ARTICLE

Cloud is not just somebody else's computer

New paradigms for security threats in modern cloud applications

Cody Burkard
Senior Cloud Security Consultant

AFTER READING THIS ARTICLE, YOU WILL:

- Become familiar with modern security pitfalls in cloud environments
- Learn how to model cloud-based threats against applications
- Know how cloud-based threats impact application security in the cloud

Make no mistake; application development for public cloud infrastructure is the new norm. Whether this is because of the speed of development and the lack of infrastructure maintenance, the native automation capabilities in cloud environments, or a variety of other factors, it is safe to say that application development in the cloud is here to stay. This leads to the question: what new security considerations are there for cloud-native applications?

Modern cloud security issues

The term "cloud security test" may invoke a number of thoughts and assumptions depending on whom you speak with. To some, this simply means web application testing when the application is hosted in a cloud environment. To others, this could mean a traditional penetration test, where the goal is to gain access to a certain objective within infrastructure on a Virtual Network. We won't debate the details here, but for the sake of this article, we choose to define a cloud security test according to the following goal:

"Identify security weaknesses and vulnerabilities in the cloud environment that would allow an adversary to impact the confidentiality, integrity, or availability of any service within or otherwise dependent on the environment"

This definition is intentionally quite broad. It is almost as vague as the term "cloud security test" - it encompasses almost any type of vulnerability that could exist within the cloud or its infrastructure, be it on a virtual machine or through a cloud configuration. However, the results of these tests commonly centre on a few security issues. Specifically, amongst the most prevalent issues in cloud deployments are networking misconfigurations, poor secret management, and Identity and Access Management (IAM) misconfigurations. Let's take a closer look at why each is prevalent in cloud environments.

Network misconfigurations

Most cloud resources in a modern cloud provider can be connected to a network, and it is possible to enforce network access control lists (ACLs) on inbound or outbound connections to the resource. However, it is also possible to run most resources without connecting them to a virtual network. Many times, this is the default behaviour for cloud services.

For example, Storage Accounts in Microsoft Azure and S3 Buckets in AWS are by default accessible via direct network connections from the Internet. While both require authentication in order to connect, an adversary with a stolen secret could connect from their own environment. To limit network access, custom bucket policies in S3 or IP address whitelisting in Azure Storage must be utilised to limit network connectivity. During cloud security reviews, we frequently observe that developers believe direct internet connections to cloud resources, such as storage, are safe. Common reasoning behind this assertion ▸

The focus on internal threats is not a novel concept, and is very much in-line with traditional approaches to offensive security on technologies such as Active Directory. However, the nuances in the details are interesting.

follows that the cloud provider manages the storage services, and because storage services still require authentication to access the data. At first glance, this logic seems reasonable because Microsoft will automatically patch vulnerabilities on the host, and the secrets for the storage services are cryptographically secure, so traditional attacks seem infeasible. However, as the rest of this article shows, there are additional security concerns in cloud infrastructure that cloud administrators and developers should consider.

Poor secret management

Most cloud resources require a secret in order to connect or interact. In Azure, for example, authentication with a secret key is required on storage services, caching services, messaging services, and managed API endpoints in cloud environments, as well as many other frequently used services. As an application developer in cloud environments, this means that these secrets need to be stored somewhere safe, automatically updated, and should never be leaked.

Because of the abundance of secrets and the constant need to update them, managing and protecting these secrets is a difficult task in cloud. During development and deployment, it is very common for plaintext secrets to be stored in storage services, local developer systems, or in configurations throughout the environment. Keep in mind that when an adversary steals one of these secrets, they can authenticate to the resource to which it belongs.

IAM misconfigurations

One of the most significant differences between cloud infrastructure and on-premise infrastructure is the management model for each. In the cloud, the traditional, physical installation process of new hardware has been replaced by web APIs. In essence, each cloud is one huge web application that integrates with sophisticated virtualisation infrastructure as a backend, and lets users create their virtual environments via a web interface or API.

