortunately, forensic work indicated that no malicious actors had successfully exploited these IAM misconfigurations, and Palo Alto Networks helped the customer remediate the issues.**

The following sections explain in detail how these two techniques were executed.

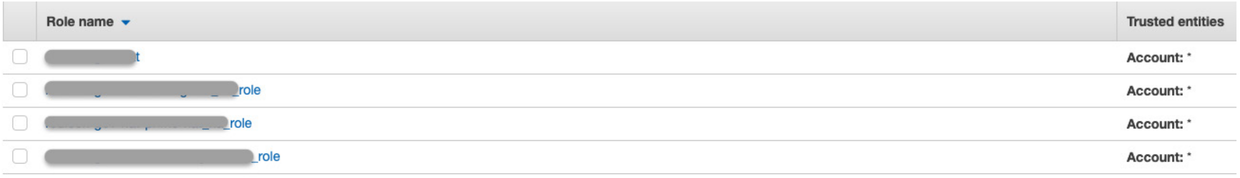
Misconfigured IAM Trust Policy

For the "outside in" approach, Unit 42 researchers were able to successfully leverage the AWS IAM feature "AssumeRole" to gain temporary internal access using an unauthenticated external AWS account. The root cause that allowed this unauthenticated access was an overly permissive IAM role trust policy. **The misconfigured policy allowed any AWS user who is not in the account to assume the role and obtain an access token.**

This issue was further amplified across multiple AWS accounts due to the use of infrastructure as code (IaC). The customer used Terraform® IaC to manage and provision services across their production, development, and government clouds. Although it is a best practice to use IaC to assure the quality of large-scale application deployments, a vulnerable configuration in an IaC template could lead to a catastrophic ripple effect—such as what would have likely happened to this customer given enough time. **The misconfigured trust policy in this customer's IaC template was replicated to multiple roles across multiple accounts**. Unit 42 researchers found more than 30 vulnerable entry points in the customer's cloud environment that could all have been exploited the same way. Put simply, the chance of an account with 30 misconfigured roles being compromised is 30 times higher than an account with only one misconfigured role.

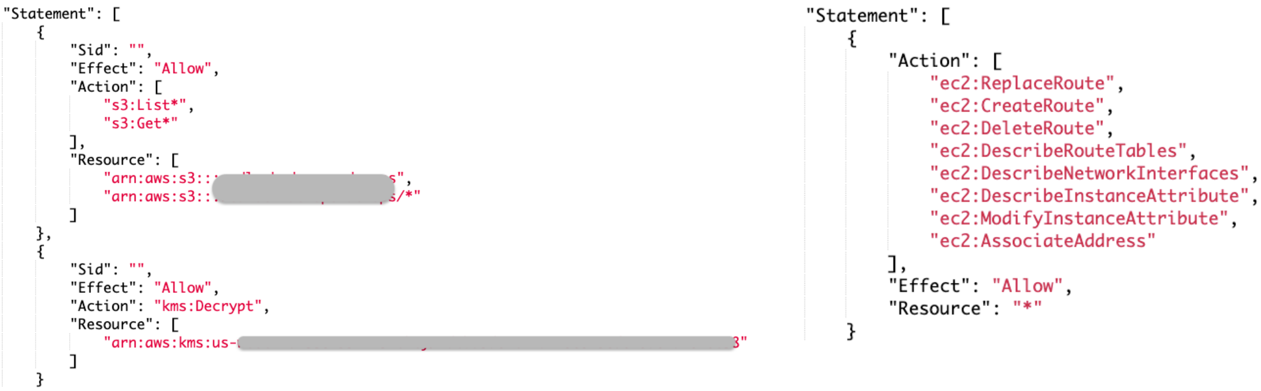
Path of Attack

Figure 2 shows an example of misconfigured IAM roles (obfuscated to protect our customer's identity, which could easily be inferred from the role names). The trusted entities of these roles are "**Account: ***," meaning that all AWS accounts are allowed to assume these roles. Figure 3 shows the permission policies attached to these roles. They all have access to Elastic Compute Cloud (Amazon EC2®), S3, and Key Management Service (KMS). Although each role has only a limited number of permissions, Unit 42 researchers were able to use these permissions to move laterally to adjoining S3 buckets attached to the accounts, and eventually gain access to sensitive data.



	Role name ▾	Trusted entities
□	[redacted]t	Account: *
□	[redacted]role	Account: *
□	[redacted]role	Account: *
□	[redacted]_role	Account: *

Figure 2: Overly permissive IAM roles trust policy



```
"Statement": [
    {
        "Sid": "",
        "Effect": "Allow",
        "Action": [
            "s3:List*",
            "s3:Get*"
        ],
        "Resource": [
            "arn:aws:s3:::[redacted]s",
            "arn:aws:s3:::[redacted]s/*"
        ]
    },
    {
        "Sid": "",
        "Effect": "Allow",
        "Action": "kms:Decrypt",
        "Resource": [
            "arn:aws:kms:us-[redacted]3"
        ]
    }
```

```
"Statement": [
    {
        "Action": [
            "ec2:ReplaceRoute",
            "ec2:CreateRoute",
            "ec2:DeleteRoute",
            "ec2:DescribeRouteTables",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeInstanceAttribute",
            "ec2:ModifyInstanceAttribute",
            "ec2:AssociateAddress"
        ],
        "Effect": "Allow",
        "Resource": "*"
    }
```

Figure 3: The misconfigured IAM role has limited access to S3, EC2, and KMS services

Figure 4 illustrates how the researchers discovered, exploited, moved laterally toward, and eventually gained access to credentials in the customer's AWS cloud environment. Unit 42 researchers identified and confirmed the step-by-step actions an attacker could take to compromise the environment:

1. **Obtain a list of role names through reconnaissance and enumeration** (e.g., `prodApp-nat`, `prodApp-app2-nat`). Because the role names are short and somewhat predictable, it is feasible to find a misconfigured role through enumeration.
2. **Procure a temporary access token by assuming the misconfigured role**. With the access token, the attacker can enumerate the permissions and find the resources the role can access.
3. **See all the EC2 instances and obtain the metadata attached to these instances**. From the startup script attached to each VM, the attacker can obtain information such as the Docker images the VM deploys, the database that the VM queries, and the S3 buckets the VM pulls data from.
4. **Access the S3 buckets found in the EC2 metadata and download all the data**. There are certificate keys, multiple shell scripts used for deploying applications, and a few encrypted files containing credentials.

5. **Use the AWS KMS decrypt capability available to the role to decrypt the ciphertext**, giving them plaintext access credentials.
6. **Acquire the plaintext credentials**. With these, the attacker can move laterally and access the Docker Hub repository, Splunk server, and databases.

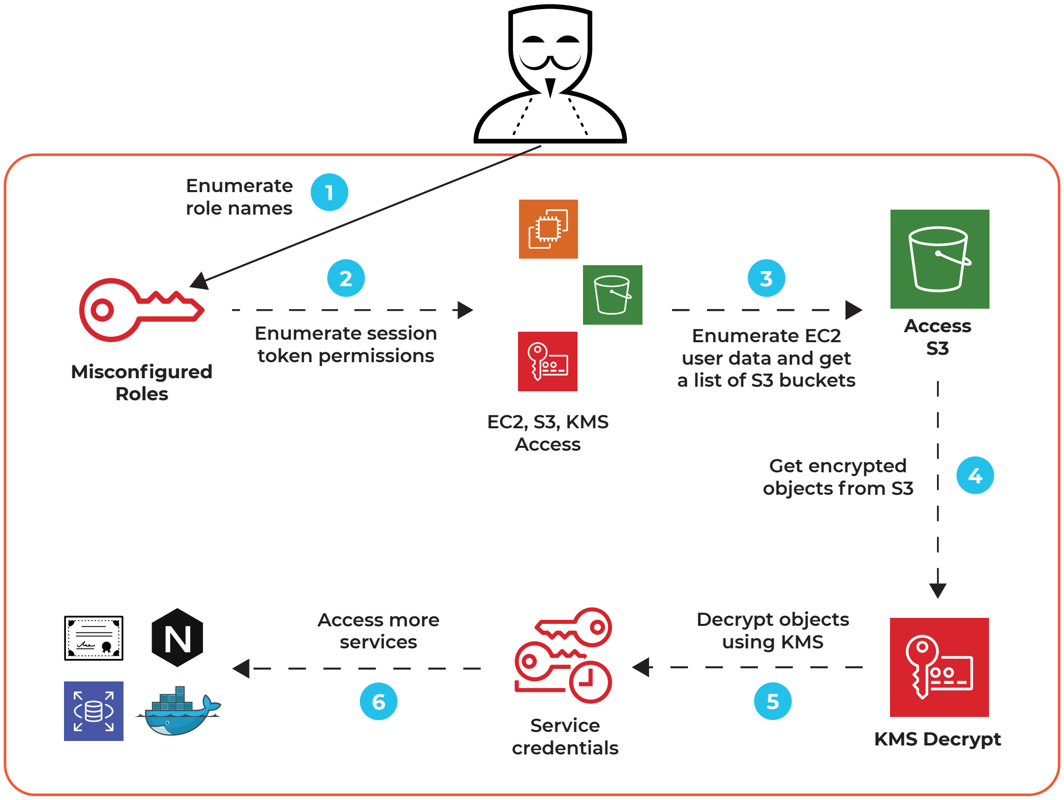

Enumerate role names
1
Misconfigured Roles
2
Enumerate session token permissions
EC2, S3, KMS Access
3
Enumerate EC2 user data and get a list of S3 buckets
Access S3
Get encrypted objects from S3
4
KMS Decrypt
Decrypt objects using KMS
5
Service credentials
Access more services
6
N

