Anatomy of a Cloud Breach: How 100 Million Credit Card Numbers Were Exposed.

July 2021

zscaler™

Table of contents

Financial Institution Data Breach	**3**
Likely Attack Scenario	**4**
Recommendations	**7**
AWS Governance Practices	7
AWS Configurations	8
Auto-remediation	9
Conclusion	**9**
About Zscaler	**10**

Major Financial Institution Data Breach

The US-Based major financial institute data breach two years ago, which exposed the personal data of more than 100,000 customers, was one of the most devastating data breaches of all time. A trusted financial services brand, has been a leader in digital transformation within the banking industry and a sophisticated user of cloud infrastructure. Careful analysis of this breach gives helpful insight into prevention methods that will benefit organizations storing confidential information in the cloud.

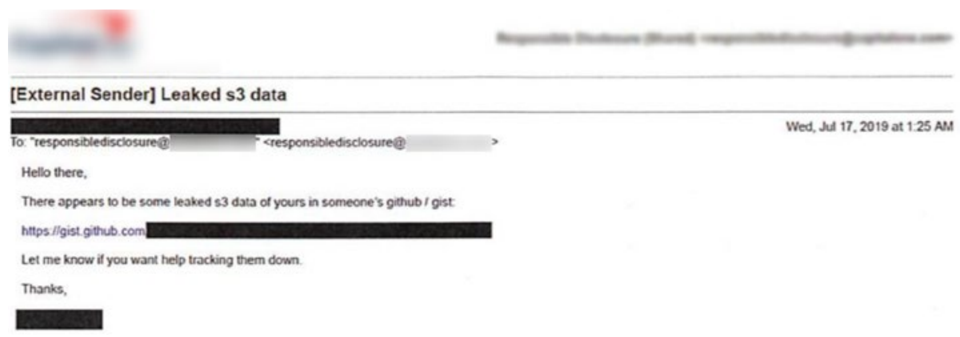
Soon after the breach was reported, unwarranted speculation spread on the internet, including suggestions that a single product or a set of professional services could have prevented such an attack. Taking advantage of information that has since become available, Zscaler has updated this research note with a technical illustration of the attack and possible ways to prevent such data breaches.

On July 29, 2019, the FBI arrested Paige A. Thompson (also known by the alias "erratic") for allegedly hacking into Financial Institution databases and stealing the data[1]. Financial Institute disclosed that the event affected approximately 100 million individuals in the United States and approximately 6 million in Canada. It also includes data loss of approximately 1 million Social Insurance Numbers of Canadian credit card customers, about 140,000 Social Security numbers, and 80,000 linked bank account numbers of the credit card customers[2].

AWS provided their assessment of the incident: "Financial Institution outlined in their public announcement, the attack occurred due to a misconfiguration error at the application layer of a firewall installed by Financial Institution, exacerbated by permissions set by Financial Institution that were likely broader than intended. After gaining access through the misconfigured firewall and having broader permission to access resources, we believe an SSRF attack was used (which is one of several ways an attacker could have potentially gotten access to data once they got in through the misconfigured firewall)."[3]

The criminal complaint[4] and indictment[5] documents provided additional insights. According to the FBI, the following happened.

[1]Seattle Tech Worker Arrested for Data Theft Involving Large Financial Services Company
[2]Information on the Financial Institution Cyber Incident
[3]Amazon Letter to Senator Wyden RE Consumer Data
[4]Department of Justice - complaint document
[5]Department of Justice - indictment document