This means that user permissions on this "web app" are extremely important. A developer should not have access to modify the network, in the same way that developers in on-premise applications do not have access to the datacentre. In practice, each cloud provider governs user permissions differently to solve this problem. For example, AWS uses IAM policies while Azure uses role-based access control (RBAC) roles. However, the principle is the same: limit user access to specific cloud services, and to specific actions on those services.

However, as cloud environments grow, these IAM assignments become more complicated. Maintaining fine-grained IAM assignments creates management overhead, and frequently leads to mistakes. If an adversary already has limited access to an environment through a previous vulnerability or as an insider, IAM misconfigurations can lead to the adversary gaining more access to the network than they would with a proper configuration. In some cases, misconfigurations may even allow an adversary to escalate their IAM privileges.

This is a particularly dangerous type of misconfiguration, but its impact is also very dependent on the IAM role that an attacker can access. For example, it is more dangerous if an attacker can add new data to a storage service than if an attacker can simply read cloud configurations. We will explore this further in a later section.

Identity is the new perimeter

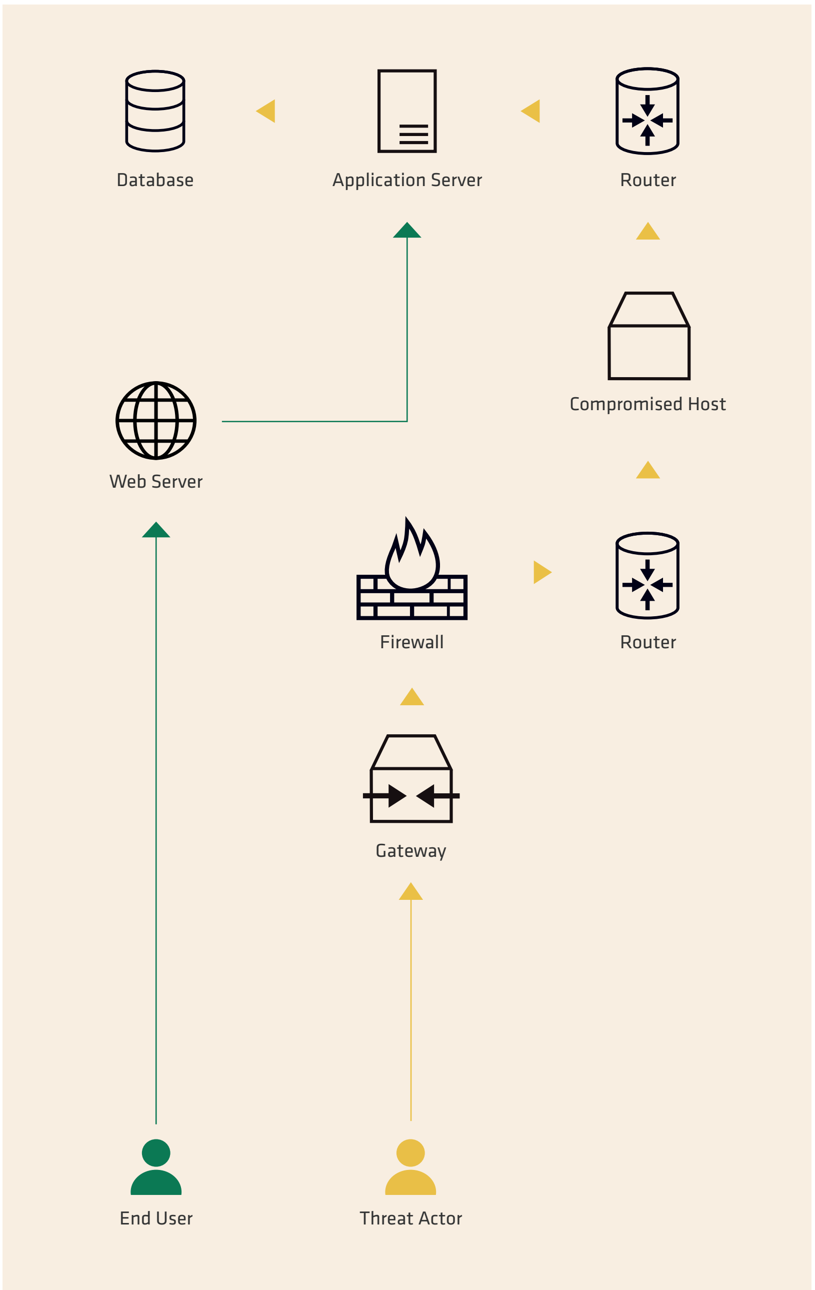
A brief analysis of the three security issues discussed above yields an interesting characteristic about cloud security: most security issues in cloud focus on defence in depth. Whether we discuss network attack surface, secret management, or IAM, the insider threat or an adversary with a compromised user account is the most likely to exploit each. This supports one of the modern security principles in the cloud: identity is the new perimeter.