Figure 4: Attacker's use of the AssumeRole feature chain of events

If attackers had beaten the Unit 42 team in discovering this IAM misconfiguration, they could have launched any number of attacks against this customer. This specific situation would have proven fertile ground for an APT adversary. This type of attacker looks for misconfigurations like this to gain a foothold, and likely would have implanted backdoor code into the source code, spelling disaster for both the customer and the customer's clients.

Lateral Movement and Privilege Escalation

For the "inside up" approach, Unit 42 researchers were able to successfully move laterally with non-administrator access by leveraging a misconfigured IAM role related to flow log management. They escalated their privileges from a restricted developer account to gain persistence within the customer's AWS cloud environment by hijacking an administrator account. **By leveraging an overly permissive IAM role related to flow logs, Unit 42 researchers gained administrative access to the entire cloud environment, allowing for the unrestricted manipulation of cloud resources.** They were then able to create new Amazon EC2 and Amazon Relational Database Service (RDS) instances as well as modify user and policy permissions.

Path of Attack

Unit 42 researchers identified developer accounts that could be leveraged to expand the attackable surface area of the cloud environment. Traditionally, developer accounts are allowed a higher level of access to cloud environments than standard users. While this was also true within this cloud environment, the developer accounts were subject to security constraints. They were not allowed to perform any administrative actions outside of the given development environment (see figure 5).

For example, users of these developer accounts would not have administrative access to IAM resources, such as user accounts or policies. The developer accounts were limited to the following set of AWS resources and were only able to manipulate these resources within a predefined development environment:

- Database administration
- System administrator
- Network administrator
- Key management service
- Amazon Elastic Map

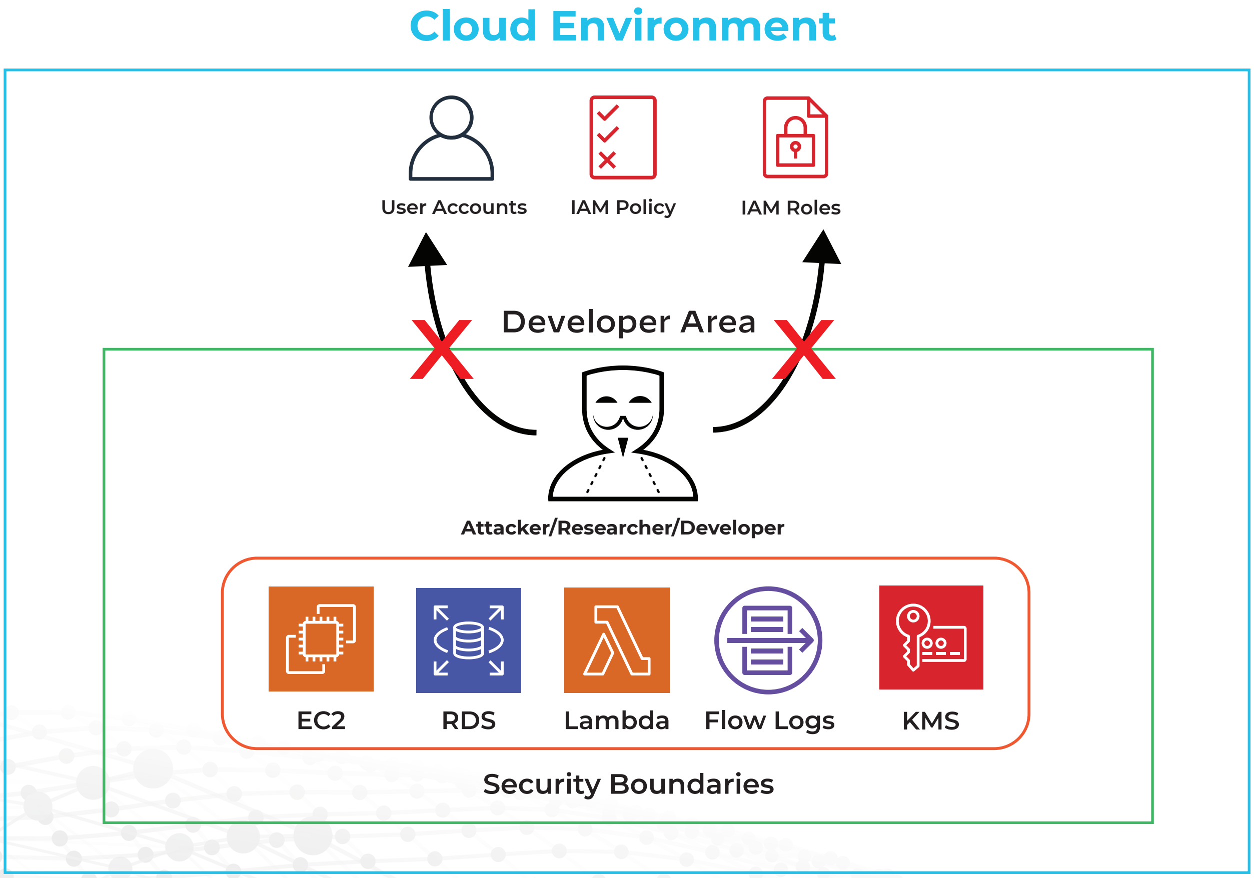


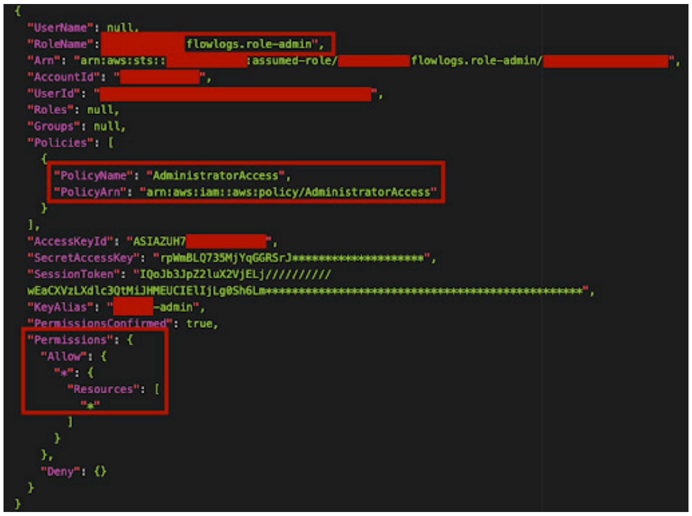
Figure 5: Developer access limitations

Upon enumerating the permissions of the developer account, Unit 42 researchers identified a glaring security hole: developer accounts had access to a custom organization-specific IAM role called `flowlog*`. The asterisk signified that all flow log subtypes were available to the developer account, which included a role titled `flowlogs.role-admin`. This flaw would allow an attacker to escape the developer area and achieve administrative access to the entire cloud environment. The attacker would then be able to manipulate any resource within the customer's cloud environment—all courtesy of an overly permissive admin IAM role.

Since the developer account contained access to the IAM `role flowlogs.role-admin` and also had the ability to create new EC2 instances, Unit 42 researchers created a new EC2 instance and assigned to that instance the administrator flow log IAM role. Upon the successful creation of the new EC2 instance, enumeration of the system's permissions was performed using the AWS command line interface (CLI) command `describe-resource-permissions` (see figure 6). The image was created using the output of a 3rd party tool, Pacu, which concatenates several commands into a single output. Using the AWS CLI alone will not produce the same output.

