Top 10 API Security Considerations

By Gunnar Peterson

Gunnar Peterson, Acting Principal of Arctec Group is a well known industry advisor focused on providing guidance — from a technology and vendor-independent perspective — to help improve business results. He blogs (1raindrop.typepad.com) and tweets (@oneraindrop) about distributed systems, security, and software that runs on them.

Introduction

APIs are a certified "cool" technology because they provide data and functionality to developer, users and platforms. But APIs are risky too, because they are the entry point to the enterprise.

API stakeholders include developers, architects and businesses whose goals are to use APIs to drive the next generation of software architecture and create new channels for customer engagement (such as mobile applications). To succeed with APIs over the long haul, architects must define and deploy a security model that integrates with each API's design and usage.

In this paper, we'll explore the top 10 considerations for a security architect who wants to tackle one of today's biggest security architecture challenges.

1. Implement Model-Approach-Controller architecture

Information security is usually very focused on dealing with threats and vulnerabilities, and less aligned with architecture. A core principle of architecture is DRY (Don't Repeat Yourself), which means that systems should be based on design patterns that allow for scalability and manageability.

TO DO

Increase the use of design patterns. One pattern worth considering is the Model-Approach-Controller pattern. The goal of this pattern[1] is to separate concerns based on the system's ability to evaluate a request against access-control policy.

1 http://1raindrop.typepad.com/1_raindrop/2014/09/model-approach-controller.html

The *Model* contains the business logic and data that are the crown jewels in any system; think of this as the back end or data tier. The *Approach* is the front end, or presentation layer, which is not a trustworthy environment because each request that comes in from the client must be verified before granting access to the data model. The system that enforces the policy is the Controller, which is an API gateway that facilitates the validation of the request, executes the access control policy and communicates the request to the back end.

2. Know and contain your assets

"Normally, everything is split up and problems are solved separately. That makes individual problems easy to solve, but the connections between the problems become very complicated, and something simple ends up in a real mess. If you integrate it in the first place, that turns out to be the most simple solution. You have to think ahead and you must always expect the unexpected."

— Jan Benthem, Schipol Airport Chief Architect

The basic mapping for access control is pretty simple. Subjects (like users and clients) request access to objects (like data, applications and services), and access controls mediate the decision to grant or deny access. However, this simple subject-request-object mapping quickly becomes complex when you factor in management considerations.

The first challenge is to identify where the access policy in a distributed system should be enforced. Putting this decision on the server side makes sense, since the server has a better protection profile (the client side is a hacker's dream of easy reverse engineering). However, your security system is put in place to safeguard the object (data, functionality), and defensive in-depth thinking dictates that these access-control decisions should be enforced before the request reaches the resources the system is protecting. This makes an excellent case for integrating a gateway to enforce policy upstream of the assets you are protecting.

The subject side of the access-control equation is usually managed in a structured way. Users are kept in Active Directory, LDAP and/or applications databases. However, resource management is a challenge for most organizations because there is usually no unified directory or repository of all the exposed applications, data and functionality. Additionally, in most enterprises, the object-management regime is not nearly as mature as the subject (user) management programs.

Both subject and object access control are required to establish effective access policy governance. An "integrate first" approach is a good way to define, enforce and manage security policy. An API architecture must also define a chokepoint where APIs are published.

TO DO

Work toward the same degree and granularity of management over APIs as you have achieved for user management.

3. Design for malice

Most security architectures devote the lion's share of attention and focus to access-control services that establish the rules of engagement for authenticating and authorizing users. But what about malicious actors that are focused on defeating the system? They often know the exact protocols of the access-control system and are deliberately trying to bypass it. Or, they are trying to exfiltrate data and other valuables from the enterprise.

The answer here is less about access-control schemes per se, though they do play a role. To design for malice, you need to assume the attacker is not playing by the access-control rules and in fact is working outside the areas governed by policy.

TO DO

Build out design patterns for defensive services and ensure that the system does not naively extend trust. The API must be protected by defensive services that are capable of ensuring a safe distributed computing environment through input validation, content filtering, output encoding and data-sanitization routines. These services should not be by-passable and must be processed on all input and output to and from the API.

4. Monitor for flaws — API attacks *are* happening

"You don't know who's swimming naked until the tide goes out."

— Warren Buffett

Mobile security gets a lot of attention, and rightly so. But look at where the attacks are happening — on the server side. Apple's recent challenges with iCloud are a great example of this. Apple users endured high-profile breaches even though their data was generated on iPhones, stored on the iCloud server and protected by passwords. The attackers were able to crack those passwords by using a brute-force cryptanalytic attack (continuously guessing password possibilities). According to Apple CEO Tim Cook, "Apple could have done more to make people aware of the dangers of hackers trying to target their accounts or the importance of creating stronger and safer passwords."[2]

2 http://online.wsj.com/articles/tim-cook-says-apple-to-add-security-alerts-for-icloud-users-1409880977

Attackers are pragmatic. That shiny piece of metal in your pocket holds valuable data about you, but servers hold the keys to the kingdom. Passwords by themselves are not good enough to protect this data.