This assertion of identity being the new perimeter is also supported by the prevalence of attacks we see – namely phishing attacks, which are often targeted specifically towards capturing identities. Most threat reports highlight phishing as one of the main techniques employed by sophisticated threat actors, likely due to the high success rate. If the impact of a phishing attack is a stolen IAM role in cloud, and the attacker has access to modify infrastructure via API calls, it is no wonder that securing user identity has become such a large focus.

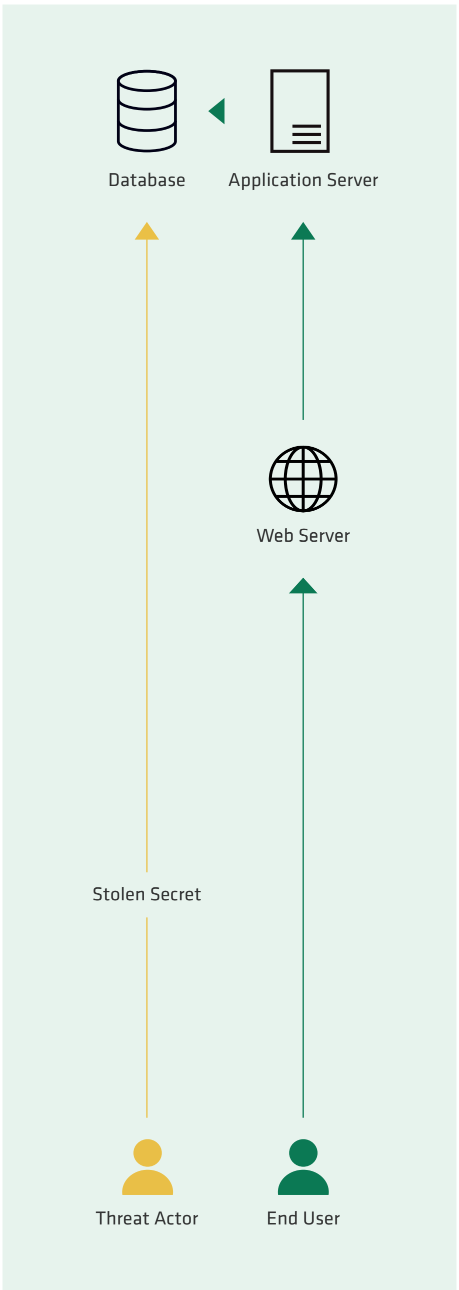
The focus on internal threats is not a novel concept, and is very much in-line with traditional approaches to offensive security on technologies such as Active Directory. However, the nuances in the details are interesting. For example, consider a scenario with both excessive network exposure and an IAM based attacker. If the attacker escalates their privileges entirely within the cloud provider IAM roles, they can directly connect to internal resources without the need to pivot or move laterally through the network. This situation makes the attacker's job much easier by removing all of the obstacles in the network, and making it easy to maintain persistent access to the internal resource without compromising any hosts.

The diagram below shows two example scenarios that highlight this point. In scenario B, the IAM-based attacker can directly access a cloud database with a stolen secret. In scenario A, the traditional attacker must traverse a number of network devices to gain access to the database.