In the screenshot in figure 6, `AdministratorAccess` was granted to the EC2 instance, and the instance maintained open permissions to all resources within the cloud environment.

A note on asterisks: Any time an asterisk exists within an IAM role or policy, or even within an IaC template, the asterisk designates a wildcard, or rather a "select-all" functionality, for any entity that matches the given criteria. Asterisks should be considered red flags for security professionals, and efforts should be made to enumerate all possible subtypes presented within an asterisk to ensure overly permissive resources are not present. "When in doubt, spell it out" should become a mantra for IT admins, development engineers, and security professionals alike.



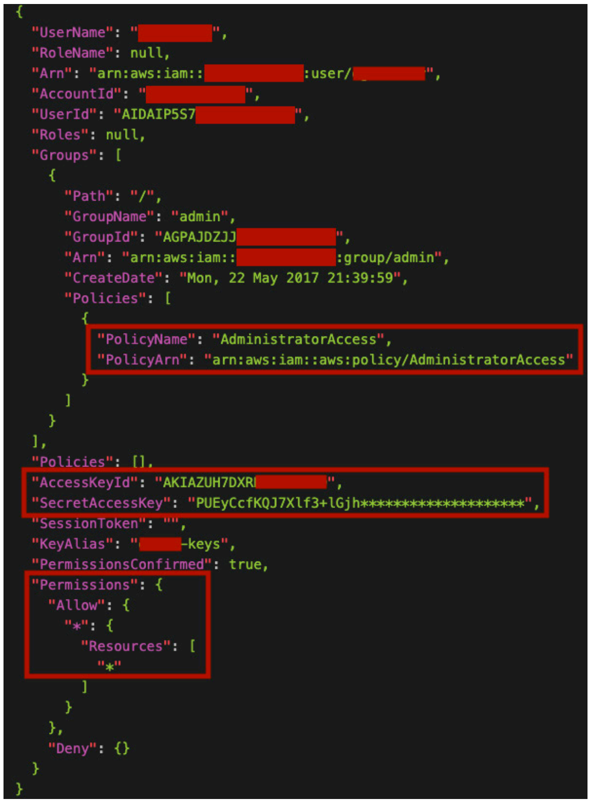
```
{
  "UserName": null,
  "RoleName": [redacted]flowlogs.role-admin",
  "Arn": "arn:aws:sts::[redacted]:assumed-role/[redacted]flowlogs.role-admin/[redacted]",
  "AccountId": "[redacted]",
  "UserId": "[redacted]",
  "Roles": null,
  "Groups": null,
  "Policies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ],
  "AccessKeyId": "ASIAZUH7[redacted]",
  "SecretAccessKey": "rpWmBLQ735MjYqGGRSrJ********************",
  "SessionToken": "IQoJb3JpZ2luX2VjELj//////////
  wEaCXVzLXdlc3QtMiJHMEUCIElIjLg0Sh6Lm************************************************",
  "KeyAlias": "[redacted]-admin",
  "PermissionsConfirmed": true,
  "Permissions": {
    "Allow": {
      "*": {
        "Resources": [
          "*"
        ]
      }
    },
    "Deny": {}
  }
}
```

Figure 6: Permissions for `flowlogs.role-admin`

Having gained full administrative rights to the cloud environment via the creation of a new EC2 instance with administrative rights, Unit 42 researchers attempted to gain persistence within the environment. They hijacked an administrator account and used that legitimate administrator account to create new users and resources. By using the `list-users` command from within the newly created EC2 instance, Unit 42 researchers were able to collect user access IDs. By using the commands `get-user`, `list-permissions`, `list-groups-for-user`, and `list-user-policies`, they gathered additional insight into the rights granted to the hijacked administrator account (see figure 7).

Figure 7 shows that Unit 42 researchers were no longer restricted to the developer area and could create as well as modify any resource within the entire cloud environment (see figure 8).

An attacker with this access could have followed any number of playbooks against the customer, including data exfiltration, disruption of business operations, and/or locking down the environment with ransomware.



```
{
  "UserName": "[redacted]",
  "RoleName": null,
  "Arn": "arn:aws:iam::[redacted]:user/[redacted]",
  "AccountId": "[redacted]",
  "UserId": "AIDAIP5S7[redacted]",
  "Roles": null,
  "Groups": [
    {
      "Path": "/",
      "GroupName": "admin",
      "GroupId": "AGPAJDZJJ[redacted]",
      "Arn": "arn:aws:iam::[redacted]:group/admin",
      "CreateDate": "Mon, 22 May 2017 21:39:59",
      "Policies": [
        {
          "PolicyName": "AdministratorAccess",
          "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
      ]
    }
  ],
  "Policies": [],
  "AccessKeyId": "AKIAZUH7DXR[redacted]",
  "SecretAccessKey": "PUEyCcfKQJ7Xlf3+lGjh********************",
  "SessionToken": "",
  "KeyAlias": "[redacted]-keys",
  "PermissionsConfirmed": true,
  "Permissions": {
    "Allow": {
      "*": {
        "Resources": [
          "*"
        ]
      }
    },
    "Deny": {}
  }
}
```

Figure 7: Hijacked administrator user permissions

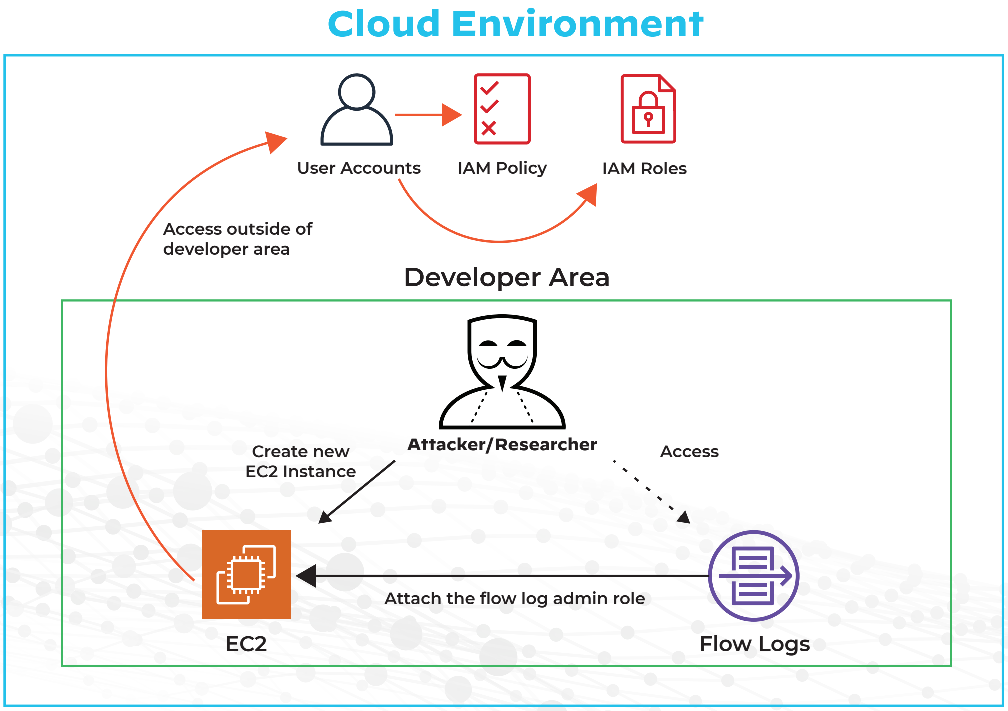


Figure 8: Attacker’s access to administrative rights

Real-World Examples

Since this type of misconfiguration can be observed and exploited in the wild, Unit 42 researchers were curious to discover how common these types of IAM role misconfigurations were. The researchers' approach was to use publicly available data to conduct the reconnaissance operations.

Two critical pieces of information for assuming an IAM role are the AWS Account ID and AWS IAM Role Name. An AWS account ID is a 12-digit number that uniquely identifies an account. AWS account ID is included in every Amazon Resource Name (ARN) used to uniquely identify AWS resources.

For example, `arn:aws:iam::123456789012:user/Development/Alice` identifies user Alice in AWS account 123456789012. `arn:aws:lambda:us-west-2:987654321098:function:demo-function` identifies the Lambda function `demo-function` in AWS account 987654321098.

Neither AWS account IDs or IAM role names should be treated as secrets. They are commonly shared and used by multiple entities. However, it is a best practice to use unguessable role names as they can help mitigate brute-force attacks.

Most IAM roles (other than Service-Linked Roles) are created and maintained by the account owner, not AWS. The naming syntax is similar to username, and the role names are usually created based on the functionalities of the role. For example, the role `lambda_basic_execution` is expected to have Lambda invocation permission, and the role `ecsInstanceRole` is expected to be attached to an ECS instance.