Timeframe	Activities
January 2019 to July 2019	"Erratic" used TOR (The Onion network) to attempt connections, develop command scripts and other readiness activities. Multiple connections were made to connect to the servers, download pilot files to test out the end-to-end scenarios
April 21, 2019 Data leaked	Financial Institution's customer information (700 folders worth of data) were posted on the erratic's public GitHub pages IP address of a specific AWS server Code for 3 commands used for the attack • **Get Credentials** - First command when executed obtained security credentials known as ****-WAF-Role account (an IAM account) for an elevated role access AWS Web Application Firewall (WAF) • **List Buckets** - Second command, when executed, used the security credentials *****-WAF-Role account to list files and folders (aka S3 buckets) • **Download Files** - Third command, when executed used the *****-WAF-Role account to download files that were accessible by the credentials.
June 26, 2019	"Erratic" shared the collected information casually on a public Slack channel used by a Meetup group to communicate with its members.
July 17, 2019	Financial Institution received an email informing about leaked data  **[External Sender] Leaked s3 data** [redacted] Wed, Jul 17, 2019 at 1:25 AM To: "responsibledisclosure@[redacted]" <responsibledisclosure@[redacted]> Hello there, There appears to be some leaked s3 data of yours in someone's github / gist: https://gist.github.com[redacted] Let me know if you want help tracking them down. Thanks, [redacted]

Likely Attack Scenario

It's important to note that, unlike many other high-profile AWS breaches, the Financial Institution S3 buckets were not left externally exposed to the internet. Misconfiguration along with human and non-human identities with excessive high-risk permissions wield enormous power to disrupt business operations, often without organizations' awareness.

While the indictment is not specific about the nature of the attack, the following is our best guess regarding the likely steps taken by "erratic" to compromise the data.

STEP 1: Identified and Exploited Misconfigured WAF

The attacker identified a misconfigured WAF that enabled accessing the corresponding AWS EC2 instance/ ECS task *metadata* using Server-side Request Forgery (SSRF) and call the metadata service endpoint using

http://169.254.169.254/iam/security-credentials command

The endpoint must have returned a role (according to the indictment '*****-WAF-Role')

STEP 2: Gain temporary credentials

Using the role name, the attacker then queried the specific endpoint to gain access to temporary credentials

http://169.254.169.254/iam/security-credentials/*****-WAF-Role

The above would return the full set of temporary credentials

```
{

AccessKeyId: "<access key>",

SecretAccessKey: "<secret key>",

}
```

STEP 3: Gain access to S3 buckets by calling AWS S3 list and Sync

CLI commands

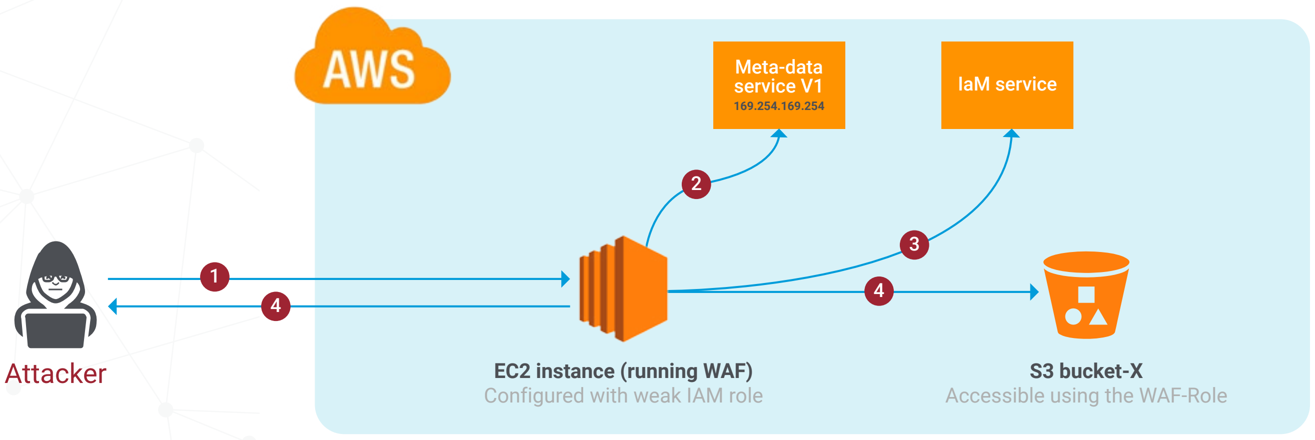
```
$ aws s3 ls
```

The ls command would list all the S3 buckets accessible using the IAM role

```
$ aws s3 sync s3://somebucket
```

The sync command would download all resources from the 'some bucket

In summary, the most likely root cause of the attack was a poor security architecture design that exposed S3 buckets via AWS WAF/EC2 instance to anyone with an IAM role. While S3 buckets were not exposed to the Internet like many other breaches, an EC2 instance with an excessive IAM role might have been the culprit. The deployed architecture would have looked something like this. It was a trivial step to "compromise" the poorly configured WAF.