If this type of thinking still seems outlandish, consider a real world case study: the breach of Capital One in 2019 that resulted in the loss of personal information for over 100 million customers. A public FBI complaint includes some details from a technical analysis of the breach, which is enough to gain ▸



Scenario A: Example of an attacker gaining access to on-premise application database



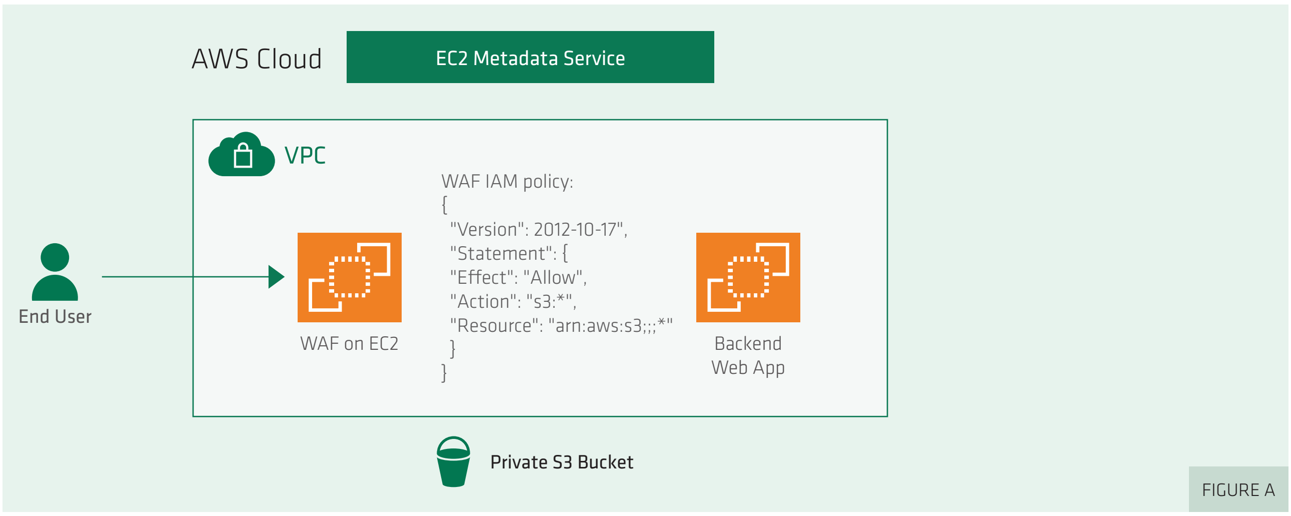
Scenario B: Example of an attacker gaining access to cloud application database a basic understanding of the attacker's path. To begin with, a small diagram of one potential representation of the Capital One environment is included below in Figure A.

The S3 Bucket is not connected to a virtual network. The WAF is running on an EC2 instance (Virtual Machine) in AWS, and has an IAM role attached, as seen in the diagram. The IAM role allows all actions on all S3 Buckets in the account, since it's using wildcards, which means it can access all data on those S3 Buckets. So what does this mean in practice?

The scope and permissions of the WAF's IAM role make it possible for the WAF to read the contents of the internal S3 Bucket, while the public network access of the S3 Bucket makes it possible for an authenticated user to read the contents of the bucket directly over the internet. This creates a dangerous scenario, where an attacker with access to the WAF can impersonate the WAF's identity, and access the S3 Bucket from their own location. The industry broadly assumes that this is exactly what happened. Most analyses suggest that a server-side request forgery (SSRF) vulnerability allowed the attacker to steal an IAM temporary credential from the WAF, and directly dump the contents of the S3 Bucket, as seen in Figure B.