Misconfigurations Are Relatively Easy to Find

Searching for misconfigured IAM roles is similar to searching for exposed databases that allow anonymous login. Instead of scanning for IP addresses, however, Unit 42 researchers performed a scan for AWS account IDs. For each valid AWS ID identified, its role names were checked against a pre-generated list created by crawling GitHub and aggregating the top 500 most-used IAM role names.

If any of the role names existed in the account and were misconfigured, Unit 42 researchers (or attackers) could presumably assume this role and obtain an access token by using the methods described previously. This step is like guessing default usernames and passwords in a database. Although it is possible to enumerate all 12 digits of an AWS account ID from 000000000000 to 999999999999, the enumeration is time-consuming and can be blocked by AWS.

Instead, Unit 42 researchers crawled GitHub for *potential* AWS ID and role name matches. Due to the popularity of IaC templates such as AWS CloudFormation and Terraform, it was straightforward to analyze the components in cloud infrastructure by parsing IaC text files.

The amount of data in GitHub was sufficient for a proof of concept. Overall, Unit 42 researchers analyzed 283,751 files across 145,623 repositories, identifying 32,987 confirmed AWS account IDs and 68,361 role names.

Figure 9 shows the high-level research methodology:

- **GitHub search**: Use GitHub API to search keywords used in ARNs. The AWS IAM document lists a set of possible ARNs.
- **File analysis**: Use static analysis to parse out potential AWS account IDs and IAM role names.
- **Validation**: Check if the parsed AWS account IDs exist by sending HTTP requests to the console login page.
- **Role name enumeration**: Enumerate validated AWS account IDs with the parsed role names from GitHub.
- **Configuration check**: Check the configuration of each (AccountID, role name) pair by assuming this role. The role can be successfully assumed if its trust policy allows anyone to use it.

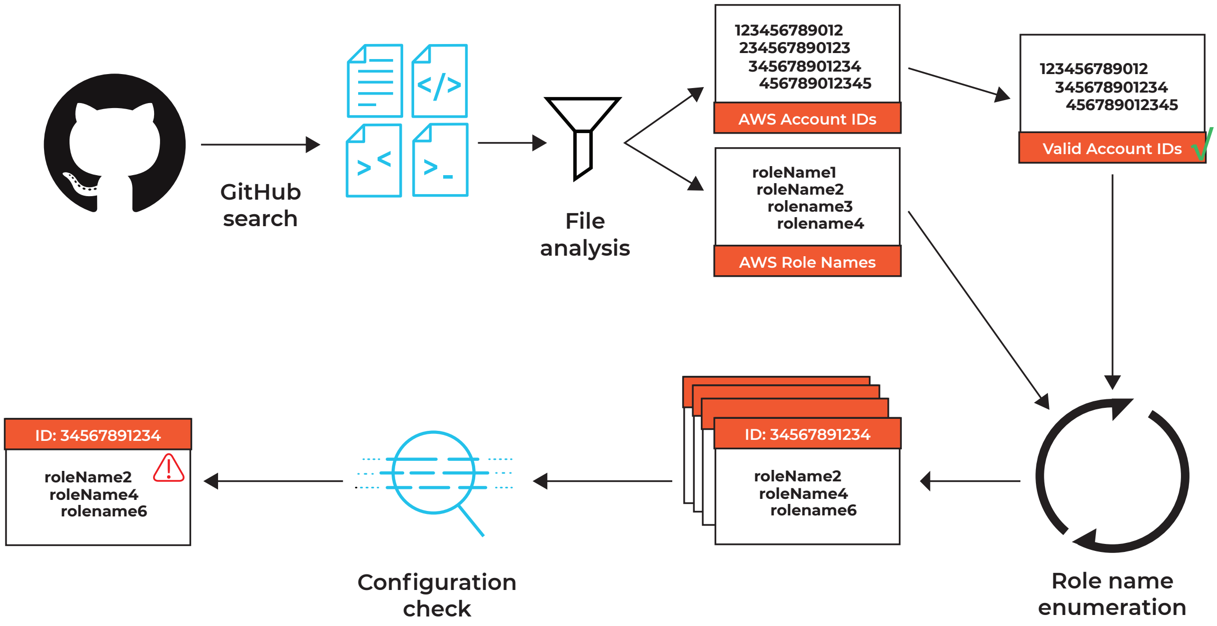


Figure 9: Research methodology illustration

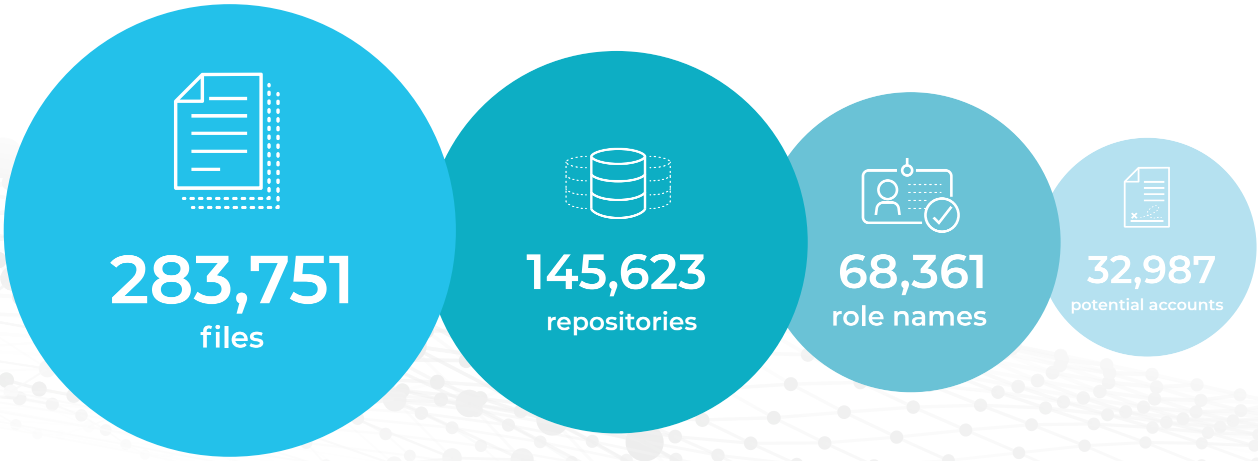


Figure 10: GitHub search result totals

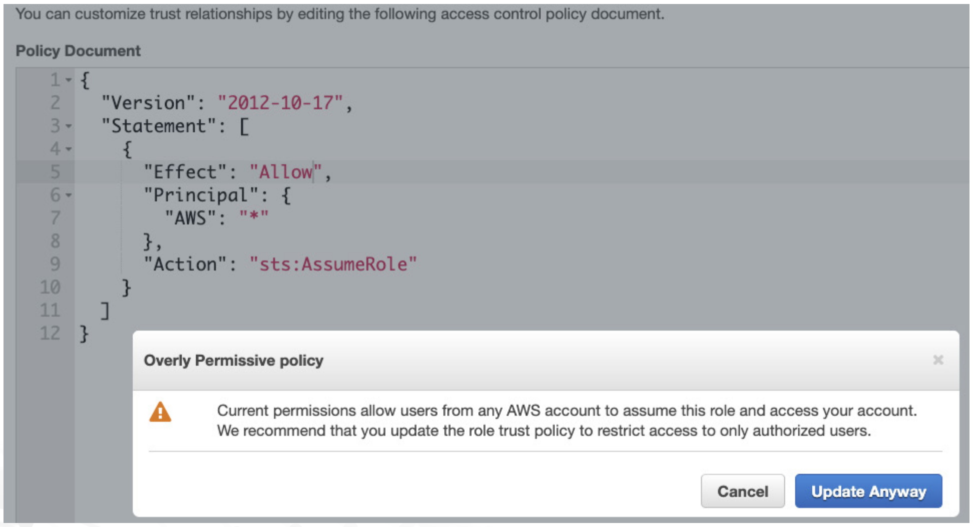
Thousands of Public-Facing Misconfigurations

The data from GitHub, when combined with other sources, revealed more than 175,000 EC2 snapshots, hundreds of S3 buckets, and many RDS snapshots and KMS keys. Make no mistake: if attackers had discovered what Unit 42 researchers found, they would have assuredly taken steps to “own” these cloud accounts. The resources leaked from a misconfigured IAM role depended on the permission policy of the role itself. Researchers discovered misconfigured DevOps roles that had near-system admin permissions. Also found were misconfigured DBAccess roles that had access to database services such as Amazon DynamoDB® and Amazon Redshift®. Finally, there were LambdaExecution roles that allowed only basic `Get` and `Invoke` actions on Lambda functions.