AWS
Meta-data service V1
169.254.169.254
IaM service
1
2
3
4
4
Attacker
EC2 instance (running WAF)
Configured with weak IAM role
S3 bucket-X
Accessible using the WAF-Role

2 Invokes meta-data v1 service to get security credentials for a specific role (***WAF-Role).

```
$ curl http://169.254.169.251/iam/security-credentials/(sepcific role ***WAF-Role)
returns credential {AccessKey1, Secret_passcode_1}
```

3 Uses credentials and assumes the WAF-Role that is permitted to access S3 buckets

4 Lists all S3 buckets, pickes one of them and “sync” (download) its content

```
$ aws s3 is
$ aws s3 syncs3://somebucket > would download all objects from somebucket
```

AWS Components	Predicted Configurations and Usage During the Attack
Firewall	A misconfiguration of AWS Web Application Firewall.
IAM role access to S3	Excessive permissions to an IAM role allowing access to private S3 buckets “Financial Institute determined that the first command, when executed, obtained the security credentials for an account known as ***WAF-Role that, in turn, enabled access to certain of Financial Institute folders at Cloud Computing Company. (III.A.11)”
SSRF attack using AWS metadata service	An SSRF attack tricks a server into executing commands on behalf of a remote user, enabling the user to treat the server as a proxy for his or her requests and get access to non-public endpoints. An EC2 instance was likely used to access **AWS metadata service**, accessible at **http://169.254.169.254.** A particularly important function of the metadata service is to provide temporary credentials that give the node access to other AWS services based on a permission policy defined in the instance’s IAM role. IAM roles are an alternative to long-lived user access keys and secrets; rather than hard coding an access key into an application’s configuration, the application simply requests credentials from the metadata endpoint periodically.

AWS Components	Predicted Configurations and Usage During the Attack
EC2	Exposing unnecessary shell access, probably from a left-over development / staging or a production debugging deployment.
S3 bucket	Financial Institute determined that the third command (the “Sync Command”), when executed, used the ***-WAF-Role to extract or copy data from those folders or buckets in Financial Institute storage space for which the ***-WAF-Role account had the requisite permissions. (III.A.11) Suggests that the attacker had access to ‘aws s3 sync’ command. i.e. the IAM role used S3 list (e.g. aws s3 ls) and Read access (e.g. aws s3 sync).

Recommendations

AWS Governance Practices

The following AWS governance practices would prevent such attacks:

1. Review all access paths and permissions from human identities or non-human identities (e.g., ec2 machine) to data storages (e.g., S3 buckets). Use Cloud Infrastructure Entitlement Management (CIEM) solutions to automate the detection of over-privileged identities and over-exposed data.
2. AWS lists a few basics here https://aws.amazon.com/premiumsupport/knowledge-center/secure-s3-resources/. Zscaler Workload Posture with Cloud Security Posture Management (CSPM) automates 100’s of these policies. It provides security and compliance views across all AWS, including verification of the use of version 2 of the EC2 Metadata service instead of version 1 that would have prevented the compromise of the credentials.
3. Use CloudTrail, CloudWatch, and/or AWS lambda services to review and automate specific actions taken on S3 resources.
4. Ensure each application, EC2 instance, or autoscaling group has its own IAM role. Do not share roles across unrelated applications.
5. Scope the permissions of each role to enable access only to the AWS resources required. The “WAF” role described above did not require access to list S3 buckets “in the normal course of business” (according to the indictment).
6. If possible, include a “Condition” statement within the IAM role to scope the access to known IP addresses or VPC endpoints.
7. Don't allow EC2 instances to have IAM roles that allow attaching or replacing role policies in any production environment.
8. Clean up unused cloud resources (especially EC2 instances and S3 buckets) leftover from prior development or production debugging efforts.

