Tips for Securing a Modernized Service-Based Architecture

Over the decades, the monolithic, tiered approach to building software has given way to a more **distributed, component-based architecture (https://www.upwork.com/blog/2017/01/upwork-modernization/)** commonly referred to as "microservices," although some components aren't "micro" at all. Separating out an app's services into isolated, interoperable **containers (http://www.upwork.com/hiring/development/what-is-containerization-and-is-it-the-right-solution-for-you/)** has revolutionized the way developers are able to update, add to, or expand parts of an app.

Microservices aren't the perfect fit for every scenario, but for many this approach makes it far easier to build out a complex application from loosely coupled, reusable pieces. Whether an app is being run on your server, or within **Docker (https://www.upwork.com/hiring/for-clients/docker-benefits/)**, or on a cloud-based platform like Amazon Web Services (AWS), creating a distributed environment allows developers to divide and conquer without a lot of dependencies.

That said, while microservices have freed us from many of the constraints of the monolith, these benefits come with increased complexity, vulnerabilities, and risks that need to be mitigated with a tailored security strategy.

MONOLITH VS. MICROSERVICE

If you're **modernizing your app (https://www.upwork.com/blog/2017/05/modernizing-upwork-micro-frontends/)** by making the move to a microservice-based architecture or starting from scratch, here's a look at some key things to consider when it comes to securing your app from attack.

In a world of rapid iteration and frequent deployments, the monolith approach can be much like a Jenga tower: tinkering with parts of the whole may cause the whole structure to topple. When credentials are compromised, it's easier to take down the whole app. On the flipside, it can be easier to secure because authentication is global (only one auth request needs to be made per user request), and there's likely only one type of data store to encrypt.

However, when updates, bug fixes, or failures are being made to isolated components in a microservice-based app, all of a sudden it gets much easier to patch parts and pieces without updating the entire structure. Consider this vs. a traditional, monolithic approach, which is tiered instead of compartmentalized, and securing a microservice-based architecture will look quite different.

Right off the bat, you're able to create a more diverse, app-specific security strategy and it's easy to apply access permissions on a component-by-component basis. To implement that strategy well requires strict API management, visibility into each component, and a clear roadmap of how everything works together—and that's where things get tricky.

> "Establishing and respecting clear component boundaries is an architectural imperative as system complexity grows. HTTP-based services and service contracts constitute clear, bright lines that support the separation of logical subsystems and help engineers maintain and evolve these boundaries consciously."

Stratis Karamanlakis, Chief Technology Officer, Upwork, from ***Upwork Modernization: An Overview*** **(https://www.upwork.com/blog/2017/01/upwork-modernization/)**.

When it comes to securing an application from hacks, knowing how it's built and how it works is paramount to knowing how to properly secure it, but getting a bird's eye view of that many moving parts requires discipline, organization, and foresight.

Paul S.
AWS Certified | Microservices Developer

(/o/profiles/users/_~01ee5c780709ab8de4)

THE MICROSERVICE SECURITY CHALLENGE

No matter how an application is constructed, security concerns typically reside in one bottleneck: the network. For microservices, this gets exacerbated due to the various network connections and APIs used to forge communication channels between all those components. These create more inroads for attack and interception of in transit data.

With a monolith, there are typically one or more servers within a network, which makes it easier to focus on the exposed ports and APIs, identify an IP address, and configure a perimeter around it. With microservices, this gets much more complex with many exposed ports and API gateways—and authentication becomes a distributed matter, too. At the risk of oversimplifying, all this means that running a distributed system means you need to enforce a distributed security approach, and everyone has to be on board to make it a success.

THE MICROSERVICES CULTURE SHIFT: ADOPTING A DEVSECOPS PHILOSOPHY

Like we mentioned above, microservices aren't the best solution for every application, but for those that it offers clear benefits, much of getting over that hurdle from monolith to microservices is a shifting of mindset.