Regardless of the types of resources these misconfigured roles could access, they all leaked information that malicious actors can exploit. **A compromised cloud account could be much worse than a compromised cloud host because a cloud account may have access to hundreds or thousands of cloud resources**. There have been instances where attackers ran cryptojacking software on compromised cloud hosts and walked away with AWS credentials. The seemingly endless resources in the cloud make the infrastructure an attractive target. Even an account with only Lambda execution permission could pose significant financial impact by invoking a large number of function calls.

As mentioned earlier, AWS has tried its best to detect and alert users when an IAM trust policy is misconfigured. When a new IAM role is created in the console, its trusted principle is always limited to specific AWS services, accounts, or identity providers.

Although IAM trust policies are secure by default, users can still override the policies and introduce insecure configurations. If a user edits the trust policy in the AWS console, multiple warnings are alerted in the policy editor and policy viewer, as shown in figure 11.



You can customize trust relationships by editing the following access control policy document.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Current permissions allow users from any AWS account to assume this role and access your account. We recommend that you update the role trust policy to restrict access to only authorized users.

Figure 11: Overly permissive policy warning in the AWS console

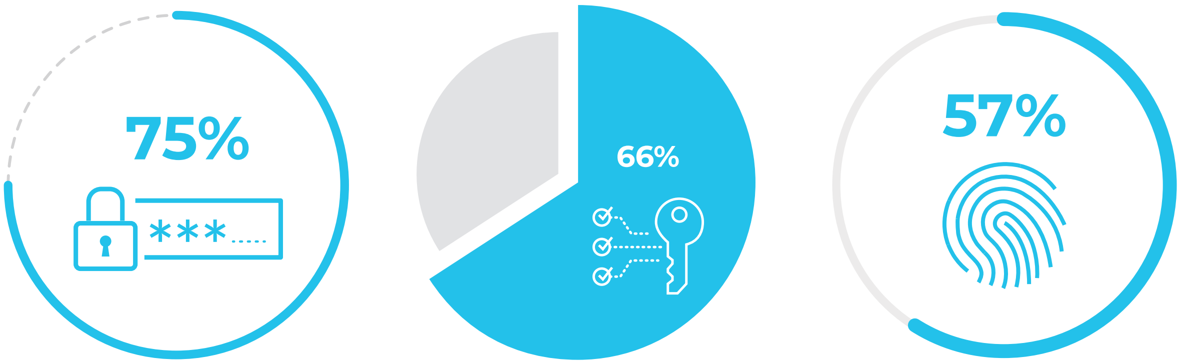
Unfortunately, insightful events and alerts such as these are too often ignored. The following section explores additional examples.

02

Evidence-Based Findings

Industry Trends

As part of each Cloud Threat Report, Unit 42 researchers present updates on trends they have been tracking, looking for clear indications of the overall security posture of cloud infrastructure around the world. Substantiating the real-world effect of IAM misconfigurations became possible by investigating the alerts and events that originate from cloud accounts. Unit 42 researchers pinpointed specific areas where organizations may need to spend more time securing and verifying their configurations prior to deployment.



75% of JAPAC organizations over-prescribe permissions to Google Cloud hosts

Using administrative access accounts within VM instances can allow attackers to easily compromise wider cloud resources. See Lateral Movement and Privilege Escalation for an example of this compromise.

66% of organizations worldwide are using access keys for longer than **90 days**

As with traditional passwords, access key maintenance is a critical component of cloud security hygiene. Access keys should be rotated regularly to prevent compromise.

57% of JAPAC organizations do not have multi-factor authentication (MFA) enabled for AWS IAM user

Additionally, 27% of JAPAC organizations do not have MFA enabled for AWS Root accounts. MFA remains one of the surest ways to add immediate defense in depth to cloud infrastructure. While there are many forms of MFA, some stronger than others (phone-based SMS being on the weaker side; hardware tokens being the strongest), what's most important is that some type of MFA be implemented for users with elevated (root) privileges.

Figure 12: Trend of policy violations

IAM Trends by Cloud Service Provider

When it comes to IAM risks, each CSP addresses them differently and, more often than not, in a siloed fashion. In examining IAM risks, Unit 42 researchers normalized findings to make a fairer, apples-to-apples comparison between each of the CSPs as well as provide a regional component to the research.

Analysis revealed a common structure in the IAM findings. Researchers found the most common risks within each of the three regions—the Americas, EMEA, and JAPAC—closely mirrored each other when compared across the usage of the three primary CSPs (AWS, Microsoft Azure®, and Google Cloud) within those regions.

Unit 42 researchers found that 75% of JAPAC organizations and 74% of EMEA organizations were using Google Cloud VM instances with administrative rights. This is in contrast to only 58% for organizations in the Americas region. This may not be surprising, given research by IDC in 2018 as part of the MaturityScape Benchmark: Cloud in Asia/Pacific (excluding Japan). IDC found that more than 85% of APAC organizations are still in the nascent stages of cloud maturity. In O'Reilly's Evolving Data Infrastructure report in 2018, the survey found that only 24% of European organizations felt they were "sophisticated cloud users." The lack of skills and talent among the workforce was pointed to as the leading contributor for this sentiment. Organizations that fall into these buckets can find themselves needlessly exposing their cloud resources to attacks, such as cryptojacking, ransomware, and intellectual property theft.

Table 1: Google Cloud VM Instances with Admin Rights by Region

User Account Permissions	Americas	EMEA	JAPAC	Global
Google Cloud VM instances with administrative rights	58%	74%	75%	62%

Unit 42 researchers also found that JAPAC organizations (60%) were more likely than Americas organizations (49%) to allow non-corporate,[1] third-party accounts, such as managed security service provider (MSSP) accounts or personal accounts, to access their Google Cloud environments. It appears that APAC (excluding Japan) is well on its way to meeting IDC's 2024 prediction that 60% of APAC organizations will rely on third-party organizations to assist with containers, open source, and cloud native application development.

JAPAC organizations (82%) are also more likely than EMEA organizations (74%) and Americas organizations (72%) to have AWS user accounts inactive for more than 30 days. At this time, however, it appears that JAPAC—as well as EMEA and the Americas, to some degree—have yet to implement IAM governance best practices when it comes to managing user access to cloud environments.

Table 2: Account Violations by Region

User and Key Access Violations	Americas	EMEA	JAPAC	Global
Non-corporate accounts have access to Google Cloud resources	49%	51%	60%	51%
AWS inactive users for more than 30 days	72%	74%	82%	73%
AWS access keys are not rotated for 90 days	67%	72%	67%	68%
Google Cloud user managed service account keys are not rotated for 90 days	64%	62%	60%	62%

[1] Non-corporate accounts are those that do not match the corporate domain naming convention (e.g., companyname.com). Examples include any supporting MSSP accounts, where MSSP employees use their own company accounts to log in to a customer cloud environment or personal account, such as gmail.com or outlook.com.

Another security component of IAM lies with the usage and maintenance of access keys. Access keys are used to provide a robust form of authentication when accessing cloud resources. Access keys are made up of two components, a 16 to 128 character Access Key ID, like AKIAIOSFODNN7EXAMPLE, and a Secret Access Key, like wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY. In order to access an account, the Access Key ID and the Secret Access Key need to be used together to authenticate. In the same manner as a username and password, the Access Key ID can be known to several people but the Secret Access Key is a private key that should only be known to the account owner.

However, access keys, just like traditional passwords, are only useful if managed correctly. More trust can be placed in access keys—and more importantly, in the integrity of the data they protect—if they are maintained and changed regularly. Private keys can be leaked, stolen, or acquired by various means. Changing or rotating access keys is a critical part of maintaining cloud security.

Worldwide, Unit 42 research found that 68% of organizations using AWS and 62% of organizations using Google Cloud have service account keys older than 90 days.

In an analysis of MFA practices by cloud organizations, Unit 42 researchers found that a staggering 57% of JAPAC organizations did not enable MFA within their AWS environments for IAM user accounts. This is in contrast to 46% and 45% of organizations within the Americas and EMEA, respectively. The number of organizations without MFA enabled for their IAM root accounts was also surprisingly high. JAPAC organizations again performed poorly, with 27% of IAM root accounts not using MFA functionality. Organizations in EMEA and the Americas were not far behind, with a respective 25% and 24% not enabling MFA functionality for IAM root accounts.

