DZone

How Do You Secure Microservices?

by Tom Smith STAFF · Oct. 08, 17 · DevOps Zone · Interview

To gather insights on the state of microservices today, we spoke with 19 executives who are familiar with the current state of microservices architecture. We asked them, "What kind of security techniques and tools do you find most effective for securing microservices?" Here's what they told us:

APIs, Acess Controls, and Gateways

- Well, **most of our microservices support our internal API**, so they are just not visible and not reachable by external agents. If you break our API, you have HTTP-only access to our microservices: limit what can be done in HTTP, open only HTTP access from your Cloud itself and it is enough for our needs. A rule of thumb we applied to many microservices: no access to DB. Just process JSONs and reach other microservices or API itself again.
- Security domain boundaries – role-based authentication in place. Identification and authentication – **propagate security from the database tier to the application API** – who can access, deploy, scale. Do not lose control over the service.
- **API access control for microservices.** Borrowing open-web API capability. Put together standards for access control in the API architecture – certificates, tokens, networks using Torch, Kubernetes, and CloudFoundry.
- **Need API gateway key or login**. SSL at the transport layer. Whitelist ISPs that can access.
- JSON Web Tokens, Amazon Virtual Private Cloud, and **API Gateway pattern**.
- **API gateways**.
- There are no silver bullets. We take a layered approach with coding standards, peer reviews, automated testing, and third-party pen testing. **More secure profile in public with APIs.**
- **API gateways are the most commonly used solution**, and with good reason – they provide many great out-of-the box management services in addition to security. But when we're talking standalone microservices, OpenID and OAuth is the way to go.
- Design thinking for security with developers and architects. Think about security from the beginning. Best practice is OWAT2 with user authentication. Make sure you are doing regular security scans and that you have no leaks. Docker has security scanning. All talking via APIs. **One single point of entry with API gateways and security platforms.**
- **API security is just as critical as application security.** Not just native first-party apps. That's where ID becomes more critical. Need to integrate to the identity layer- how to onboard, how to authorize access and have end-to-end coverage from the frontend to the backend. More complex patterns from the developer

perspective. Not just front door access security. Zero-trust architecture.

- **APIs are an effective way to build governance right into the microservices architecture.** APIs become these contracts by the developers on what the microservices will do and more importantly what it won't do. This makes it easy for central IT to govern these microservices since their SLAs can be managed through API gateways that act as proxies for the microservices. This ensures that there is the right balance of governance for IT and flexibility for the domain teams.

Other

- **SELinux** is essentially required if you're running microservices. SELinux will prevent a program from running anything on the system outside of its own context. This makes it so that an attacker capable of compromising your container has very few options and often can't do anything on the system. Proper knowledge and use of SELinux for any production workload containers is a must. Apart from that, treat the container and its dependent containers the same way you would your operating system. Update and test regularly.

- Because we focus on application modernization for enterprises with mature ecosystems, we often shift our security model to suit what is in place. In our ideal configuration, we are able to **tie into the existing SSO and authorization schemes to generate a JWT model**. We “double wrap” the encryption of our JWTs to allow services to appropriately access public session data, including insecure APIs outside the enterprise. The inner layer of encryption is readable only to our trusted servers that hold the keys. The JWT with the inner security layer keeps our service layers stateless while opening up interactions across different tiers of trust in the architecture.
- It’s the same as other apps. **Integrate static application security testing into the build process.** This works even better with microservices because it happens more frequently. Process-wise you bring security into the process definition. Turning to dynamic versus static testing. Able to see more risks better once the application is deployed.
- The efficacy of security techniques and tools depend on the environment in which the microservice is deployed into. **An obvious tool for securing endpoints is OAuth.** Most big providers of web service endpoints are relying on OAuth for security.
- **Encrypt all data at rest. Full disc encryption.** Authenticate, authorize security group settings in tooling. We’re behind load balancers on the web. Automate everything. Think through security with regards to the architecture as well as encryption. Use third-parties to come in and audit your practices. Document, audit, and certify your security process to ensure everything is locked down for PCI, ISO, and other compliance requirements.
- **Work with information security, architects, and engineers** for code reviews and static code analysis. InfoSec takes problem reports and watches for patterns and conduct proof of concept attacks.
- **Using a great container scheduler** to manage microservices can help with security.
- As you deliver microservices with smaller team responsibilities you take a supply chain approach to delivery with proper role-based access controls. Lockdown inputs into service delivered so you can control who is putting it out there. **Robust, centralized authentication and authorization policies for access.**

How are you securing the microservices you are working on?

Here's who we spoke to:

- Thomas Butt, CTO, CardCash
- Matt McLarty, Vice President, API Academy, CA Technologies
- Brian Dawson, DevOps Evangelist, CloudBees
- Lucas Vogel, Founder, Endpoint Systems
- Ali Hodroj, V.P. Products and Strategy, GigaSpaces
- Job van der Voort, VP Product, GitLab
- Kevin Sutter, MicroProfile and Java EE Architect, IBM
- Sandeep Singh Kohli, Director of Marketing, MuleSoft
- Karl McGuinness, Senior Director of Identity, Okta
- Ross Smith, Chief Architect, PITSS America
- Mike LaFleur, Director of Solution Architecture, Provenir
- Gianni Fiore, CTO, Rebrandly
- Peter Yared, CTO, Sapho
- Sha Ma, V.P. Software Engineering, SendGrid
- Keshav Vasudevan, Product Marketing Manager, Swagger/SwaggerHub, SmartBear
- Chris McFadden, V.P. Engineering and Operations, SparkPost
- Christian Beedgen, Cofounder and CTO, Sumo Logic
- Todd Millecam, CEO, SWYM Systems, Inc.
- Tim Jarret, Senior Director of Product Marketing, Veracode

Topics: DEVOPS , MICROSERVICES , SECURITY , API SECURITY , API GATEWAY

IN PROGRESS