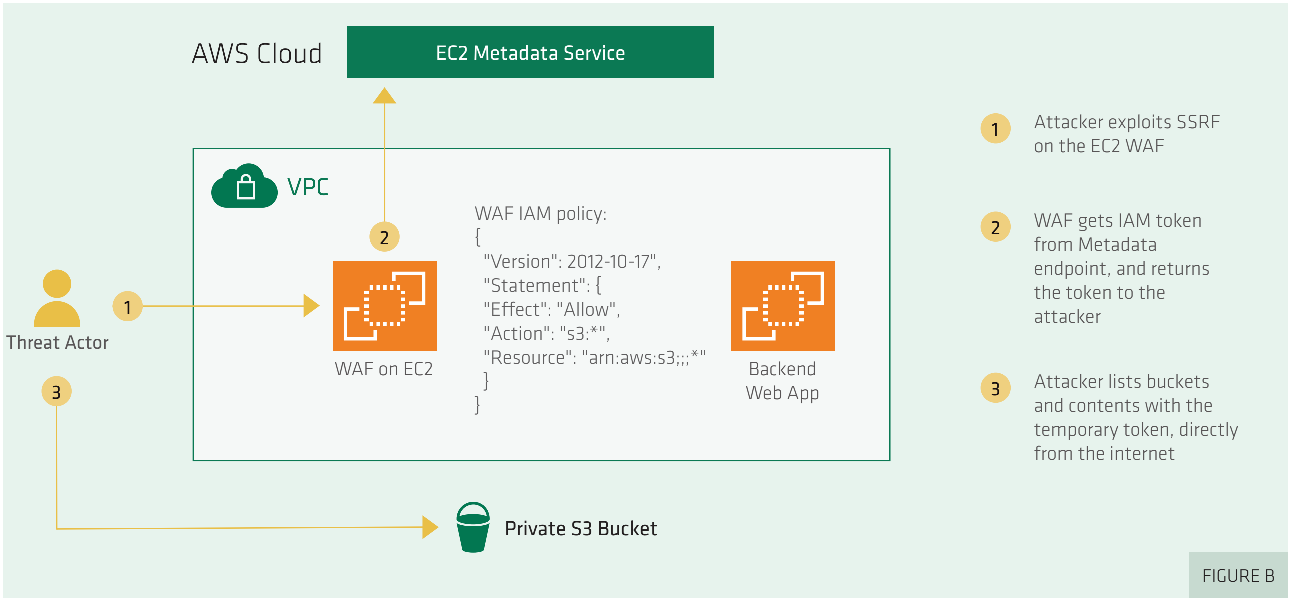
Based on the presumed setup in AWS, the root cause for this issue lies in the overly permissive IAM role on the WAF and the public network availability of the S3 Bucket. The SSRF in the WAF gave the attacker the first step into the environment, but the misconfigured network and IAM policies made it possible to pivot to new resources. In short, this breach offers a simple lesson that we as an industry have known for years: traditional security principles such as defence in depth always apply, and we simply need to adopt them to new environments.

Because of the centrality of IAM within cloud security, attackers are more focused on acquiring secrets and credentials, such as the temporary access token, or escalating their privileges to access more capabilities on the cloud APIs. By escalating privileges, adversaries can avoid the need for complex application-level vulnerabilities that are only



```
WAF IAM policy:
{
 "Version": 2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Action": "s3:*",
 "Resource": "arn:aws:s3;;;*"
 }
}
```

FIGURE A



```
WAF IAM policy:
{
 "Version": 2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Action": "s3:*",
 "Resource": "arn:aws:s3;;;*"
 }
}
```

1 Attacker exploits SSRF on the EC2 WAF

2 WAF gets IAM token from Metadata endpoint, and returns the token to the attacker

3 Attacker lists buckets and contents with the temporary token, directly from the internet

FIGURE B

exploitable at the network layer. This also provides them with new attack capabilities, such as directly modifying existing infrastructure in the cloud.

The impact on application security in the cloud

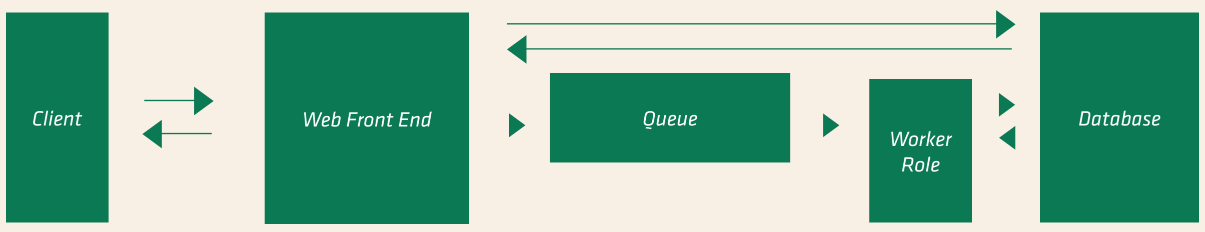
So how should this influence the mindset of application developers? We are talking about IAM and networking, but how much of this is the developer actually responsible for? And how do these issues impact their applications? The short answer again comes back to defence in depth and essentially reducing the blast radius. Developers should at a minimum be aware of these threats, and ideally assume a breach scenario to limit the impact of an internal threat. Let's consider a practical example.