MFA functionality greatly increases the security of IAM accounts by requiring users to provide two forms of authentication to prove their identity. The failure to properly secure IAM root accounts can have devastating consequences. If an attacker were to gain access to an IAM root account, they would gain admin rights to all cloud resources available to that account. Attacks such as malware installation, persistence operations, and data exfiltration would be trivial.

Of note, analysis of Azure environments **did not** reveal a detectable measurement of user account policy violations, such as inactive user access or use of expired keys. This is likely due to Azure's usage of Azure Active Directory® (AD) to control and manage user accounts. Prisma Cloud added general availability for Azure AD support in August 2020. Unit 42 researchers will continue to monitor Azure IAM findings as customers begin to make use of this new functionality within the Prisma Cloud feature set.

Table 3: AWS MFA Enablement by Region

MFA Features	Americas	EMEA	JAPAC	Global
AWS MFA not enabled for IAM users	46%	45%	58%	47%
AWS MFA is not enabled on root account	24%	25%	27%	24%

03

Cloud Infrastructure Threats

Cloud environments are being targeted by cybercrime groups focused on malicious cryptomining operations, also called cryptojacking. These operations remain among the highest profile attacks against cloud infrastructure. **Unit 42 research shows that cryptojacking affects at least 23% of organizations globally that maintain cloud infrastructure**.

HACKER STORY

Unit 42 researchers are tracking a cryptojacking variant that is actively targeting and scraping AWS credentials in parallel to its mining operations. The tool, called TeamTnT, searches the compromised system for AWS credentials `~/.aws/credentials` and AWS configurations `~/.aws/config`. If either or both of these files are found, they are uploaded to a C2 server. This type of functionality exposes attackers' evolving efforts to further gain access to cloud environments.

Attackers typically use public Monero (XMR) mining pools, or in some instances a private mining pool, to perform their mining operations. The public mining operations can be tracked via Prisma™ Cloud, and Unit 42 researchers continue to update Palo Alto Networks URL Filtering, a Next-Generation Firewall subscription, to block connections to these networks as they appear.

Unit 42 researchers have found that attackers typically use misconfigured cloud infrastructure to initially compromise cloud instances. These misconfigurations lead to the bypass of firewalls, AWS Security Groups, Amazon and Google Virtual Private Clouds (VPCs), or Azure Virtual Network (VNet) policies, and can expose an organization's cloud environment. Figure 13 illustrates how an attacker scans cloud environments for exposed Docker daemon APIs. Upon detection, the attacker installs mining software on the exposed Docker daemon instance.

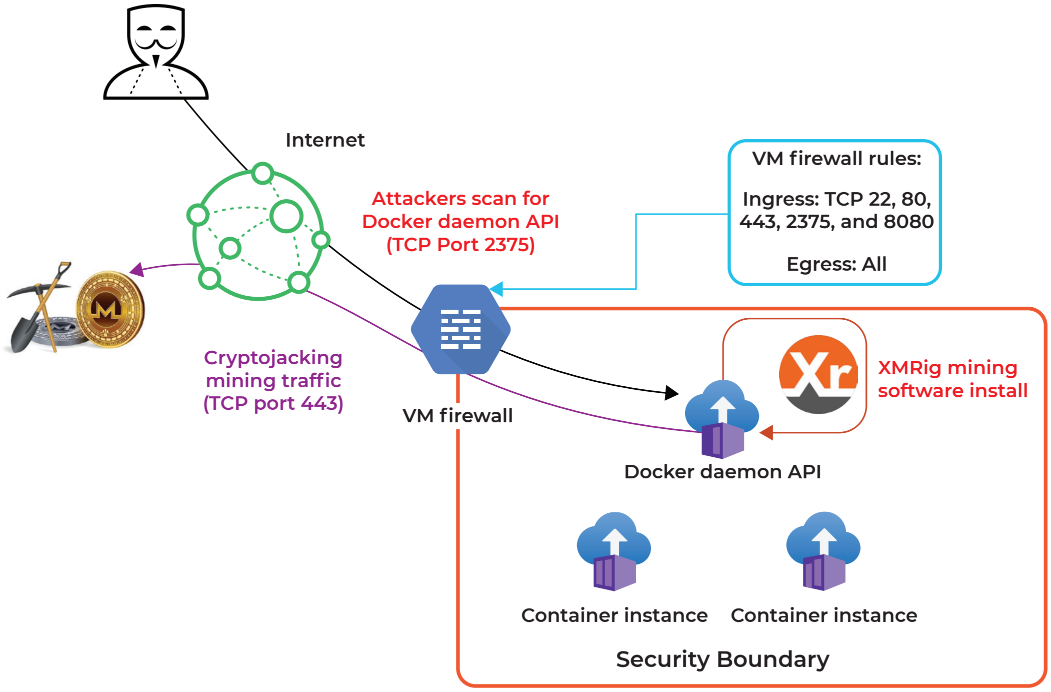

Internet
Attackers scan for Docker daemon API (TCP Port 2375)
VM firewall rules:
Ingress: TCP 22, 80, 443, 2375, and 8080
Egress: All
Cryptojacking mining traffic (TCP port 443)
VM firewall
XMRig mining software install
Docker daemon API
Container instance
Container instance
Security Boundary

Figure 13: Attacker exploiting an exposed instance via permissive firewall rules

By using cloud network security boundaries in the same manner as legitimate users, attackers can hide in the noise of the network. In figure 13, all mining network traffic is configured to use TCP port 443, thus obscuring attackers' tracks within legitimate network traffic noise. If an attacker were to use unconventional communication channels, they would be easier to identify and have a higher risk of detection.

Cryptojacking Operations

Analyzing a dataset from June through August 2020, Unit 42 researchers identified cloud organizations with persistent network connections to public cryptomining pools. During this 90-day period, the researchers looked for network connections originating from cloud organizations with destinations to any of the 46 most popular XMR mining pools. They found that cloud organizations communicated with 185 unique addresses.

Each of the following URLs should be added to organizational blocklists, including cloud environments. Table 4 lists the 46 most popular Monero public mining pools used in this research.

Table 4: URLs Organizations Should Add to Block Lists

alimabi[.]cn	minexmr[.]com	moneropool[.]com	stratum.f2pool[.]com	xmr.prohash[.]net
bohemianpool[.]com	monero.crypto-pool[.]fr	moneropool[.]nl	supportXMR[.]com	xmr.suprnova[.]cc
cryptmonero[.]com	monero.hashvault[.]pro	moriaxmr[.]com	teracycle[.]net	xmrpool[.]de
dwarfpool[.]com	monero.lindon-pool[.]win	myo-xmr[.]com	usxmrpool[.]com	xmrpool[.]eu
ethereumwallet[.]com[.]au	monero.miners[.]pro	pool.t00ls[.]ru	viaxmr[.]com	xmrpool[.]net
iwanttoearn[.]money	monero.riefly[.]id	pool.xmr[.]pt	xmr.2miners[.]com	xmrpool[.]xyz
mine.moneropool[.]com	monero[.]us[.]to	pooldd[.]com	xmr.crypto-pool[.]fr	–
minemonero[.]gq	monero[.]hashvault[.]pro	poolto[.]be	xmr.mypool[.]online	–
minercircle[.]com	monerohash[.]com	ratchetmining[.]com	xmr.nanopool[.]org	–
minergate[.]com	moneroocean[.]stream	sheepman[.]mine[.]bz	xmr[.]org[.]uk	–

Most Common Public Mining Pool

Of the most common destination connections, Unit 42 researchers found the most active mining pool was `xmr.nanopool[.]org`. This domain accounted for 20% of the total connections. `minercircle[.]com` had the second most, at 12% of connections, and `cryptmonero[.]com` and `usxmrpool[.]com` tied for the third most, at 11% of connections. `xmrpool[.]xyz` came in fifth with 9% of connections. Table 5 shows the top 10 public mining pools based on network traffic.

Table 5: Top 10 Public Mining Pools

Mining Pool	% of Connections	Count	Total Packets	Total Bytes
xmr.nanopool[.]org	20%	99,468	41,932,290	13.6 GB
minercircle[.]com	12%	58,922	6,904,587	881 MB
cryptmonero[.]com	11%	56,253	1,668,445	216 MB
usxmrpool[.]com	11%	55,217	7,336,290	881 MB
xmrpool[.]xyz	9%	42,231	16,197,889	4.05 GB
teracycle[.]net	6%	29,563	7,969,380	1.5 GB
monero.hashvault[.]pro	6%	29,417	21,565,838	18.3 GB
iwanttoearn[.]money	5%	24,583	29,225,318	9.9 GB
moriaxmr[.]com	3%	14,584	2,696,015	1.06 GB
moneroocean[.]stream	1%	6,169	816,334	626 MB

Upon analyzing the network traffic, Unit 42 researchers found that while `xmr.nanopool[.]org` contained the highest number of total connections, it came in second when looking at the total volume of traffic. `xmr.nanopool[.]org` generated 13.6 GB of traffic, where `monero.hashvault[.]pro` generated 18.3 GB of traffic with only 6% of the total connections. Higher total byte volumes indicate mining services are spending more time mining and not reconnecting or refreshing their network sessions. Unit 42 researchers believe that the connections with higher traffic volumes indicate a more established and resilient mining operation by the attackers within the cloud environment.

