nordicapis.com

Which Is More Secure: Monolith or Microservices? | Nordic APIs |

Thomas Bush

7-9 minutes

There's a lot to like about microservices: they facilitate scaling, aid in isolating faults, and make it possible to feed an entire development team with just two pizzas. With that said, a greater concern for some is that of security…

Are microservices more or less secure than monoliths? What do you need to consider when switching from one architecture to the other? We'll answer those questions — and others — in this five-part comparison of monolith and microservices security.

Attack Surfaces

A significant security consideration when choosing between a monolith and microservices is the number and size of attack surfaces. An attack surface is the sum of vulnerabilities in an application — the smaller it is, the better. In a monolith, there is only one attack surface, albeit a sizable one. With microservices, however, each service has its own attack surface, but of lesser size.

Here, the winning architecture often depends on the size and complexity of your application. A small, straightforward application has fewer vulnerabilities, so the singular attack surface of a monolith may still be manageable. On the flip side, the weaknesses of a single, multi-faceted application may become overwhelming unless divided, as in a microservices approach.

> “The net effect of this is that each [micro]service has a smaller attack surface: the sum of the points at which a piece of software may be attacked. The smaller attack surface also makes it simpler for security personnel to know where to distribute security patches. A monolithic application executable may end up with many different functions and software components inside of it, making it difficult for security professionals to know exactly where the software that needs to be patched is running.” – Bernard Golden, *4 ways to exploit microservices architecture for better app sec*

Coupling

Another factor to consider when comparing monolith security with microservice security is the level of coupling. The individual parts of a monolith are inherently more connected than the services in a microservices architecture. This means that if one aspect of a monolith is exploited or otherwise fails, others may follow.

On the other hand, microservices are decoupled by design. While this does mean there are more attack surfaces to secure (as discussed previously), the likelihood of a cascading vulnerability across multiple aspects of the application is much lower. Not to mention, decoupling also makes it easier to address individual issues and roll upgrades without shutting down the entire app.

> “In a monolithic architecture, all the code is part of a single product. If one security problem develops, the entire system can be open to attack and can be affected as a result. This makes it very difficult to isolate potential security threats as your entire system may now be compromised and malicious software could spread throughout. Locking down parts of the system where a breach occurred can be quite difficult with this architecture since you do not want to turn off

> the entire system to fix one problem…Separation creates fail-safes that preemptively stop security issues within specific services before they spread to other areas of the system." – Bob Brodie, *Increase Security by Transitioning From Monolith to Microservices Architecture*

Authentication

Authentication is definitely a pertinent topic when it comes to security, especially in the case of microservices. Whereas monoliths have all the necessary resources at hand, microservices must communicate between one another to access resources; an effective authentication scheme ensures that only appropriate services and users can gain access.

> "Authentication between microservices isn't something that teams think about when they're coming from a monolith. You don't need to authenticate yourself when calling a function in code between libraries on a server. Microservices are a totally different situation. It's critical to make sure that each service is able to authenticate not just the server requesting data, but also the context of the logged-in user. In a monolith, this is information that's just stored in memory. On a microservice, that's information that has to be passed with each request." – Eric Boersma, *Top 10 security traps to avoid when migrating from a monolith to microservices*

When implemented correctly, authentication (and authorization) are invaluable assets in a microservices architecture, acting as an additional security check with every resource accessed. Implemented poorly — which is true of many in-house authentication systems — may introduce new security vulnerabilities and a lot of redundancy. To quote API security expert Keith Casey, "if you want to attract hackers, build your own authentication system."

Monitoring

In a perfect world, security is having no product vulnerabilities whatsoever. In reality, though, security is knowing that some threats may be hidden from sight. When hackers exploit vulnerabilities, security professionals must have suitable means to diagnose and respond. As such, monitoring for abnormal behavior is an invaluable process in any security program.

> "Most importantly of all, [hackers] want to act undetected and unmonitored. If they can act slowly — over time — they get the opportunity to do a lot more." – Keith Casey, *How to Build an Effective API Security Strategy*

Monitoring usage is definitely easier with a monolith than with microservices — all the more so if you want to amalgamate usage data from different parts of an application. Of course, monitoring is still very doable for microservices but requires greater investment.

> "In my experience, service monitoring on microservices is skipped for two reasons. The first is that it's tedious. You need to add that monitoring to each service, and that takes time. Secondly, it doesn't feel like that monitoring is needed—the services are so simple in structure that an engineer will "just know" if something's going wrong." – Eric Boersma, *Top 10 security traps to avoid when migrating from a monolith to microservices*

Containerization

A final (indirect) security consideration to bear in mind is that of containerization. Often, microservices are hosted in containerized environments, which may have several security benefits. For instance, there are numerous container monitoring solutions, which can facilitate the otherwise tedious monitoring process described above.

> “The ability to measure and compare observed network behavior to a predictive model for a microservice is a key advantage that containerized architectures provide.” – John Morello, as quoted in *Microservices Security: Probably Not What You Think It Is*

Conclusion

Having reviewed five paramount security considerations in the monolith versus microservices debate — attack surfaces, coupling, authentication, monitoring and containerization — one thing is clear: neither architecture towers above the other. Both microservices and monoliths have pros and cons in terms of security. However, what is clear is that the monolithic approach may make security easier for smaller applications and vice versa.