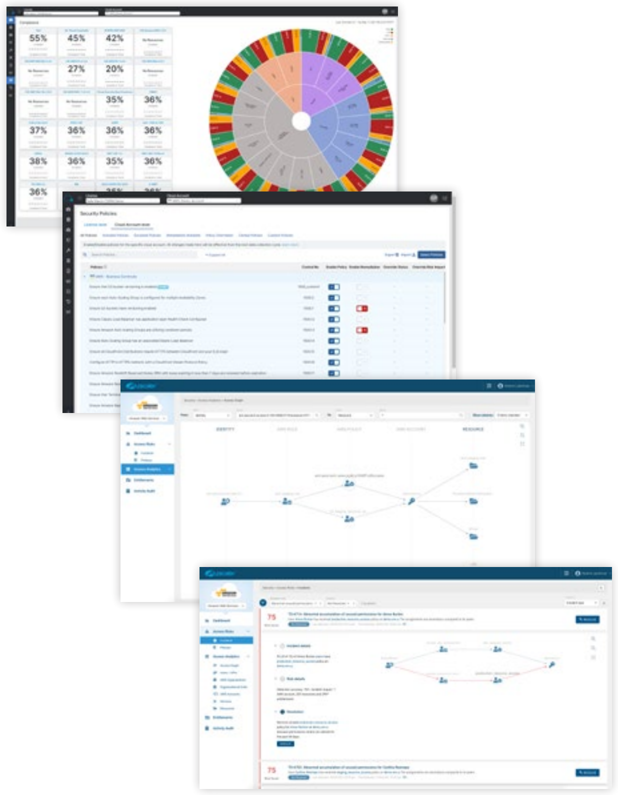
AWS Configurations

Following AWS configurations would prevent such attacks:

1. **AWS IAM:** Enforce least privileged IAM controls
2. **AWS IAM:** Ensure IAM policies are attached only to groups or roles
3. **AWS S3:** Ensure AWS S3 buckets do not allow public READ access
4. **AWS S3:** Ensure AWS S3 buckets do not allow public READ_ACP access
5. **AWS S3:** Ensure AWS S3 buckets do not allow public WRITE_ACP access
6. **AWS S3:** Ensure S3 buckets do not allow FULL_CONTROL access to AWS authenticated users via S3 ACLs
7. **AWS S3:** Ensure that Amazon S3 buckets access is limited only to specific IP addresses
8. **AWS S3:** Ensure S3 buckets do not allow READ access to AWS authenticated users through ACLs
9. **AWS S3:** Ensure S3 buckets do not allow FULL_CONTROL access to AWS authenticated users via S3 ACLs
10. **AWS S3:** Ensure all S3 buckets have policy to require server-side and in transit encryption for all objects stored in bucket
11. **AWS Networking:** Ensure no security groups allow ingress from 0.0.0.0/0 to port 22
12. **AWS Networking:** Ensure Application Load Balancer (ALB) with administrative service: SSH (TCP:22) is not exposed to the public internet
13. **AWS Networking:** Ensure no security groups allow ingress from 0.0.0.0/0 to port 22 (SSH)
14. **AWS Networking:** Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389 (RDP)
15. **AWS - Audit and Logging:** Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket
16. **AWS - Audit and Logging:** Ensure CloudTrail is enabled in all regions
17. **AWS - Audit and Logging:** Ensure CloudTrail trails are integrated with CloudWatch Logs
18. **AWS - Audit and Logging:** Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible
19. **AWS - Monitoring:** Ensure a log metric filter and alarm exist for CloudTrail configuration changes.

The above list of configurations is only for illustration purposes. For a comprehensive list, check out **Zscaler CSPM documentation.**

These configurations can be part of manual deployment documentation or, ideally, be part of the Infrastructure as Code (IaC) automation within DevOps pipelines. It would prevent these misconfigurations from getting into production in the first place. A Workload Posture management solution like Zscaler could be used by Cloud Ops or DevOps team to validate their security posture in pre-production and production environments continuously.



Auto-remediation

The next level of defense would be auto-remediation of misconfigured resources and IAM permissions. Zscaler also provides an integrated policy-driven framework to auto-remediate resources that deviate from the defined security policies. Refer to our documentation on AWS auto-remediation.