Remediating Cryptojacking

Significant mitigation efforts must be made upon identifying public mining operations. Simply removing the mining software and redeploying the cloud resources is not enough. A detailed investigation into the compromised cloud resource must begin. Unit 42 researchers recommend identifying any and all misconfigurations, vulnerabilities, or initial compromise events that led to a cloud resource compromise. These findings must be mitigated prior to redeployment of the cloud infrastructure; otherwise, reinfection is highly likely.

Most Common Port Usage

Unit 42 researchers also looked at the overall network connection traffic destined for public mining infrastructure. Ports 443 and 80, the most common ports used to connect to mining pool infrastructure, saw the highest total network connections and the highest total traffic volume. Port 443 led with 52% of the total connections and the largest total traffic volume at 48.5 GB. Port 80 came in second, accounting for nearly 37% of the total network connections and 8.7 GB of transferred network traffic.

HACKER STORY

The Unit 42 research team dove into a wormable cryptojacking variant it calls Cetus. Cetus, which has connections to the worm TeamTnT, targets and scans for exposed Docker daemons. Cetus is not the first cryptojacking worm, but what makes it powerful is its impersonation of legitimate Docker processes. Using the legitimate name “Portainer” (a Docker UI tool for managing Docker containers), the worm downloads the scanning tool “masscan” and an XMRig variant called “docker-cache”—also a legitimate name for a Docker binary. After installing the mining software, Cetus configures the software to use port 443 to communicate with the public mining pool. This operation displays the traits of process and traffic obfuscation as well as network reconnaissance along with cryptojacking operations.

Other ports of interest were ports 8080 and 9997. While port 8080 only maintained 4% of the total connections, it transferred 152 MB of network traffic, and port 9997 had a tiny 0.3% of the total traffic but managed to transfer 8.5 MB of network traffic.

This research shows the vast majority of cryptojacking operations use standard network protocols to perform their functionality (i.e., ports 443, 80, and 8080). It is also interesting to note that port 3333, the default port configuration for the most common Monero mining software XMRig, only contained a total of 77 network connections, or 0.01% of the total traffic. This shows that the majority of cryptojacking operations are using custom configurations for their mining operations. This can further indicate that attackers may have at least some knowledge of the cloud environment's configuration, or even its architecture, allowing them to disguise and maintain operations.

Table 6: Ports Most Commonly Used for Mining

Destination Port	Count	Total Packets	Total Bytes	% of Total
443	256,723	109,928,643	48.58 GB	52%
80	180,206	32,573,027	8.72 GB	37%
25	20,289	189,312	11 MB	4%
8080	15,089	1,111,362	152 MB	3%
137	11,610	200,173	15 MB	2%
123	2,516	25,635	835 KB	0.51%
389	1,619	18,946	708 KB	0.33%
9997	1,583	1,422,655	85 MB	0.32%
53	770	56,868	2.3 MB	0.16%
20304	728	15,930	1.3 MB	0.15%
445	490	3,546	17,589	0.1%

Top 5 Traffic Destination Countries

The mining pools connected to by cloud organizations were more often located in the United States, with 69% of all public mining traffic resolving to US systems. The Netherlands was second with 11%, and Germany was third with 9%.

This geographic breakdown appears to fit very closely to research performed on the mapping of Monero node topology and distribution, which detailed the geographic locations of known Monero nodes throughout the world. The research lists the United States as containing 51% of the heavily used Monero mining pools as well as 33% and 37% of the light- and medium-used mining pools. Germany, Netherlands, Canada, and France are also present in the top eight.

This corroboration of data allows Unit 42 researchers and cloud security personnel to more accurately enable defensive countermeasures to detect cryptojacking operations and prevent cloud resource misuse by attackers.

Table 7: Top Countries Where Mining Pools Connected to Cloud Organizations

Destination Country	Connections	Total Packets	Total Bytes	% of Total
United States	339,210	99,419,568	43.8 GB	69%
Netherlands	54,666	1,712,526	2.35 GB	11%
Germany	45,271	32,597,370	11.2 GB	9%
Canada	29,480	7,967,183	1.55 GB	6%
Ireland	21,130	3,827,620	713 MB	4%
Indonesia	1,799	39,759	11.6 MB	0.36%
France	791	23,356	4.05 MB	0.16%
Australia	742	17,061	4.1 MB	0.15%
United Kingdom	84	64,492	3.5 MB	0.02%
Russia	13	1,596	104 KB	0.002%

Cryptojacking Tool Spotlight

For this report, Unit 42 researchers focused on a cryptojacking tool called `Kinsing`, which is a GoLang H2miner cryptojacking operation first reported by Alibaba's cloud security team. This cryptojacking operation targets exposed Docker daemon API instances, evades security tools from Aliyun and Tencent, and performs network scanning operations using a tool called masscan, giving it worm-like capabilities. Once installed upon the compromised system, `Kinsing` will capture system information; download, install, and run masscan; and finally download and install the mining software `kdevtmpdsi`, which is a variation of the XMRig mining platform.

Of note, the malware authors inserted parts of William Shakespeare's "Hamlet" within the code. This is likely designed to distract, confuse, or taunt malware analysts, or to provide excessive noise to limit automated analytic tools. Additional identification markings of `Kinsing` include the creation of the temporary directory `/tmp/kinsing`. If a system contains this directory, it is likely that system is infected with `Kinsing`.

Table 8 shows known Kinsing malware samples performing network connections to the following hard-coded IP addresses with a specific functionality.

Table 8: Kinsing Malware Samples Connecting to Hard-Coded IP Addresses

IP Address	Malware	Function	Country
45.10.88.102	Kinsing	C2	Ukraine
91.215.169.111	Kinsing	C2	Russia
139.99.50.255	Kinsing	C2	Canada
46.243.253.167	Kinsing	C2	Russia
195.123.220.193	Kinsing	C2	Netherlands
178.170.189.5	kdevtmpfsi	Mining Operations	Netherlands
45.89.230.240	kdevtmpfsi	Mining Operations	Russia
106.15.38.249	kdevtmpfsi	Mining Operations	China
116.62.237.64	kdevtmpfsi	Mining Operations	China
116.62.241.154	kdevtmpfsi	Mining Operations	China
118.190.90.105	kdevtmpfsi	Mining Operations	China
120.79.193.21	kdevtmpfsi	Mining Operations	China

Unit 42 researchers were able to identify additional network communications originating from within the United States, although these are considered to have an extremely low occurrence. These connections were directed to the known `Kinsing` mining operation IP addresses in table 8. However, researchers were not able to find network connections directed to the `Kinsing` C2 nodes. This implies that the US-based cloud systems had been compromised and connected to the `Kinsing` C2 nodes prior to the discovery of network connections to the mining nodes.

Attribution for this mining software is currently unclear. However, it is interesting to note that the actors behind this tool have established the majority of their C2 systems within Russia, Ukraine, and the Netherlands. Interestingly, the majority of the systems for running the mining operation are located in China. This is likely done so that the network traffic for the mining operation would be less conspicuous (as most of the victims reside in the JAPAC region), whereas traffic back to the C2 system countries could raise suspicion. See figure 13 for an illustration of how attackers can use legitimate network traffic protocols to obscure their malicious actions. In the current example, by directing the majority network traffic to local or same region systems, attackers can further obfuscate malicious traffic patterns to evade detection.

Battle with Known Threat Actors

Unit 42 researchers found evidence of a resource battle between Kinsing mining operations and that of TeamTnT mining operations and the Docker cryptojacking worm Cetus. In the Unit 42 blog on Cetus, researchers identified an XMR wallet address, `85X7JcgPpwQdZXaK2TKJb8baQAXc3zBsnW7JuY7MLi9VYSamf4bFwa7SEAK9Hgp2P53npV19w1zuaK5bft5m2NN71CmNLoh`. This address is used in both Cetus and TeamTnT operations.