You've likely heard of **DevOps (https://www.upwork.com/hiring/for-clients/what-is-devops/)**; *DevSecOps* is the security-specific philosophy that ensures a component's security doesn't take a back seat to its logic.

Abby Fuller, a developer advocate with AWS, said the following **in a talk with WeaveWorks (https://www.weave.works/blog/introduction-microservices-aws-perspective/)**, a container management company: "To switch from a monolithic network to one comprised of synergistically functioning microservices, a company will need to transform both its culture and its system's architecture. This process will not simply be a matter of atomizing the monolithic system into microservices. Service partnerships will be required to effectively manage the added complexities of testing, deploying and operating multiple microservices."

LEVERAGE THE GOOD; MINIMIZE THE BAD

With your app's components and services isolated, there are some clear security benefits, not the least of which is a **reduced attack surface area**. The best way to think about this is to picture a bank vault. If all of the money was stored in a large vault, a thief only has to crack one safe door to access all of the money. If it's stored in many smaller safes, the thief's job suddenly gets much more difficult. Microservices shrink that attack surface area because you don't have all of your eggs in one basket.

Other advantages:

- **You can be more granular.** If you think of security as coarse-grained or fine-grained, microservices security can be much more fine-grained, applying service-specific rules to containers, APIs, and firewalls.
- **You can be more diverse.** Without all of your eggs in one basket, you can apply different rules and layers to different components, mixing up your tactics.
- **You can tailor access requirements.** Implement a "model of least privilege" on a service-by-service basis to tailor what services can do, e.g. read-only vs. read/write.
- **You can better distribute responsibility.** Each team can distinctly define precisely how their component should be interacted with. This is an advantage on a larger scale because the people who are closest to the code know its nuances better than a firewall team further upstream.

What all of this means for a microservices-based app is that, right off the bat, it's much easier to protect with a "**defense in depth**" strategy—something that's more difficult to accomplish with a monolithic architecture. To really capitalize on these benefits, it's important to adjust the way you think about security for your application—because it's a big shift from the old way of securing things.

Note that a microservices environment can amplify some existing security issues of the monolith, like network communications. Because services are decentralized and separated, how they communicate with one another is no longer internal, **it's happening over a network**. The network security bottleneck therefore gets tricky, and access control gets more complex.

Within a microservices architecture, there are bound to be lots of transactions and interactions. Thus, the security of the app hinges on those container interactions, and knowing what they are and when they happen.

TIPS TO APPROACHING MICROSERVICES SECURITY

The keys to securing a microservices environment? Getting into a DevOps state of mind, securing REST API gateways and permissions, and creating plenty of visibility into each service and the way they communicate with one another. Let's break these down into some actionable tips.

1. Establish visibility into the entire infrastructure by mapping out each microservice and inter-service communication.

Having a clear, **detailed map of each service (https://www.upwork.com/hiring/for-clients/mapping-approaches-for-distributed-cloud-applications/)**, its dependencies, what other services it communicates with, and when and why those interactions happen will not only give you insight into how to best secure them and authenticate their access, it will also be a baseline to compare any aberrations against.

2. Implement "defense in depth."

Layer your network security defenses. System firewalls aren't enough; you'll want to go component-by-component, pinpointing the most sensitive functions and data and locking them down with a "defense in depth" approach.

This can include **building fine-grained firewalls between services**. Fine-grained firewalling of containers and services]creates safe dividers between services, creating and applying sets of firewall rules to each container with host-to-host encryption. Automation becomes a near necessity as this gets incredibly time-consuming and complex with traditional configuration management tools.

3. Rethink team structures.

The very philosophy behind microservices—many parts contributing to a greater whole—should also apply to a team structure supporting the development and security of the app. Security shouldn't be a separate, standalone role from development, especially when there are so many small, moving parts.

Assemble teams whose contributions improve your product, not just a specific project. You want teams to complement one another and be aware of interrelated issues, not be silo-ed. Security isn't always in the forefront for developers who are more (and rightfully) focused on logic, but making this a part of the team mindset can keep it more top of mind. Sitting them beside a security expert, while a simplified analogy, is a great way to ensure it's secure from the ground up. This is something Netflix has done with its Platform Security Team, with great results.

