FOREWORD

Imagine if sending money to a friend required more than opening an app and making a few clicks. Or if monitoring your daily steps, exercise data, and nutrition information meant checking three separate applications. Or if comparing airfares involved manually visiting each airline's website.

Of course, it's not hard to imagine this world: we lived in it not too long ago. But APIs have changed all that. They are the glue that has enabled collaboration across companies and transformed how enterprises build and run applications. Indeed, APIs have become so pervasive that an Akamai report from October 2018 found that API calls accounted for an astounding 83 percent of all web traffic.

But as with most things on the internet, if there's something good, cybercriminals will take notice. To these criminals, APIs are highly fertile and profitable ground, and for good reason. These services offer two highly desirable traits: (1) rich sources of sensitive information and (2) frequent security gaps.

Consider the role APIs play in a typical application architecture. When you check your bank balance on a mobile app, an API behind the scenes requests that information and sends it to the app. Likewise, when you apply for a loan, an API allows

the bank to request your credit history. APIs sit in a critical position between users and the sensitive systems on the backend. If a cybercriminal can compro-

While APIs have reached an unprecedented level of adoption, their security continues to lag. I recently spoke with the chief information security officer of a 100-year-old energy company and was surprised to learn they use APIs throughout the organization. But, he quickly pointed out, "whenever we look under the hood, we find they are often over-permissioned."

This isn't very surprising. Developers live under constant pressure to fix bugs, push new releases to consumers, and add functionality to their services. Rather than scheduling releases every few months, they must cycle through nightly builds and daily commits. There literally isn't enough time to consider the security implications of every change they make, and so undiscovered vulnerabilities weasel their way into products.

Unfortunately, lax API security practices too often result in unexpected outcomes. Take the US Postal Service (USPS). The agency published an API called Informed Visibility that allowed organizations and users to track packages. Appropriately, the API required users to validate their identity and authenticate in order to access any information via the API. However, once authenticated, a user could look up the account information of any other user, exposing the information of 60 million users.

Peloton, the fitness company, also powers its apps (and even its equipment) with APIs. But because one of its APIs required no authentication to issue a call and get

responses from the Peloton server, it allowed the requester to look up the account information of any other Peloton device (of which there are four million) and access potentially sensitive user information. Even US president Joe Biden, a well-known Peloton user, had his information exposed by this unsecured endpoint.

Here's a third example: the electronic payment firm Venmo relies on APIs to power its applications and connect to financial institutions. One of its APIs served a marketing function by showing recent, anonymized transactions. While user interfaces took care of stripping out any sensitive information, the API would return all transaction details when called directly. Malicious users harvested some 200 million transactions via this API.

Incidents like these have become so commonplace that the analyst firm Gartner has predicted that API breaches will become the "most frequent attack vector" by 2022, and IBM has reported that two-thirds of cloud breaches are the result of API misconfigurations. The breaches also highlight the need for new approaches to securing APIs. The application security solutions of the past focus only on the most common attack types and vulnerabilities. For example, automated scanners search the Common Vulnerabilities and Exposures (CVE) database for flaws in IT systems, and web application firewalls monitor traffic in real time to block malicious requests containing known flaws. These tools are well suited to detecting traditional threats, but they fail to address the core security challenges faced by APIs.

The problem is that API vulnerabilities are not common. Not only do they vary highly from one API to another, but they also tend to differ from those found in

traditional applications. The breach at USPS wasn't a security misconfiguration; it was a business logic flaw. That is, the application logic contained an unintended loophole that permitted an authenticated, valid user to access data belonging to another user. This type of flaw, known as broken object level authorization, is the result of application logic that fails to control what an authorized user is able to access.

Put more succinctly, these unique API logic flaws are effectively zero-day vulnerabilities, each of which belongs only to a specific API. Because of the scope of these threats, a book like this one is crucial to educating penetration testers and bug bounty hunters interested in keeping APIs secure. Additionally, as security shifts "left" to the engineering and development processes, API security is no longer strictly the domain of companies' information security departments. This book can be a guide to any modern engineering team that conducts security testing alongside functional and unit testing.

When done properly, API security testing programs are continuous and comprehensive. Tests conducted once or twice a year won't keep up with the pace of new releases. Instead, testing should become part of the development cycle, such that every release gets vetted before moving to production, and cover the API's entire footprint. Finding API vulnerabilities takes new skills, new tools, and new approaches. The world needs *Hacking APIs* now more than ever.

Dan Barahona
Chief Strategy Officer, APIsec.ai Inc.

San Francisco, CA