Consider an API that handles banking transactions, which is built in a WQW architecture within Azure. This API should never suffer downtime, and must provide non-repudiation on all transactions. This means that it must be possible to validate the authenticity of each transaction and verify that each transaction occurred as was intended. The consequences for attacks against such an application are high, and we can assume the presence of a sophisticated threat actor.

A traditional approach to the threat modelling of such an application may look like what you find in Figure C.

Based on this diagram, we assert that the worker role only polls from the queue, and the queue only contains data from the web-worker. While attacks such as second-order ▸

What is Web-Queue-Worker Architecture?



A WQW architecture is built to take advantage of scalability and managed messaging services in cloud environments. These architectures include three main components: a front-end, a queue, and a worker role. The lightweight HTTP-based front-end service normally handles requests with low-computational requirements, such as pushing new data to the internal queue, or communicating with an external microservice. The worker role handles complex business logic and long-running computation tasks. These two services can easily be scaled up or down using cloud-native features. Most public cloud vendors provide some type of messaging system as a PaaS offering. Often there are multiple options. In Azure, you may use Storage Queues, Service Bus, or Event Hubs. AWS includes options such as Simple Queue Service (SQS), Simple Notification Service (SNS), or Kinesis. GCP offers Task Queues and Pub/Sub.

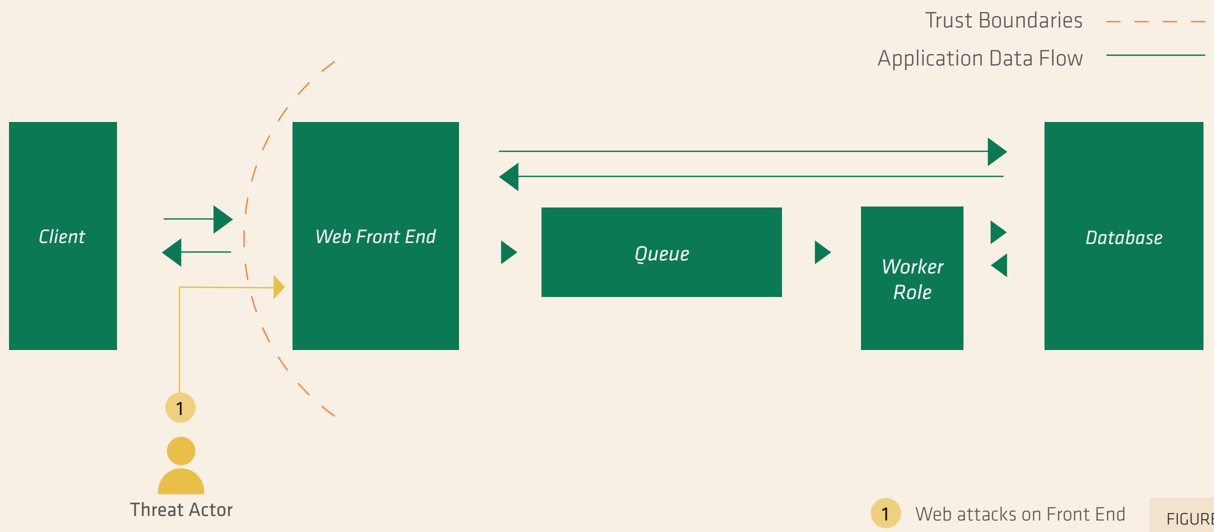


FIGURE C

injection against the worker role are possible, we can just prevent them with sanitisation and validation on the web role, which protects the worker role. This could be a reasonable approach in a traditional on-premise application security architecture.

