Volume 7, Issue 4
[State of the Internet]

API: The Attack Surface That Connects Us All

Table of Contents

Letter from the Editor

Welcome to Akamai's State of the Internet / Security (SOTI) report, Volume 7, Issue 4. Whether you've been reading this report from the beginning or are a new reader, we welcome you and hope our research provides you with intelligence you don't see elsewhere.

Did you know the first instance of an application programming interface (API) in the wild was created by Salesforce.com on February 7, 2000, according to the API Evangelist blog? What was originally meant as a relatively simple system-to-system communication method has evolved into one of the biggest drivers of internet traffic.

We reached out to Chris Eng, Chief Research Officer at Veracode, to ask him to discuss the growth of APIs and the vulnerabilities they expose organizations to. Chris started in security around the time the first API was written, and because of Veracode's role in the application security space, he has a deep understanding of the topic. Unsurprisingly, Chris sees significant parallels between the early days of software development and the current state of API traffic.

We believe attacks on APIs are underdetected — and underreported when they are detected — making them one of the biggest threats organizations face. DDoS attacks and ransomware are both major issues, and they're both in the news today because their impact is so immediate and visible. The attacks on APIs don't receive the same level of attention, in large part because criminals use APIs in ways that lack the splash of a well-executed ransomware attack.

Our drive to evolve the SOTI report has always been one of our key strengths. We want to continue to push ourselves to bring to the forefront new research and interesting ideas that haven't been seen before. We plan on using data sources from across a broad spectrum, including the intelligence gathered by partners like Veracode. We look forward to learning more about what's really happening in the darkness surrounding APIs and sharing the intelligence with you.

Martin McKeay
Editorial Director

GUEST ESSAY

API Security: The Past Is Repeating Itself

Chris Eng
Chief Research Officer, Veracode

The internet itself is remarkably robust and self-healing in the face of random disruptions, but the closer you get to the application layer and especially the human layer, the more you wonder, how did this thing ever work in the first place?

High-profile cyber attacks have become more common and far-reaching, particularly ransomware. This shouldn't bother you though, unless you rely on silly things like gasoline, beef, air travel, or your data backups.

This report focuses on API security, and if you've spent any time looking at APIs, you already know that security is too often an afterthought.

The first rule of writing secure software is don't make assumptions about how people will interact with the finished product. When I started in this industry over two decades ago, as a penetration tester, pretty much every website was trivially hackable due to bad assumptions, e.g., "it's impossible to change the value of a dropdown menu," "client-side validation works," and my favorite, "nobody would ever do that." Web developers were already so many abstraction layers away from the underlying technology that most didn't even understand the HTTP protocol – and the same is probably true, or even worse, today.

Over time, web applications have slowly improved with the emergence of more sophisticated testing tools and more robust SDLCs, but with APIs we're seeing the same patterns all over again. As an example, a list of API-related incidents in 2020 has been collected by CloudVector's Lebin Cheng. While not an all-encompassing list, it serves as a direct example of the repeated patterns observed in the early days of application development and testing.

APIs are often hidden within mobile apps, leading to the belief that they are immune to manipulation. Developers make the dangerous assumption that users will only interact with the APIs via the mobile user interface (UI).

Compare the OWASP Top 10 to the OWASP API Security Top 10. The latter purports to address the "unique vulnerabilities and security risks" of APIs, but look closely and you'll see all of the same web vulnerabilities, in a slightly different order, described with slightly different words. To add more fuel to the fire, API calls are easier and faster to automate (by design!) – a double-edged sword that benefits developers as well as attackers.

We're making all the same mistakes with API security that we made with web security 20 years ago.

The security-development API (i.e., the people)

Thinking about APIs, which allow software components to interface with each other, leads to thinking about the interface between security and development teams.

Security and development have never really spoken the same language, in part because those personas have very different experiences, vocabulary, and priorities. However, that strained relationship is becoming ever more critical, especially with the inevitable push to deliver more features, release faster, and do anything under the "DevOps" umbrella, so as to embrace the latest engineering craze. What needs to change to get better aligned?

Let's sidebar for a moment. In my experience, attackers make the best defenders. This is not to suggest you recruit your infosec staff from PLA Unit 61398 or Lazarus Group. My definition of an attacker is someone with an offense-oriented mindset and expertise — a "breaker."

Breakers have a much better grasp on the art of the possible. Often a breaker will report a vulnerability, and it'll get patched just enough to stymie the proof-of-concept exploit but nothing more. Flip one character from lowercase to uppercase, or try the same attack on a different part of the website, and it works again. Why? The defender didn't do a thorough fix. Why not? Because they don't really know why the attack works and how the attacker thinks. It's not as simple as telling them to "just think like an attacker" if they haven't done it before; that's like telling a hairstylist to "think like a plumber" and expecting that they magically have the know-how to reroute a sewer line.

People who haven't been breakers also tend to mischaracterize risk, usually in the "Chicken Little" direction. The exploits are sometimes black magic to them, because they've never done it themselves, so they err on the side of thinking everything is a gaping security hole. This is ultimately unproductive, because if everything is a priority, nothing is a priority.

On the other hand, pure breakers tend not to understand how development processes work. They oversimplify the cost and complexity associated with code changes, and they often just don't speak the developer's language. They may have never had to build or ship a product, and are unable to grasp the complexity of business trade-offs. And sometimes they don't want to — breaking into things is a lot more fun.

This is where we are today. Very few security professionals can straddle both worlds effectively. They're the proverbial purple squirrels — perfect for the job, but pretty hard to find.

What now?

In product management circles, you'll sometimes hear about the Time to Value (or Time To First Value, TTFV), which describes how long it takes for a new customer or prospect to first realize value from whatever it was that you've sold them. Naturally, the goal is "as fast as possible."

We need to minimize Time to Value for a new security practitioner.

Referring to the recent scale and impact of breaches, Jeremiah Grossman posited that it's not so much that offensive techniques have improved, but rather that adversaries have proven more capable than the infosec industry at recruiting and training entry-level roles.