TO DO

Do not give attackers a royal road to your users and company data.

- **Log access requests to all APIs**
- **Monitor all access attempts for brute-force and lateral attacks**
- **Employ risk-based access control that adapts to how the application is used**

5. Think mobile and beyond

The mobile computing age is upon us — at this writing, there are more mobile users (1.8 billion) than web browser users. Security in the mobile age is at least as much about server-side API security as it is about securing mobile devices.

Delivering security to a wide proliferation of different kinds of clients is a daunting task. There are tens of thousands of variants to consider just in the Android ecosystem alone.

The real challenge is to define where and how to centralize security policy enforcement. If you have 18,000 different devices, you don't have a controllable system; you have a zoo.

For all the attention that mobile computing gets, its important to take an holistic view. Attackers look for the weakest links and so must security architects. B2B, FTP, third-party integration, and SOAP Web services may not be in the daily tech headlines as much, but they're still in wide use in the enterprise. Attackers are intent on getting access to data by whatever means, so you need to focus on the most valuable assets regardless of how over- or under-hyped the technology platform.

TO DO

Centralize security enforcement for APIs with an API gateway. The gateway should centralize access-control policy for authentication and authorization, including enforcement of OAuth protocols and token management.

Distributed systems are notoriously hard to manage. By centralizing access-control protocols and management with an API gateway, the burden is lifted from the front and back ends.

6. Think of sessions, not just APIs

"Cognitive scientist Gerd Gigerenzer noted something unusual when he took a guided tour through Daimler-Benz Aerospace, maker of the Ariane rocket. A poster tracking the performance of all ninety-four launches of Ariane 4 and 5 showed eight accidents, including launches sixty-three, seventy, and eighty-eight. Curious, Gigerenzer asked his guide what the risk of accident was. The guide replied that the security factor was around 99.6 percent.

When Gigerenzer asked how eight accidents in ninety-four launches could translate into 99.6 certainty, the guide noted that they didn't consider human error in the computation. Rather DASA calculated the security factor based on the design features of the individual rocket parts."

— Summary of "Calculated Risks" by Gerd Gigerenzer from "More Than You Know" by Michael Mauboussin

Signing on to an API is one thing, but what about the second, third and nth call? Initial authentication differs from session authentication in that the latter is usually based on session keys. Session keys must be generated in such a way that they are not easily guessed; in practice, this means long and random identifiers. Session identifiers must be protected both in transit (for example, with TLS/SSL) and at rest, but securing a local sandbox for storing session identifiers is a major challenge today.

In addition, the API session is usually a "midstream" session. Consider a mobile application where a session is running on the client device, another is running between the client and the API gateway, and at least one more is running between the API gateway and the back end. Even in this simple example, at least three sessions fire up when the user presses thumb to glass.

There are substantial security engineering implications here: How will you coordinate session timeouts? How will you synchronize identifiers? What if one times out? Simply getting a consistent user experience is a large-scale task and getting the security model to be consistent on top of that is one of the biggest challenges in Infosec.

TO DO

Ensure that all analysis, design, development and testing uses a synthetic view of the system. Unit tests and component-level views are necessary, but not sufficient. There is no substitute for end-to-end analysis, design, development and testing.

This does not mean there is one single session across all the different interaction points — in fact, that is almost never desirable. It does mean three or more full-session lifecycles (generation, propagation, usage, timeout and reinstantiation) must all work together cohesively and in accordance with security policy.

7. Simplify user experience

Users don't care about the details of identity and security protocols; they just want to use the system. Unfortunately, the security industry has historically placed users at the center of the security protocol, and asked them to make intelligent (and technical) risk-based decisions by answering questions like, "Do you trust this certificate?"

It's not realistic to expect users to know the guts of a PKI in order to browse a site or consume an API from a mobile application. API security should be engineered in such a way that usability and security can find common ground.

TO DO

Ensure the default mode (or only mode) is the highest level of security the system can achieve, and don't ask users to toggle or upgrade security configurations on their own. If there is a case where the user absolutely must make a security decision, explain choices using language free of technical jargon. (Rule of thumb: if it's a term found in a CISSP book, avoid it.)

If possible, educate the user on how best to consume the API in a secure manner. Where user interaction is critical, describe the options by focusing on functionality. Aim for clear, simple, action-oriented language, for example: "This app wants to post tweets on your behalf."

8. Simplify the developer experience

Developers are users, too. APIs can unwittingly create vulnerabilities by not arming them with sufficient knowledge. For example, a developer may hit an API too many times and degrade performance. Or a developer may not know why or how to protect API and session keys. These problems can result in Denial of Service, Privilege Escalation and other security issues.