However, consider the cloud-based threat actors that possess a stolen secret or have escalated to privileged Azure RBAC roles. Those IAM roles may provide an attacker with new attack surfaces on the application. If the IAM role allows the attacker to modify the network, the adversary may even whitelist their own IP on the queue, giving themselves direct network access. This is an entirely new threat scenario, based entirely on a user with access to Azure roles.

Attacking from the inside out

Based on a threat actor in this scenario, we must adjust the previous threat model. However, the new threat model is entirely dependent on the attacker's capabilities, which in turn are dependent on the RBAC roles or secrets they acquired.

So, let us consider what an attacker's capabilities would be if they acquired some of the built-in Azure roles. Figure D shows that if an attacker gains access to either the Owner or Contributor roles, they can make administrative changes to the entire application, so that developers cannot add any protection. Attackers with lower privileges that are limited to accessing the queue, such as Storage Contributor, Storage Queue Data Contributor, or Storage Queue Data Message Processor still pose a risk to the application; however, developers are in a position to mitigate these risks.

So let's consider an adjusted threat model that takes these roles into account, and assumes the cloud-based attacker has gained read and write permissions on the storage account using the Storage Contributor role as shown in Figure E.

There are now two new trust boundaries – one on the input of the queue, and the other on the output. This is a much more powerful attacker, because they are no longer limited to external attacks on the web front end. In addition, the

Role	Capabilities	Scope	Developer can mitigate if compromised
Owner	Assign roles, full administration access on all resources	Web role, queue, worker role	✗
Contributor	Full administration access on all resources	Web role, queue, worker role	✗
Storage Contributor	Modify queue network access, list keys for queue	Web role, queue, worker role	✓
Storage Queue Data Contributor	Read/write/delete items on queue	Web role, queue, worker role	✓
Storage Queue Data Message Processor	Read/delete items on queue	Web role, queue, worker role	✓