The cybersecurity skills gap we hear so much about is largely a reluctance (or worse, a systemic inability) to train people up. What if we put more effort toward on-the-job training? For example, learn offense and defense skills simultaneously by exploiting that SQL injection vulnerability, then writing the code to fix it. And then figure out how to circumvent your own fix. Now, repeat the exercise on a crusty 10-year-old enterprise application that needs to pass all the unit and integration tests before the fix can be shipped.

The other strategy we can take is to recruit developers into security! Find an opportunity to demonstrate some exploits and look for the developers with lightbulbs illuminating over their heads. Harness the curiosity and build from there. Imagine a security team that is composed of seasoned breakers working alongside former developers. You get deep experience in both worlds, and most importantly, you begin to break down silos and work collaboratively instead of shouting from the sidelines.

Disclaimer: The views and opinions expressed in this essay are those of the author, and are not reflective of the views and opinions of Akamai.

Introduction

In this edition of the State of the Internet / Security, we're going to talk about the security of application programming interfaces (APIs). It's an important topic – Gartner predicted that "by 2022, API abuses will move from an infrequent to the most-frequent attack vector, resulting in data breaches for enterprise web applications."[1]

In addition to our own research, we've partnered with Veracode for this report, as their insights into the application security space helped further our research into APIs and development challenges. As you've seen, Veracode's Chief Research Officer, Chris Eng, authored our guest essay.

Concerning the state of attacks online, we looked at 18 months of attack traffic between January 2020 and June 2021. In June 2021, on a single day, Akamai observed 113.8 million attacks. That's more than three times the number of attacks that we saw during the same time frame in 2020. With 6.2 billion attempts on record, SQL Injection (SQLi) remains at the top of the web attack trending list, followed by Local File Inclusion (LFI) with 3.3 billion, and Cross-Site Scripting (XSS) with 1.019 billion.

Credential stuffing has been the source of a steady stream of attacks so far this year, with both dips and peaks in the first two quarters. Akamai observed two notable peaks in January and May 2021, when the number of credential stuffing attacks surged past 1 billion. Coincidentally, the two peaks were recorded on the second of each month; there is no indication as to why criminals were hitting their hardest on this particular day of the month.

The United States was the top target for web application attacks during this observed period, with nearly six times the amount of traffic than the United Kingdom, which ranked second. The United States was also in the top spot on the source list for attacks, taking first place away from Russia, with almost four times the amount of traffic.

Finally, DDoS traffic has remained consistent in 2021 so far, with peaks recorded earlier in Q1 2021. In January, Akamai recorded 190 DDoS events in a single day, followed by 183 on a single day in March.

API Security

Interfacing with the world

APIs are everywhere. If there is an application or service available on the internet, you can be sure it's supported, in some way, by an API. These days, APIs power mobile applications, the Internet of Things (IoT), cloud-based customer services, internal applications, partner applications, and more.

In addition to security concerns posed by APIs, there is also the performance aspect to consider. For Akamai's part, we see the performance improvements offered by APIs on a regular basis, as API traffic is offloaded from origin servers to edge servers on the CDN side of things. This configuration speeds up access and ensures availability.

But there's a growing problem. Organizations that defend their APIs with traditional network security solutions are having moderate success at best, if they have any success at all. This is because the old standards of network defense can only do so much. For the most part, the same risks that exist for websites and web applications will apply to APIs, but they need to be addressed separately.

APIs greatly expand the attack surface that organizations must be concerned about. That means defenders and development shops need to work harder to address these problem areas. According to Gartner, "by 2021, 90% of web-enabled applications will have more surface area for attack in the form of exposed APIs rather than the UI, up from 40% in 2019."[1]

The good news is that business leaders and security teams are already adopting stronger security postures related to API practices. However, there is plenty of room to grow, and criminals are certainly taking advantage of the API security gaps.

Gartner predicted that "by 2022, API abuses will move from an infrequent to the most-frequent attack vector, resulting in data breaches for enterprise web applications."[1]

Defending the code

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. They're widely known for their OWASP Top 10 list, which highlights the most critical security risks faced by web applications, including injection attacks, broken authentication, sensitive data exposure, broken access controls, and misconfigurations.

The latest incarnation of the OWASP Top 10, at the time this report was written, was released in 2017 (A1-10:2017). OWASP has also published an API Security Top 10 list (API1-10:2019), and the overlap between the two is substantial. Over time, the security industry has developed a number of ways to track software vulnerabilities and map them to best practice guides like the OWASP lists — including Common Weakness Enumeration (CWE) definitions, which are maintained by MITRE.

Considering the number of publicly disclosed vulnerabilities and reported attacks, it is clear that APIs are facing the same sorts of problems that web-based applications have been tackling for years.

For example, let's look at hard-coded credentials.

On July 29, 2020, Cisco released a patch for Data Center Network Manager (DCNM), after it was determined that hard-coded credentials in the REST API could allow an unauthenticated remote attacker the ability to bypass authentication and execute commands with administrative privileges. This vulnerability, CWE-798 (hard-coded credentials), can be directly linked to both API2:2019 and A2:2017 via OWASP, under broken authentication.

APIs are supposed to be versatile, enabling ease of use and access for both the business and end user. Most organizations use APIs in some fashion, either internally or externally, for customers or business partners, or a mix. The key function and expectation for API development and deployment is integration and data access, but there has been tremendous growth over the past few years in using APIs for digital business lines and services (e.g., Checkr, Twilio, Scale, Segment).

This versatility, however, is where things start to go off the rails, because sometimes the trade-off between ease of use and security is a tricky one to manage.

A good example of how hard it is to balance usability and security is Twitter's API security disclosure from February 2020. In a public notification, Twitter disclosed that a large number of fake accounts were exploiting its API and matching usernames to phone numbers. The API function was supposed to make it easier for users to find friends, but malicious actors exploited this feature for data enrichment. This incident can be tracked under CWE-284, and A2:2017/API5:2019 via OWASP.