Conclusion

The industry was understandably shocked by the sheer scale of the attack against one of the most trusted brands operating on one of the most secure infrastructures. There are several lessons to be learned.

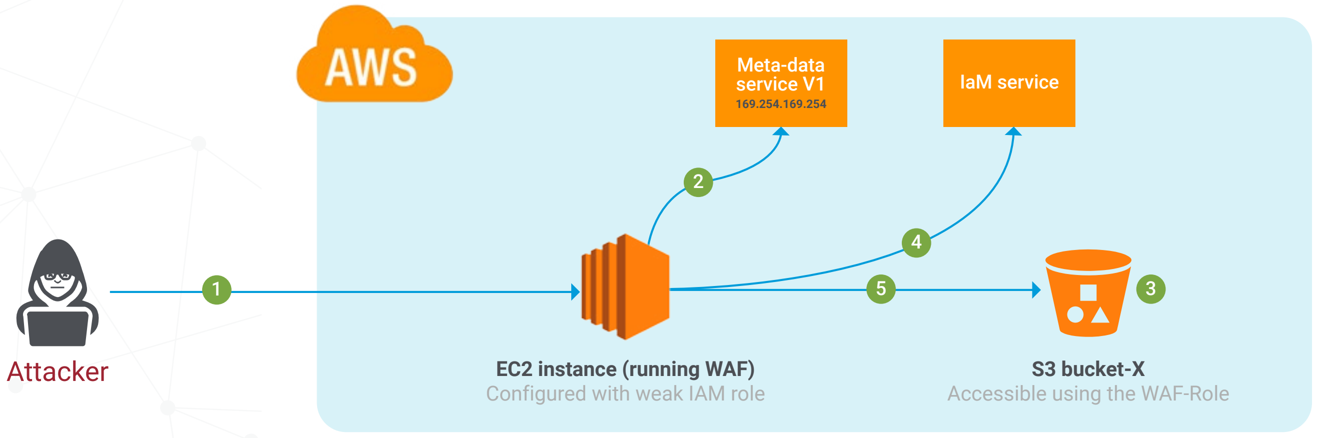
Many cloud customers misunderstand the shared responsibility model or struggle with knowledge gaps as they transition from physical data centers to the cloud.

The Financial Institution intrusion resulted from a flawed configuration performed as part of the bank's security responsibilities and discrete from the underlying AWS-secured infrastructure. Moreover, the Financial Institution's security team was not aware of unauthorized identities and credentials used to access the AWS Management Console. This lack of awareness allowed the hacker to enter the environment.

Misconfiguration and excessive permissions can cause catastrophic losses, and the Financial Institution experience is only one of many known cases.

We anticipate seeing more breaches at companies that have not kept up with the pace of change in their cloud environments and have not implemented adequate cloud security, IAM controls, and compliance assurance.

Zscaler is convinced that the critical flaws that enabled such a devastating breach can be avoided when organizations deploy a comprehensive solution that combines analysis of both the cloud service configurations and the IAM configuration to minimize the attack surface.


AWS
Meta-data service V1
169.254.169.254
IaM service
Attacker
1
2
3
4
5
EC2 instance (running WAF)
Configured with weak IAM role
S3 bucket-X
Accessible using the WAF-Role

Few examples

CSPM Validates infrastructure	1	Ensure all EC2 Instances are configured to exclusively use Instance Metadata Service V2
	2	Ensure all S3 buckets employ encryptioin-at-rest
CIEM Validates permissions	3	Ensure least-privilege access is granted to S3 buckets containing sensitive data
	4	Validate anomalous permissions using peer group analysis for EC2 instances

About Zscaler Workload Posture

Zscaler Workload Posture automates security, entitlements, and compliance in the cloud, delivering continuous visibility and enforcing adherence to the most comprehensive set of security policies, least privileges, and compliance frameworks.

Contact Zscaler

Zscaler, Inc.
120 Holger Way
San Jose, CA 95134
+1 408.533.0288
www.zscaler.com