Interestingly, TeamTnT, Cetus and Kinsing all share very similar tactics, techniques and procedures (TTPs). Both tools have scanned networks for exposed Docker daemon APIs running on ports 2375, 2376, and 2377, using masscan, TeamTnT has also used used the open-source tool pnscan for scanning; both bypass Aliyun and Tencent security software; and both have historically targeted Redis instances. Finally, TeamTnT has been reported to have copied code from Kinsing that allows the malware to search a compromised system for AWS credentials, `~/.aws/credentials`, and AWS configurations, `~/.aws/config`. If either or both of these files are found, they are uploaded to a C2 server. This type of functionality exposes attackers' evolving efforts to further gain access to cloud environments.

There is also additional information linking Kinsing to SaltStack, an open source, Python-based management platform. The connections between these two also include a shared XMR wallet address—different from the one shown above—and the incorporation of "Hamlet" in the malware's source code.

Given the similarities between two distinct malware families, Kinsing and TeamTnT, there is a great amount of evidence that TeamTnT has been directly targeting Kinsing operations to increase its own claim to compromised resources. While not all of TeamTnT's TTPs match those of Kinsing, the clear facts are that TeamTnT is directly using Kinsing code and then hijacking Kinsing compromised systems. The long battle for cryptojacking resources appears to have no end in sight.

04

Conclusion and Recommendations

IAM Best Practices

IAM governance is a critical aspect of cloud security, and DevOps and security teams must treat it as such. While cloud providers deliver a good baseline for implementing a least-privileged approach to permissions, this breaks down as cloud adoption scales across multiple providers. While a comprehensive IAM strategy is outside the scope of this particular report, Unit 42 researchers recommend starting to look holistically at all your cloud accounts, keeping in mind the following key steps:

1. **Minimize the use of administrator credentials**. The number of users with admin access should be minimized, and admin credentials should only be used when absolutely necessary. The less frequently admin credentials are used, the less likely they are to get compromised.
2. **Simplify user management by using group or role**. Users in the same project tend to have similar permission requirements and can be placed in the same group or role. Instead of managing each user's permission policy, system admins should manage the permission policy of each group or role.
3. **Enable MFA**. MFA provides another layer of security in case the primary password is compromised.
4. **Configure a strong password policy**. Passwords still matter even in the age of more common MFA. The National Institute of Standards and Technology (NIST) recommends an eight-character minimum length and skipping character composition rules as they are painful for users. When MFA is enabled, passwords should only be changed if there is evidence of compromise. For additional guidance, see NIST SP 800-63-3.
5. **Rotate access tokens regularly**. Access tokens should be assigned short expiration periods to minimize the risk of compromised credentials. This is different from standard user accounts.
6. **Monitor IAM APIs**. All major CSPs have services to monitor IAM usage. These services help identify abnormal activities, such as brute-force attacks and logging from unrecognized devices or locations.
7. **Grant least-privileged access**. Only grant the least amount of permissions needed for a job. This will ensure that if a user or resource is compromised, the blast radius is reduced to only those few things the entity was permitted to do. This is an ongoing task that is best automated.
8. **Auto-remediate excessive privileges**. Entitlement audits should not be manual processes in the cloud. Environments change too rapidly, and the combination of user and machine roles makes manual auditing completely ineffective at any scale.
9. **Leverage cloud native security platforms**. Managing a large number of privileged users with access to an ever-expanding set of sensitive resources can be challenging. On top of that, cloud resources themselves have permission sets that need to be managed. Cloud native security platforms (CNSPs) like Prisma Cloud help leverage the identity of cloud resources to enforce security policies and ensure secure user behavior across multiple cloud environments.
10. **Harden IAM roles**. Never grant anonymous access (e.g., "Principal" : { "AWS" : "*" }) to any IAM role. Make role names difficult to guess by adding random strings. If the role is shared across AWS accounts, enforce unguessable External ID as another layer of protection.

Cryptojacking Defense Best Practices

Protecting cloud infrastructure against cryptojacking starts with leveraging the aforementioned identity best practices. While many cryptojacking attacks don't involve sophisticated tactics or techniques, Unit 42 researchers have discovered several worm variants that can periodically pull new scripts from C2 servers. Additionally, cryptomining malware has been found to evolve and perform additional functions outside of simple cryptomining operations—scanning operations, proxy usage, and C2 operations are becoming more common. These evolutions, especially regarding worm activity, allow mining operations to easily repurpose themselves to ransomware or exfiltration-focused malware, further entrenching attackers deeper within the cloud environment. These types of shape-shifting threats should not be ignored. The following is a list of best practices organizations can use to guard against cryptojacking in their cloud infrastructure:

1. **Authenticate all container service connections**. Never expose a Docker daemon to the internet without a proper authentication mechanism. Note that, by default, the Docker Engine is not exposed to the internet.
2. **Block all firewall ports by default**. Use firewall rules to allow list the incoming traffic to a small set of sources.
3 **Maintain a set of trusted images and registries**: In NIST SP 800-190 (Application Container Security Guide), the section on countermeasures for major risks (Section 4) says: "Organizations should maintain a set of trusted images and registries and ensure that only images from this set are allowed to run in their environment, thus mitigating the risk of untrusted or malicious components being deployed."
4. **Leverage threat intelligence feeds**. Review network traffic for any connections to mining pools.
5. **Invest in cloud native security platforms**. Cloud security solutions, such as Prisma Cloud, can identify malicious containers and prevent cryptojacking activities across all major CSPs.

Ready to Identify the Threats in Your Cloud?

Prisma Cloud analyzes more than 10 billion events every month. This analysis shows us that poor configuration, permissive behaviors, and lack of policies lead to many openings for bad actors and unidentified threats to exploit. By proactively detecting security and compliance misconfigurations as well as triggering automated workflow responses, Prisma Cloud helps ensure you continuously and securely meet the demands of your dynamic cloud architectures.

About

Prisma Cloud

Prisma™ Cloud is a comprehensive cloud native security platform with the industry's broadest security and compliance coverage—for applications, data, and the entire cloud native technology stack—throughout the development lifecycle and across hybrid and multi-cloud deployments. The integrated Prisma Cloud approach enables security operations and DevOps teams to stay agile, collaborate effectively, and accelerate cloud native application development and deployment securely.

Unit 42

Unit 42 is the global threat intelligence team at Palo Alto Networks and a recognized authority on cyberthreats, frequently sought out by enterprises and government agencies around the world. Our analysts are experts in hunting and collecting unknown threats as well as completely reverse-engineering malware using code analysis. With this expertise, we deliver high-quality, in-depth research that provides insight into tools, techniques, and procedures threat actors execute to compromise organizations. Our goal is to provide context wherever possible, explaining the nuts and bolts of attacks, as well as who is executing them and why, so that defenders globally can gain visibility into threats to better defend their businesses against them.

Methodology

All research took place from May to August 2020 and included the Americas, EMEA, and JAPAC regions. IAM research was conducted using publicly available data from GitHub. Unit 42 researchers used the GitHub Search API to retrieve files that might contain AWS Account Resource Numbers (ARNs). The researchers then performed data mining offline to extract potential AWS account IDs and role names from the GitHub files. Each valid AWS ID was then checked with a list of role names using AWS APIs. These findings may not be representative of the entire base of CSP customers.

Palo Alto Networks Prisma Cloud

Prisma Cloud trend data utilizes multiple threat intelligence sources. Unit 42 researchers utilized proprietary data sources to gather organizational alert and event data. This data was anonymized, analyzed, and then compared to the results of previous cloud threat report analytics to produce trend information. Datasets from GitHub were also used to provide analysts with third-party data to bolster and validate the proprietary data analytics results.

Palo Alto Networks WildFire

The cloud-based WildFire® malware prevention service employs a unique multi-technique approach, combining dynamic and static analysis, innovative machine learning techniques, and a groundbreaking bare metal analysis environment to detect and prevent even the most evasive threats.

Palo Alto Networks AutoFocus

The AutoFocus™ contextual threat intelligence service provides the intelligence, analytics, and context required to understand which attacks require immediate response, as well as the ability to make indicators actionable and prevent future attacks.

GitHub

GitHub is the largest public/private source code repository for software development version control using Git. The GitHub searching API is an easy way to find topics, commits, or code across all the repositories on GitHub.

Authors

Jay Chen, Nathaniel “Q” Quist, and Matthew Chiodi

3000 Tannery Way
Santa Clara, CA 95054

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087

www.paloaltonetworks.com

Palo Alto Networks assumes no responsibility for inaccuracies in this document and disclaims any obligation to update information contained herein. Palo Alto Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. unit-42-cloud-threat-report-2h-2020-100120