Another example of where the trade-off between availability and security can be hard to manage was disclosed earlier this year (July 2021) by researcher Muhammad Sholikhin. He was able to identify members of closed groups on Facebook, using the social media giant's API. A person's membership to a closed group is supposed to be confidential, so Facebook patched the flaw and paid the researcher a bounty for his work.

GitLab, a popular Git repository manager, had problems similar to those experienced by Facebook. It was notified via its bug bounty program that researcher Riccardo Padovani discovered it was possible to view private group projects via the API. Ultimately, GitLab paid a bounty for the disclosure and patched the issue.

Criminals seeking Twilio access

Criminals pay attention to security recommendations and suggested best practices, and look for ways to exploit businesses and services that follow these suggestions, but have made seemingly minor mistakes (albeit with costly repercussions).

One such service is Twilio. Twilio is a hugely successful API service that enables developers to improve user experience by managing communications via SMS, chat, video, and email.

To secure the credentials needed for Twilio to function, the service tells developers to store the Twilio account security identifier (SID) and authentication (auth) token in a way that prevents unauthorized access. The company's documentation stresses this protection, and suggests environment variables as a means to accomplish it. Variables are a logical solution commonly associated with API development. However, where these variables are stored is important.

Twilio says SID and auth tokens can be stored in .env files, but stresses that these files should be added to the .gitignore file so the sensitive content isn't uploaded in plain text. Website administrators will often encourage that such files be stored outside of the main root directory (/www/ or /public_html/) to prevent access. But this doesn't always happen, and it can be a costly mistake.

Criminals are actively seeking access to Twilio SID and auth tokens, by scanning websites for common environmental variable names, such as those that end in .env, .dev, or even .prod. Akamai's honeypots have observed scans for "twilio.env" files directly as recently as August 25, 2021 (around the time this report was being finalized). Moreover, just prior to the scans hitting our honeypots, SANS reported seeing similar scans for Twilio access.

Those doing all the scanning to expose the SID and auth tokens can then take their successful results and do passive reconnaissance on the compromised accounts to sell them to buyers, such as the person in Figure 1.

Compromised Twilio accounts can be used for general spamming, passive phishing, targeted phishing, and other fraud, so it is important that those accounts be protected.

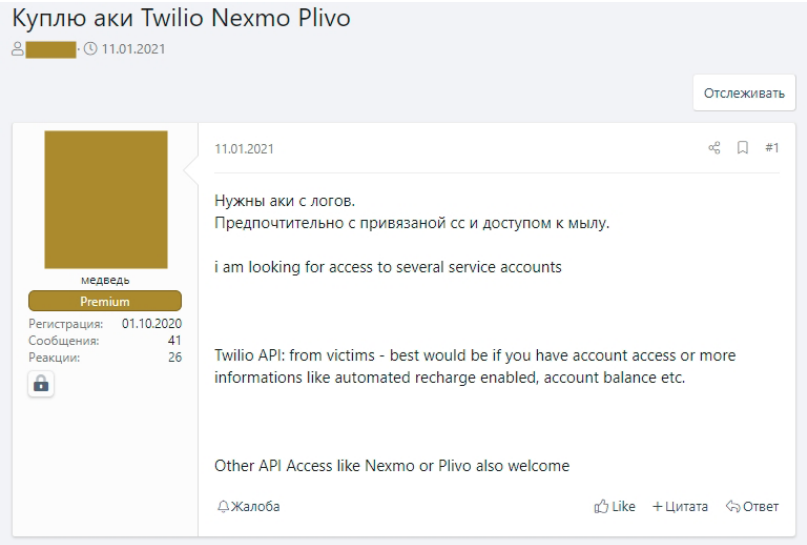

Fig. 1: A threat actor is looking to purchase Twilio access, and is particularly interested in accounts with automatic recharge enabled

Healthcare

As we said earlier, integration and data access are the key expectations for the development and deployment of APIs by an organization. Nowhere is this line of thinking more evident than in the healthcare industry.

In 2020, as the COVID-19 pandemic started to alter our lives, the ability to use mobile health (mHealth) applications – to manage prescriptions, medical records, and even doctor appointments – became crucial. In addition, people continued to use health trackers as a means of managing their diet and exercise.

In February 2021, Alissa Knight, partner at Knight Ink, published a report sponsored by Approov (an API security vendor) on mHealth application and API security. The results of this six-month investigation were astounding.

In exchange for not attributing the findings to any single company, Knight was able to openly test 30 different mHealth applications. Some of the organizations involved with Knight's research had well over 1,000 APIs serving their applications. One of the challenges with API security is identifying and tracking API deployment, and the larger that deployment base is, the harder it becomes to defend it.

Half of the APIs tested in Knight's work allowed access to pathology, X-rays, and clinical results for other patients, as well as access to admission records for patients heading into a hospital, which was outside the level of authorization granted.

All of the mHealth applications tested had APIs that were vulnerable to broken object level authorization (BOLA). This enabled Knight to view personally identifiable information (PII) and protected healthcare information (PHI) for individuals who were not assigned to the tested clinician account. Furthermore, nearly 80% of the applications tested had hard-coded API keys (including some that never expire), tokens, private keys, and even hard-coded usernames and passwords as part of their design.

"Basic cybersecurity hygiene, such as not hard coding usernames and passwords in source code and authorizing all requests, is an endemic problem in mHealth," Knight's report concluded.

"mHealth companies need to implement more of a zero-trust approach to the security of their apps and APIs, ensuring that just because someone is authenticated doesn't necessarily mean they are authorized to access the data."

In short, the mHealth applications tested by Knight were in serious need of a security checkup. Thanks to the cooperation between Knight and the companies maintaining the applications, that is exactly what they got.

Spring Boot

The Java Spring Framework enables organizations to develop enterprise-level applications that run on the Java Virtual Machine (JVM). Java Spring Boot (Spring Boot) makes developing the applications and services with Spring Framework faster and easier. By their very nature, Spring Boot applications will have an API or interact with one.