FIGURE D

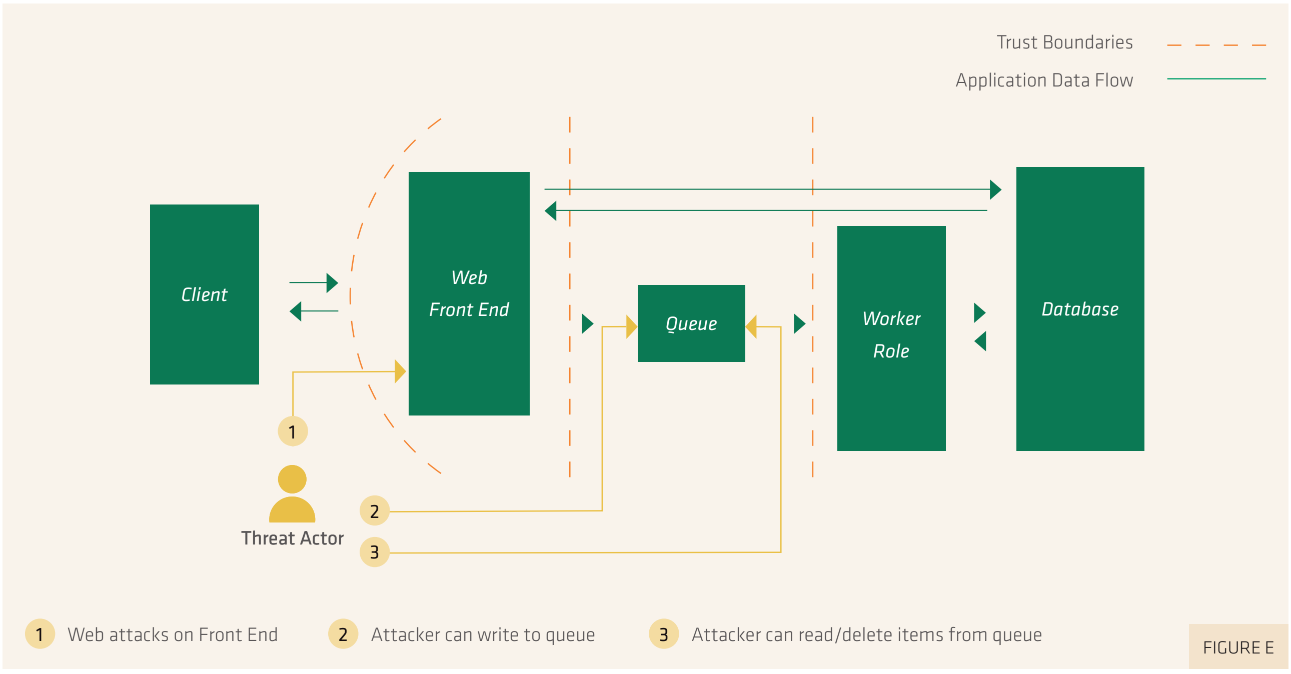

Trust Boundaries
Application Data Flow
Client
Web Front End
Queue
Worker Role
Database
1
2
3
Threat Actor
1 Web attacks on Front End
2 Attacker can write to queue
3 Attacker can read/delete items from queue

FIGURE E

attacker can read or write data on the internal messaging system for the application. This poses a challenge for the developers of the application: how do you protect against this threat, and what attacks can still be mitigated?

To begin with, consider a few possible attacks in this scenario:
1. The adversary pushes data in an unexpected format to the queue, and attempts to trigger web app attacks on the worker role.
2. The adversary pushes properly formatted data to the queue, and attempts to trigger fraudulent financial transactions.
3. The adversary reads and deletes financial transactions from the queue, preventing them from executing.

Each of these scenarios is catastrophic for the application, and developers should consider how they may protect against them.

There are a variety of ways to protect against these threats, so we won't explore all of the possibilities. A simple approach to mitigation may resemble the following:
Threat 1 and 2: Sign data using the web role, and verify the signature of each transaction on the worker role. Any time the worker cannot verify the signature, it should not process the transaction, and it should trigger an alert.
Threat 1 and 2: In the worker role, perform validation on the data format and validate all data types received from the queue.
Threat 1: Verify the signature before performing any dangerous function in the worker, such as deserialisation.
Threat 3: Number each transaction in the web role, and sign the transaction number along with the transaction data. The worker role should verify that no transaction in the chain is missing.

The takeaway here is the attack surface on an application changes when considering IAM-based threats in the cloud, and application developers are in a position to, and should consider how to protect against these types of threats.

Key takeaways for offense and defence

Secure application development in the cloud has fundamental differences with the traditional approach to application security. The primary difference is the IAM-based and stolen secret-based adversaries, as well as the prevalence of network exposure issues in the cloud. This means that the attack surface of an application is constantly in flux, and depends on each modification to IAM roles within the cloud environment. Within the security profession, employees in each role need to consider these differences and understand how they should adapt their approach to security in order to solve these challenges.

For Developers:

Developers should use threat models to highlight these differences during the development process. During this threat modelling process, consider the following questions:
- How could a stolen secret affect this application?
- How could IAM-based attackers with different permissions attack this application?
- Can network access to this application be limited further?
- How can defence in depth principles be applied to the application development process to protect against cloud-based threats?

For Security Architects:

Security Architects face the challenge of managing IAM roles in their organisation. It is important for architects to adopt centrally managed IAM and enforce the principle of least privilege to prevent these issues from arising. This becomes particularly challenging as more organisations adopt multi-cloud approaches, so consider cloud-agnostic products for monitoring, enforcing, and automating the management of IAM roles.

For Penetration Testers:

Penetration Testers and offensive specialists should consider the cloud attack vector while assessing these applications. In our WQW example, a pentester could easily miss a high-severity vulnerability that is only exploitable with an IAM role in the cloud. This is a good argument for requesting access to cloud-based users with IAM roles to help perform a web application assessment.

During offensive engagements, always consider the following:
- What is the architecture of this application?
- What are all the ways I can interact with the internal components of this application?
- Have I compromised any users with IAM permissions in the cloud?
- Can I acquire new IAM permissions using any exploits I have identified?
- How can I find or extract secrets to internal components of the application?

Overall, for both offense and defence, consider that cloud does not change the fundamental principles of security. The concepts are all the same, but application of these concepts is different in cloud, and so are the consequences of neglecting them. To conclude: it doesn't matter if cloud is someone else's computer – it's your responsibility. ●