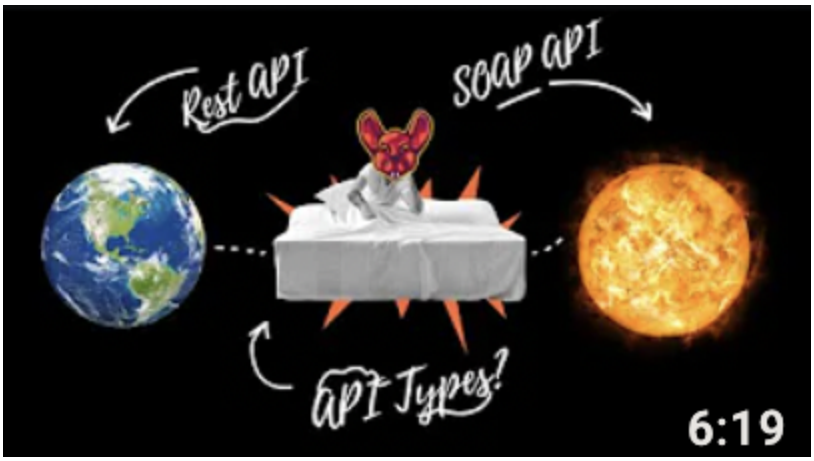
API0.2019: What is an API

Hacking APIs: Types and Architectures

Understanding APIs and their weaknesses



https://www.youtube.com/watch?v=eiZ_hW6ERWM&ab_channel=TheXSSrat

Application Programming Interfaces (APIs) are quickly becoming a point of concern amongst cybersecurity professionals. As organizations continue to expand the interoperability and coordination with each other and as companies make the move to the cloud, API's are playing a greater role in how the modern internet functions.

As cybersecurity professionals, it's important to intimately understand how attackers can target APIs, both from a red team and bug bounty perspective. As we take a look at how to target APIs, first, it's important to know and understand what APIs are and the differences between their types and architectures.

API Overview

APIs allow one application to communicate with another without having to have direct interoperability mechanisms built within. Simply put, using an API, the user of application A can communicate and use the data of application B. This gives the opportunity for a company to leverage the resources of a third-party partner, to leverage their data, or their technology.

As you can probably tell by now, there is a lot riding on APIs. This is why API security is such a big deal.

API Architecture

There are two main categories of API architecture that we want to look into, the first being one of the more commonly used forms of API, REST. REST is short for Representational State Transfer and is lighter and more flexible than the other main API, SOAP. We'll talk about SOAP in a moment, but first, let's discuss REST.

REST is an architecture that uses many different methods, including HTTP and JSON. It can also use SSL as a security mechanism, as well as SOAP. Being that it's built on the client-server principle, it is lighter and easier to use across the web. REST APIs are stateless and thus are also less complex. Being stateless means that any server can handle the API request.

While we're on the topic of REST APIs, let's pause to consider the security considerations.

REST APIs must be uniform, which means all the requests must appear the same, so the servers handling the requests can properly handle them and none are dropped. This facilitates the "availability" edge of the CIA triad (Confidentiality, Integrity, Availability).

REST APIs must also be cacheable, meaning you must have a method of retrieving previous API calls for investigative purposes. This aids in non-repudiation.

Finally, REST APIs must follow a layered system approach. Consider a network as layered, with more security encompassing the most sensitive systems in the middle, and the least sensitive and security on the outer ring. An API being used in the outer layer should not be able to access systems on the inner layer, except via a highly protected and monitored channel. That said, this will probably not be done using a REST API, and instead, will use a SOAP API. Let's dig into that.

A SOAP API is actually not an architecture, it is a protocol that builds on REST API. SOAP stands for Simple Object Access Protocol, and it uses more strict security standards to ensure the integrity and confidentiality of its contents over a REST API. SOAP uses SSL, however, it requires additional data, making it bulkier when traveling and, thus, it is less used than the REST API. It is, however, used often in sensitive environments such as banking or situations where sensitive information changes must be accessed across many resources.

API Types

Now that we understand the two main APIs, let's talk about types of API. The first being open APIs. These are also known as Public APIs. Open APIs do not require authentication to access the API or its resources. Rather, it is open to the public for anyone to use.

Next, there are Internal APIs. Internal APIs require authentication and are intended to be used by a specific user base. For example, a bank would use a series of internal APIs (likely SOAP) to allow interconnectivity across its internal infrastructure.

Third, we have Partner APIs. Partner APIs are used in the case of an organization extending its resources to a third party. This may be a result of a business agreement (think SaaS, etc.). This is not to be confused with an internal API, because internal APIs are strictly in-house. Whereas a Partner API is strictly between a client and a provider.

Finally, there are Composite APIs, which are a mix of different APIs that help make the handling of APIs more efficient. Composite APIs are considered to be an architectural solution to some efficiency problems posed by REST APIs.

Now that we understand the main architectures and types of APIs, we will begin to explore API security and, more specifically, how to attack APIs. Keep an eye out for TheXSSRat's next video covering this topic on his YouTube channel.