For this report, Veracode shared information with Akamai regarding the security posture of 5,000 Spring Boot applications over time. The results were derived from static and dynamic application security testing, as well as manual analysis. Overall, 100% of the applications tested had at least one vulnerability. While not all vulnerabilities are equal, the point is, writing code without vulnerabilities isn't as easy as it sounds. It takes considerable time and resources, as well as leadership support for such efforts.

Mapping CWEs to OWASP

If you'd like to map CWEs to the OWASP Top 10, there is a mapping graph available on the CWE website.

The following mappings are related to the Spring Boot findings by Veracode:

CWE 73:
A5:2017 via OWASP

CWE 80:
A7:2017 via OWASP

CWE 89:
API8:2019 and A1:2017 via OWASP

CWE 117:
API8:2019 and A1:2017 via OWASP

CWE 209:
API7:2019 / A6:2017 via OWASP

CWE 259:
API2:2019 and A2:2017 via OWASP

CWE 327:
A3:2017 via OWASP

The majority of the Spring Boot applications tested (86%) were vulnerable to Carriage Return and Line Feed (CRLF) Injection (CWE 117) in logs. There are many types of CRLF vulnerabilities, but for this instance we mean those in which the software in question does not neutralize, or incorrectly neutralizes, any output that gets written to logs. In cases such as this, an attacker could forge log data, or inject malicious content into the logs themselves. CRLF flaws can be exploited to carry out other attacks, including XSS. In a related note, 42% of the Spring Boot applications included in this data set were vulnerable to basic XSS (CWE 80). To be clear though, when testing, Veracode will also flag XSS vulnerabilities under CWE 79, 83, or 86, depending on the situation.

There were also various code-related problems, including 68% of the tested applications that incorrectly released resources before they were made available for reuse (CWE 404), and 50% that used broken or risky cryptographic algorithms (CWE 327).

Hard-coded passwords were an issue as well in 47% of the tested Spring Boot applications (CWE 259), as was error messaging handling, where 44% of the applications included sensitive information in error messages (CWE 209). Some of the Spring Boot applications (41%) allowed user input to control or influence paths used in file system operations (CWE 73), providing a level of control over the application that wasn't necessarily intended. Finally, 31% of the tested applications were vulnerable to XML External Entities (XXE) flaws. XXE flaws can be used to extract data, execute remote requests from another server, or trigger denial-of-service attacks.

Outside of the top 10 problems identified in the Spring Boot applications, 21% were vulnerable to CWE 89, or SQLi (Veracode will also flag SQLi under CWE 564 and 943). The same percentage also contained flaws falling under CWE 601, which addresses open redirect vulnerabilities.

Why application and API vulnerabilities exist

Considering the mHealth and the Spring Boot data, as well as the other examples previously mentioned, it's understandable to question how APIs are having the same problems that web applications did in the past. As it turns out, some of those apps have knowingly been left vulnerable. The Enterprise Strategy Group (ESG) conducted a survey on behalf of Veracode last year, which revealed that organizations are knowingly pushing vulnerable code.

Specifically, 48% of the organizations that participated in the survey admitted to pushing vulnerable code regularly. Among that group, 54% said the vulnerable code is pushed out in order to "meet a critical deadline" with a plan to remediate the known flaws in a later release. The call to deploy is often made by a team (development manager/security analyst – 28%), an individual development manager (24%), a security analyst (21%), or individual developers who assess the priority of each issue discovered (15%).

The other listed reasons for pushing vulnerable code included a feeling that the vulnerabilities were low risk (49%) and that the vulnerabilities were discovered too late in the development cycle to resolve them before the deadline (45%).

Each reason listed is a classic example of a security trade-off. The risk of exploitation was accepted to meet the needs of the business. It might not sound pretty, or even clean, but it's reality.

Security is hard enough as it is, but development security is a complex layer of choices that usually revolves around supporting the business first. If a business needs to go to market with a product, and a code library that was used in development is discovered to be vulnerable just as the product is set to go live, for many business leaders it is a reasonable request to launch now and work on a patch as soon as possible – provided that the vulnerability isn't a critical one that exposes the business or its customers/users.

Low-risk vulnerabilities, such as those that require a complex series of steps, a certain level of access, or a specific set of circumstances to exploit, could be deprioritized so that the product can hit release schedules. It's a risk decision that businesses all over the world deal with on a daily basis.

Push first, patch second doesn't mean security is ignored completely. It's just triaged in a way that prevents it from interfering with the business or the user experience. This is why it is important for security to be part of the development cycle.

Collectively, a majority of the respondents to the ESG survey (78%) said that security analysts were directly engaged with their developers. This engagement happens by working with developers to review features and code (31%), doing threat modeling (28%), or participating in daily development meetings (19%).

Another element of the security problems found in APIs and application development is the dependency on open-source code. ESG's report states that while modern codebases are heavily dependent on open-source code, fewer than half of those surveyed (48%) said they were leveraging tools to monitor the health of those open-source projects.

Finally, the developers themselves, who are driven to produce code on a deadline and follow a *push first, patch second* mentality, are lacking in training. ESG reported that 29% of those surveyed stated that developers lack the knowledge needed to mitigate the flaws that were identified in their code. In fact, 53% of the respondents stated that training is only ever done when a new developer joins the team, as part of an annual training program, or that their developers are expected to educate themselves.

Akamai by the Numbers

As mentioned, the attack statistics in this report covers 18 months, from January 2020 until June 2021.

Web attacks

Akamai observed 113.8 million attacks on a single day in June 2021, which is more than three times the number of attacks observed in June 2020 (Figure 2).

SQLi stands out, in Figure 3, as the number one attack profile over the past 18 months, with 6.2 billion attempts recorded. Coming in a somewhat distant second is LFI with 3.3 billion attacks, and finally XSS with more than 1 billion attacks.

Given the fact that a majority of traffic online these days is API-based, the web attacks observed by Akamai are almost certainly targeting organizations with public-facing applications and services.