4. Diversify your security tactics.

Your approaches to security can be varied. Think of each component like a locked room in a house. By applying different locking mechanisms to each door, if a thief breaks into one, they haven't figured out how to crack every other door. Avoiding uniformity in your microservices security is helpful.

5. APIs are like communication contracts. Enforce them.

So how do all those components join forces to become an app as a whole? They have to communicate over the network, and it's those communications that present the most complexity (and vulnerability) for a microservices-based app. If your application's services handle sensitive data that require **compliance**, you'll want to be extra strict about how services communicate with one another.

API gateways are typically how microservices communicate with one another, so **protecting those APIs (https://www.upwork.com/hiring/development/tips-for-building-a-safe-secure-api/)** are important. Create rules around these communications so they act like contracts. This means creating very well-built APIs that use features like throttling, authentication, or access tokens from a centralized authorization server.

Protecting the "pipes" should be a priority. AWS customers can use their proprietary **API gateway service (https://aws.amazon.com/api-gateway/)**, which offers the ability to host multiple app versions, safely store API keys, throttle requests to the API, and more.

6. Find an encryption strategy that works for you.

Part of any good defense in depth strategy should include **encryption measures (https://www.upwork.com/hiring/development/introduction-to-encryption-data-security/)** like **HTTPS transport security (https://www.upwork.com/hiring/development/a-quick-easy-guide-to-securing-your-site-with-https/)** to protect data *in transit* and *at rest*. HTTPS encrypted connections with certificate-based security (like **SSL (https://www.upwork.com/hiring/development/ssl-certificates/)**) become especially critical when your data travels externally from the network, and key management should be, too. While it's a great default, it's not a solution for all problems. Depending on the overhead it creates for your app, you might also consider other strategies.

Encrypting *databases* with microservices has its advantages, as well. With a monolith, it's tricky to encrypt on a by-table basis, but with data stores broken up between microservices, it's easier to decide on a component-by-component basis what needs encrypting and to what degree.

7. Code securely from the ground up.

Of course, writing code from day one that's designed to be secure is optimal, and microservices are no different from other software in this respect. However, with microservices there's a decentralized responsibility and accountability for individual components, so each has to be able to stand on its own security-wise.

8. Align with the "Principle of Least Access."

Limit permissions to the bare minimum needed. This is an excellent rule of thumb across all areas of security. Restricting access between microservices is beneficial in the same way restricting access to a human is—the less access, the better.

Like we mentioned before, authentication becomes distributed between components. You can use the **Spring Security framework (https://www.upwork.com/hire/spring-security-freelancers/)** with OAuth 2.0 and OpenID Connect for scalable delegation, access controls, and user identity. For AWS customers, **Amazon's IAM (https://aws.amazon.com/iam/)** allows for user roles and permissions. By only giving services the permissions they need to function, you're making things more difficult for hackers.

9. Automation is your friend!

There's a harsh truth that comes along with the convenience of microservices, and that's how tending to each can be incredibly time-consuming. For deployment, a service like AWS' Lambda can help with automation during the development process, but automation can also be helpful from a security standpoint, as well. For automated security assessments, AWS offers **Amazon Inspector (https://aws.amazon.com/inspector/)** to run audits and assess vulnerabilities and compliance.

10. Make testing a regular part of the build, update, and maintenance program.

The beauty of microservices is how it supports Agile development and the rapid deployment of new services, but this means that things are always changing within an app, so the security approach needs to be just as nimble.

Upwork's CTO, Stratis, also notes "Testing a large distributed system is hard, and continuous delivery makes this harder since the shared runtime environment is changing constantly. We are seeing promising results with contract testing, which is one of the various **microservices testing strategies (http://martinfowler.com/articles/microservice-testing/)** we've considered."