Security mechanisms, like acquiring, using and storing tokens safely, are not as simple as many other things developers do with APIs. The process of using an API in accordance with your company's security policy must be as simple as possible if you want developers to do it right. This is an end-to-end function, from registering a client, to building code, to testing use cases.

TO DO

Ensure that all APIs have security guidance. This should include sample code for exactly how to do the following:

- **Register application**
- **Acquire API keys**
- **Store API keys**
- **Revoke API keys**
- **Manage API keys**
- **Authenticate to the API**
- **Handle errors gracefully**
- **Test the API client and end-to-end interactions**

Just like you need to simplify the user experience, simplify the developer experience by setting a default API mode that complies with your enterprise security policy. Also remember that APIs may have many different client types, so it's especially beneficial to have SDKs and reference clients for all the different clients your API is likely to encounter, including JSON, Android, iOS, Windows clients and more.

9. Appoint an API curator

This Top 10 list is primarily focused on technology, but even in the age of machines, we humans still play an important role (for at least a few more years, anyway). As much as tech matters, it's important to get the organizational side right, too — just publishing APIs in a haphazard manner leads to chaos. Organizations need a gatekeeper who can ensure that policies and processes have been followed before exposing the API (and all the data and functionality behind it) to the outside world.

Needless to say, this is not a glamorous role, but it's a vital one. Infosec teams have long wanted to get more in synch with development teams, and appointing an API curator responsible for promoting APIs to production is the best way to achieve that goal in the long term. The value proposition is that Infosec can ensure security policies are followed, data is handled appropriately relative to its classification, vulnerability testing is conducted and the overall security architecture is in place.

TO DO

Take advantage of this opportunity to establish solid API publishing practices.. If your company has an open spot for an API curator or API governance body, nominate yourself or your Infosec group for that role. It's a soft-skill kind of role — more process than tech — but it's a role that can put Infosec on the right side of the table (literally) to not just recommend but to make the change necessary to ensure a more secure API deployment for your enterprise.

10. Be bi-directional: Notifications, Websockets, SMS

Access-control paradigms are changing. Client/server communications mainly follow Request-Response models, but with mobile, multi-factor authentication and HTML5, we are starting to see wholly new protocols in use. These include:

- Websockets where data is sent back and forth in full-duplex mode
- Mobile push notifications where the sever initiates communication to the client (in effect inverting the roles so the client is the "server")
- SMS-based authentication where servers send text messages with credentials for log in

TO DO

Ensure that API Gateways can manage both inbound and outbound communications. The APIs lot in life is a glue or integration layer. It will increasingly have to speak a wide spectrum of different protocols and step into non-traditional roles where servers initiate communications.

Because API security is so important, we are taking this top ten list to eleven.

11. Focus on the data

"I very frequently get the question: 'What's going to change in the next 10 years?' And that is a very interesting question; it's a very common one. I almost never get the question: 'What's not going to change in the next 10 years?' And I submit to you that that second question is actually the more important of the two — because you can build a business strategy around the things that are stable in time. …[I]n our retail business, we know that customers want low prices, and I know that's going to be true 10 years from now. They want fast delivery; they want vast selection. It's impossible to imagine a future 10 years from now where a customer comes up and says, 'Jeff I love Amazon; I just wish the prices were a little higher,' [or] 'I love Amazon; I just wish you'd deliver a little more slowly.' Impossible. And so the effort we put into those things, spinning those things up, we know the energy we put into it today will still be paying off dividends for our customers 10 years from now. When you have something that you know is true, even over the long term, you can afford to put a lot of energy into it."

— Jeff Bezos

The parallels to Infosec are pretty clear. It's a faddish industry, but there is persistent common denominator — not APT, worms or malware, but rather the simple fact that we must protect user and company data. Attackers want it, individuals and businesses need it. It's the lifeblood. That is not going to change, and focusing on data must be a central tenet of any security architecture program.

Protecting data requires a lot of attention to detail and a long-term focus. Data classification is critical to ensuring that secrets and confidential data are not breached. The key here is to make sure you invest your time and resources in protecting your company's most valuable assets.

Data classification, message data security, data integrity, data confidentiality and data validation are not new concepts in information security, but they do require some investment to fully deploy. Whether the technology is FTP or web or mobile API, data security remains the cornerstone of information security.

Conclusion

APIs represent both threats and opportunities for Infosec teams. The threats are obvious — your company and user data is going to be more vulnerable to attack unless you take all of the necessary precautions. The opportunities are to:

- Build better deployment and API curation teams and processes
- Upgrade the access-control architecture
- Ensure that assets are contained and monitored

APIs are coming to your enterprise and they will not and cannot protect themselves. This is the time for security architects to step up and play a leading role in system design.

For more information, visit www.axway.com