In fact, when Akamai looked at the data shared by Veracode, many of the Spring Boot applications tested were vulnerable to SQLi and XSS. There is tremendous overlap with the attacks targeting web applications and APIs, and as our guest essay pointed out, the problems of old are starting to reappear in today's development lifecycles.

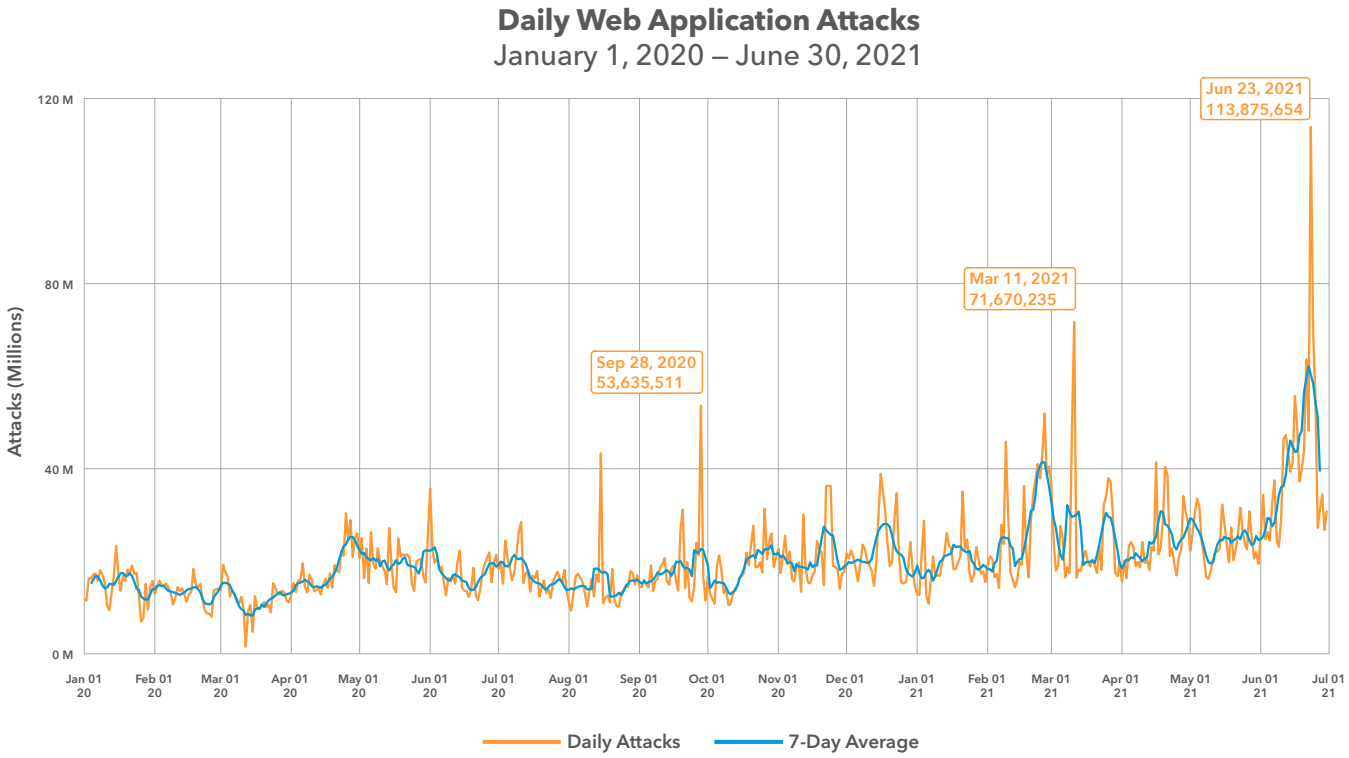

Fig. 2: Web attacks spiked in June 2021, with a peak of 113.8 million attacks in a single day