Every time a component is updated, test it and scan it for vulnerabilities. If they're already running in their own containers, this is a bit easier. Or, you could set up a monitoring platform to keep an eye on the system as a whole, picking up any abnormal communications between microservices that might indicate a problem. Either way, the map we mentioned earlier still plays an important role and needs to remain updated. With everything visualized, anomalies will be easier to spot.

Sidenote: Running on AWS?

AWS is packed with security features that you can pick and choose from to suit your needs, covering everything from key management with CloudHSM and SSL certificates to Identity Access Management (IAM), firewalls, and DDoS protection. We won't get into all their specific offerings here, but to see a full list and learn more about each, visit **here (https://aws.amazon.com/products/security/?nc2=h_l3_db)**.

BY CAREY WODEHOUSE (HTTPS://WWW.UPWORK.COM/HIRING/AUTHOR/CAREYWODEHOUSE/) - FREELANCE CONTENT MARKETER AND WRITER

Carey Wodehouse is a freelance content marketer and writer based in Richmond, VA who's worked for clients ranging from online retailers and global market research... more (https://www.upwork.com/hiring/author/careywodehouse/)

RELATED SKILLS

DevOps (/Hire/Devops-Freelancers/)

Microservices (Https://Www.Upwork.Com/Hiring/Tag/Microservices/)

Security (Https://Www.Upwork.Com/Hiring/Tag/Security/)

Previous article (https://www.upwork.com/hiring/mobile/chatbot-trend/) Next article

RELATED ARTICLES

IS THE INTERNET OF THINGS USEFUL FOR YOUR BUSINESS? (HTTPS://WWW.UPWORK.COM/HIRING/FOR-CLIENTS/IS-THE-INTERNET-OF-THINGS-USEFUL-FOR-YOUR-BUSINESS/)

August 16, 2017

Tapping into the world of IoT functionality can help companies work smarter and create products that modern consumers want. Here are some specific ways businesses can leverage the IoT.
Read More (https://www.upwork.com/hiring/for-clients/is-the-internet-of-things-useful-for-your-business/)

TRENDS IN CYBER SECURITY THREATS & HOW TO PREVENT THEM (HTTPS://WWW.UPWORK.COM/HIRING/DEVELOPMENT/TRENDS-IN-CYBER-SECURITY-THREATS-AND-HOW-TO-PREVENT-THEM/)

August 29, 2016

There are a number of ways to help protect your data, systems, and customers' information against security threats. But new types of threats are emerging that can compromise your business. Here's a quick guide to some trends in IT security and a few ideas to safeguard yourself against them.
Read More (https://www.upwork.com/hiring/development/trends-in-cyber-security-threats-and-how-to-prevent-them/)

WHAT IS CONTAINERIZATION AND IS IT THE RIGHT SOLUTION FOR YOU? (HTTPS://WWW.UPWORK.COM/HIRING/DEVELOPMENT/WHAT-IS-CONTAINERIZATION-AND-IS-IT-THE-RIGHT-SOLUTION-FOR-YOU/)

April 10, 2017

In an ongoing effort to streamline infrastructures, maximize server resources, and ensure that applications run smoothly and securely, different approaches have shaken up the traditional server-side architecture—things like the cloud, virtualization, and containerization. Here's a look at how containerization works, how it's supporting the DevOps culture, and some of its benefits so you can decide if it's right for your application.

Read More (https://www.upwork.com/hiring/development/what-is-containerization-and-is-it-the-right-solution-for-you/)

TIPS FOR BUILDING A SAFE AND SECURE API (HTTPS://WWW.UPWORK.COM/HIRING/MOBILE/TIPS-FOR-BUILDING-A-SAFE-SECURE-API/)

November 1, 2016

APIs funnel data between applications, the cloud, and a multitude of different API users, all of whom need to be verified and authorized to access that data. Good API security should be a cornerstone of any API strategy. Here's a look at some of the basics of API security.
Read More (https://www.upwork.com/hiring/mobile/tips-for-building-a-safe-secure-api/)