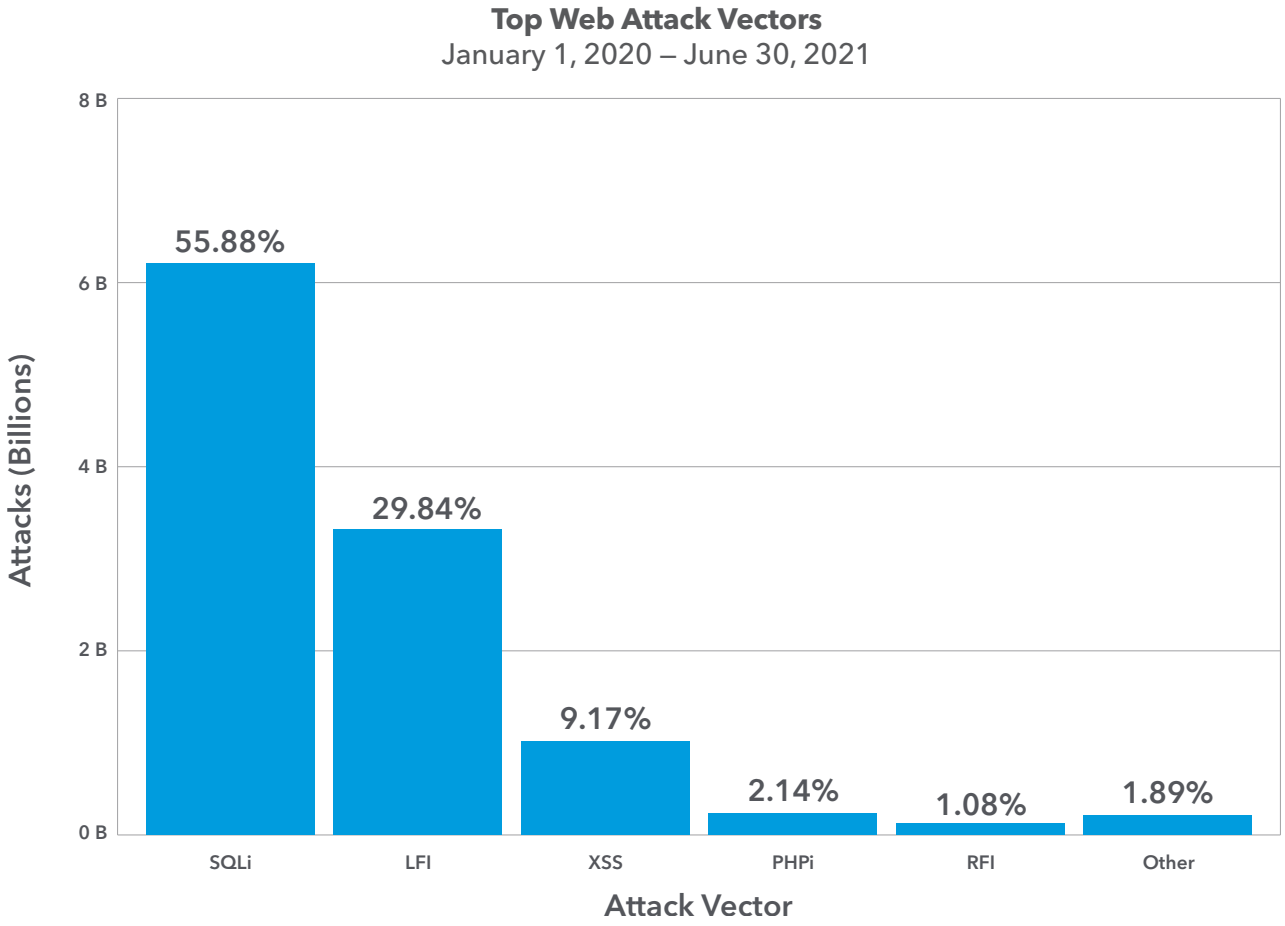

Fig. 3: SQLi remains the top web attack vector, as criminals look to exploit applications and APIs for access to sensitive or protected information

Credential abuse

Over the past 18 months, credential stuffing attacks have remained steady, with dips and peaks in the first two quarters, followed by two notable attacks in January and May 2021. On those dates, credential stuffing attack traffic surged past 1 billion attacks for the day (Figure 5).

The main cause of these attack spikes is unknown, but it's possible they are related to a number of credential services that appeared on several markets in 2020 and continue to operate to this day. One of those services, which started offering access in February 2020, promises to deliver constant updates and serviceable dehashed credentials in a user:pass format, which is exactly what criminals look for when buying or trading combination lists.

To be fair, some of these services are scams, but those operations don't last long, as marketplaces tend to self-regulate against scammers. The seller referenced in Figure 4 was selling access to more than 200 GB worth of credentials at the time of the initial offering. The service offers buyers a mix of basic or more targeted combinations, including user:pass records focusing on retail, gaming, crypto markets, countries, etc.

Moreover, this individual updates their collection on a somewhat regular basis. For example, based on their posts, it was claimed that more than 430 million combinations were added between February and April, and 144 million in May 2021.

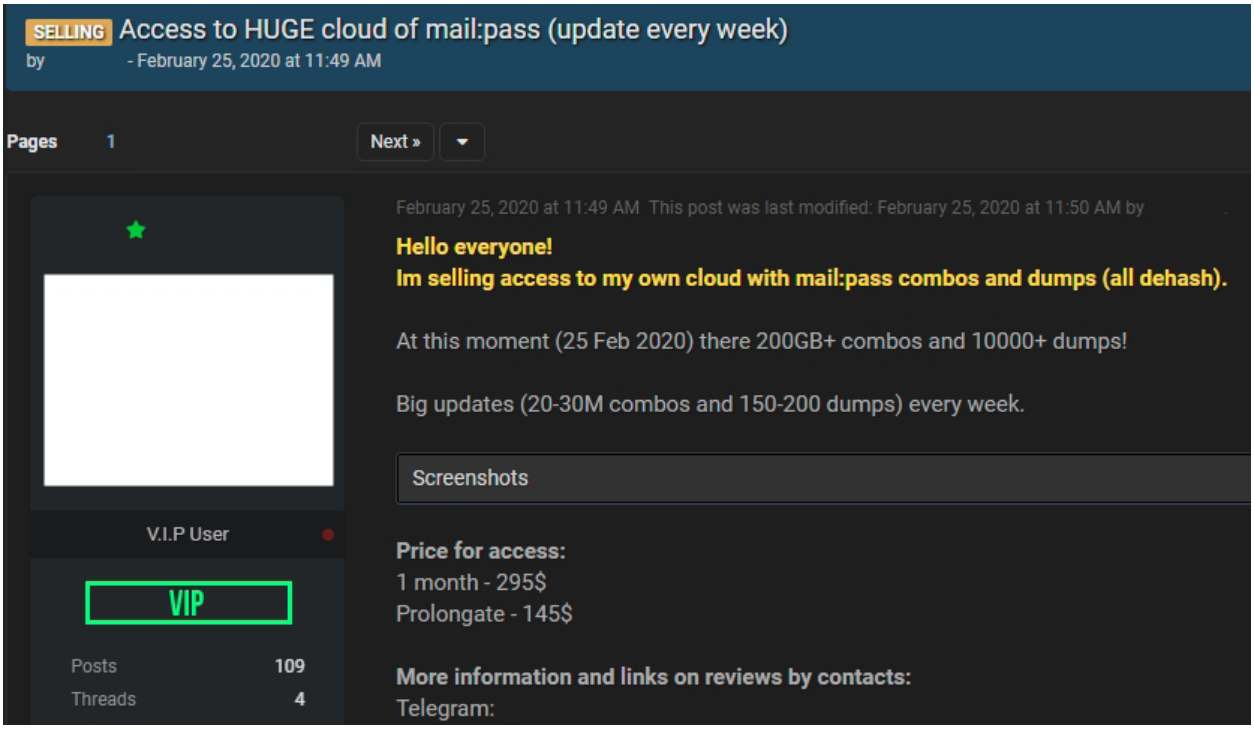

Fig. 4: One service on a popular forum offers a steady stream of credentials that can be used in credential stuffing attacks

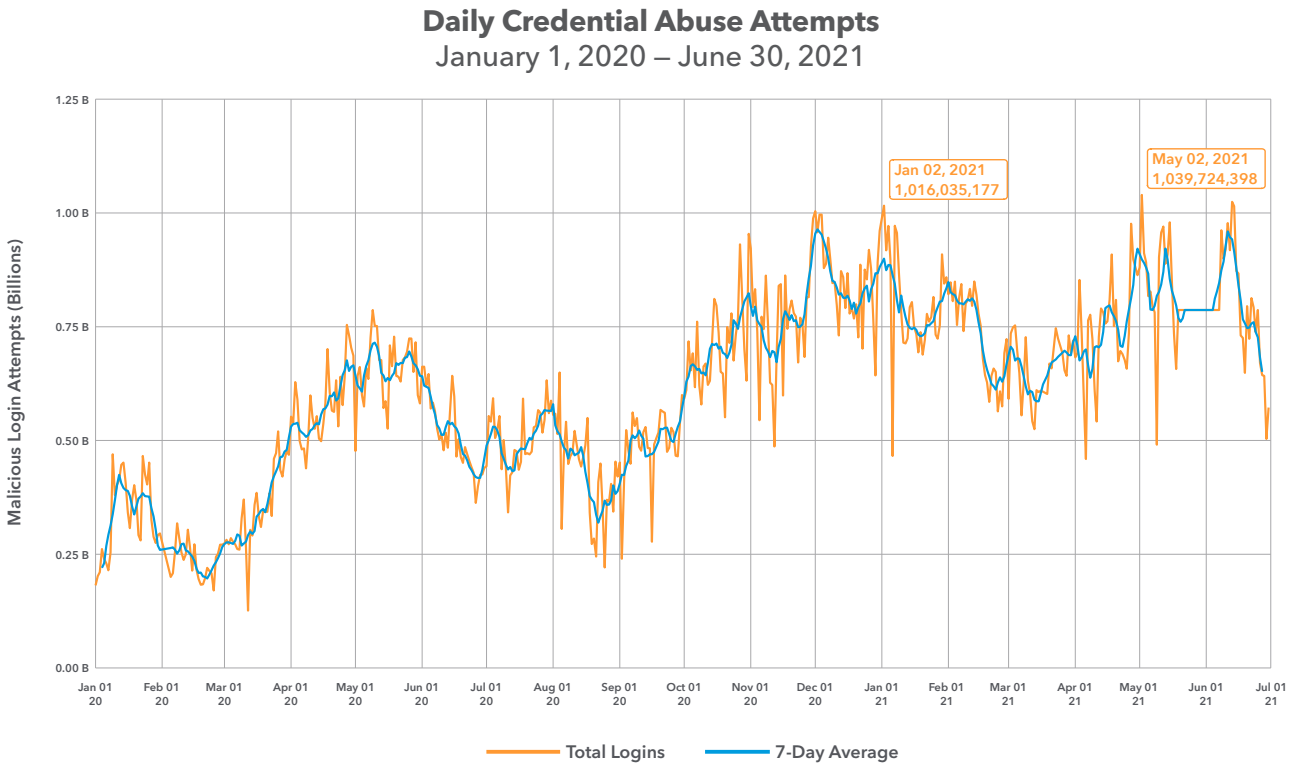

Fig. 5: Credential stuffing attacks remained steady over the past 18 months, but two days in Q1 and Q2 2021 hit peaks of over 1 billion daily attacks

It could just be a coincidence, but the two peaks in our data set were recorded on the second of each month (January and May 2021), respectively. There is no indication as to why criminals were hitting their hardest on the second, compared to any other day during those months.

Targets and Sources

The United States claims the top spot as a target for web application attacks during the recorded period, taking on nearly six times the amount of attack traffic in the United Kingdom, which ranked second (Figure 6). The list is rounded out by India, Austria, and Canada. Given the fact that the United States is home to most of the major targets on the internet, their place as the top target doesn't come as a shock to us.

The United States also claims the top spot on the source list for attacks, taking first place away from Russia, with almost four times the amount of traffic (Figure 7).

Top Target Areas for Web Application Attacks
January 1, 2020 – June 30, 2021

TARGET AREA	ATTACK TOTAL	GLOBAL RANK
United States	5,998,188,041	1
United Kingdom	1,021,638,223	2
India	825,061,439	3
Austria	309,373,274	4
Canada	282,846,738	5

Fig. 6: The United States was once again the top target for web attacks, followed by the United Kingdom

Top Source Areas for Web Application Attacks
January 1, 2020 – June 30, 2021

SOURCE AREA	ATTACK TOTAL	GLOBAL RANK
United States	4,019,434,857	1
Russia	1,146,258,871	2
India	910,264,770	3
Netherlands	642,859,781	4
Germany	640,368,111	5

Fig. 7: The United States was the top source for attack traffic, with nearly four times the traffic coming out of Russia

The attack source is an interesting metric to follow, since Akamai can only see the final hop in the attack chain – where the attacker finally makes a connection to the target. While the United States is on top, that doesn't mean the attack originated from there. In reality, criminals will relay attack traffic from a number of sources, with a goal of hiding their identity and location.

DDoS

DDoS traffic has remained consistent, with peaks recorded earlier in Q1 2021. In January, Akamai recorded 190 DDoS events in a single day, followed by 183 in March. We expect as the year moves on, we'll see additional spikes, as criminals tend to favor DDoS alongside other attacks.

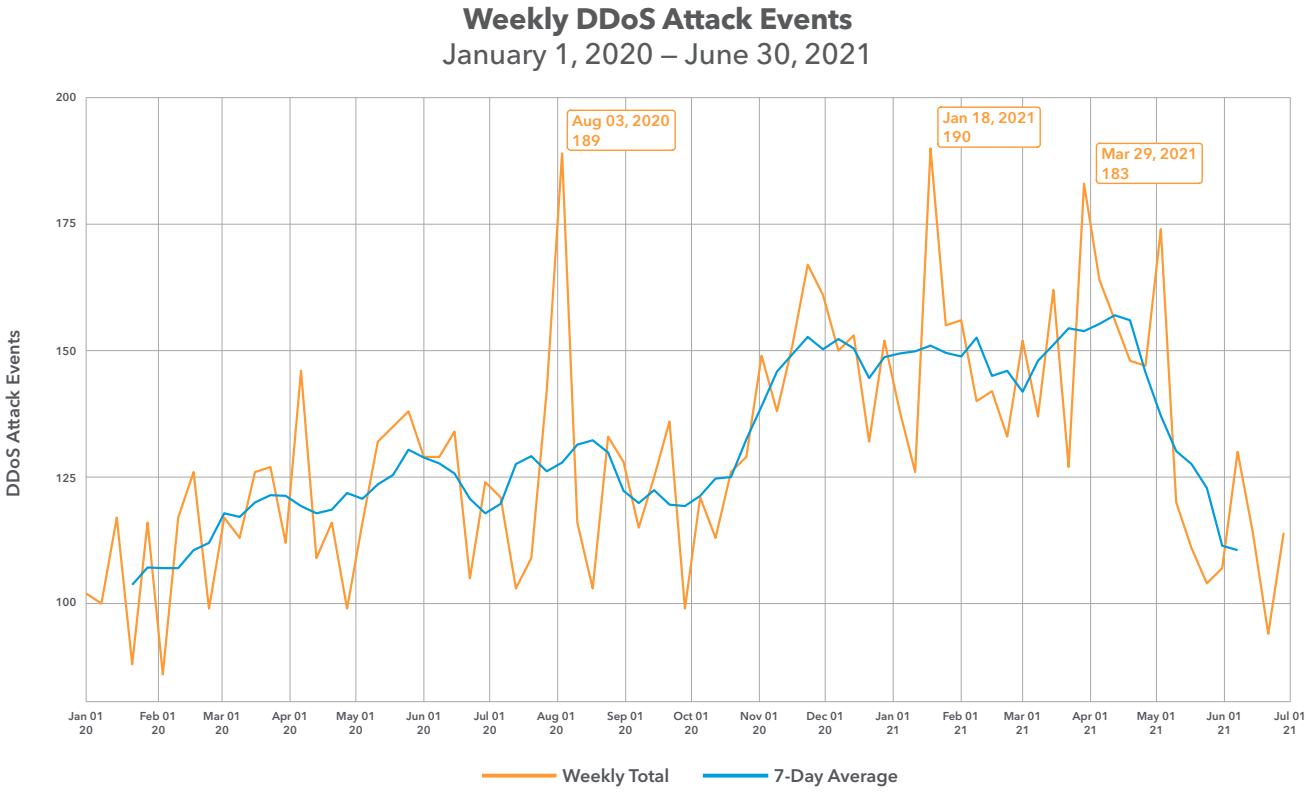

Fig. 8: DDoS attacks remained consistent over the 18 months observed, with spikes in Q1 2021

Conclusion

Application security, either on the API side or the web application development side, is a complex mash-up of features, functions, and business demand. Finding balance within this realm isn't as easy as one would think.

As our data shows, the attackers are clearly out there lurking, developing new techniques and attack methods – and API functionality is one of their primary targets. While teams are moving toward having security baked into the development lifecycle, the process is slow. This leaves organizations behind the eight ball and forces them in some cases to launch known vulnerable code into the wild, because the business use for said code is critical.

In addition to some recommended practices in Appendix A, it's clear that more continuous training is needed for app development and API implementation, and only then can development teams deliver on the expectation of security.

Appendix A – Best Practices: API Security

For this report, we spoke to a number of experts, and considered several sources for information related to API and application security, including Veracode, OWASP, MITRE, and 42Crunch.

1. Discover your APIs and track them as you would inventory. Many organizations have experienced an incident involving an API that they were not even aware existed. So knowing where the APIs are, and what they're used for, is essential. Also related to this are the external APIs that the organization consumes. Those too will need to be identified and secured, or at the very least registered as possible risk items and assessed.

2. Once you know where all your APIs live, test them and understand what vulnerabilities exist within them. This will require testing tools and solid developer education, as well as partnership with existing security teams.

There will need to be a discussion about risk tolerance and plans developed to fix the vulnerabilities sooner rather than later. Start off by looking for hard-coded keys, logic calls, and whether API traffic could be compromised by an impersonation attack. It's also a good idea to scan storage and repositories for keys that could be used to compromise the API or anything associated with it.

3. Leverage existing WAF infrastructure, as well as any identity management and data protection solutions, alongside any specialized API security tools, both during development and launch. In addition, ensure that API security is a continuous thing and not a one-off checkbox during development. New vulnerabilities and attacks are discovered all the time, and single-instance checks will leave the attack surface exposed.

4. When it comes to API policies, try to avoid the use of unique policies per API, instead favoring a blanket set of policies that can be reused. Moreover, don't code policy directly into the APIs that need protection. Doing so violates separation of duty mechanics, adds unnecessary complexity, adds additional overhead burdens for those who maintain the code, and denies visibility to security teams. A good rule of thumb is making the default access level to any resource null, or denied. This enforces least privilege, and makes authentication a constant requirement.

5. API development needs to — on some levels — include various stakeholders. This includes development teams, network and security operation teams, identity teams (if they fall outside of the operation teams), risk managers, security architects, and legal/compliance teams (to ensure the product follows all governance and regulatory laws).

When it comes to API security at the OWASP level, APISecurity.io is hosting a cheat sheet developed by 42Crunch, which is well worth reading.

The entire OWASP Top 10 for applications is available on the OWASP website. Each item offers tips and tricks for defenders.

Appendix B — Methodologies

Web application attacks

This data describes application-layer alerts generated by Kona Site Defender and Web Application Protector. The products trigger these alerts when they detect a malicious payload within a request to a protected website or application. The alerts do not indicate a successful compromise. While these products allow a high level of customization, we collected the data presented here in a manner that does not consider custom configurations of the protected properties.

The data was drawn from Cloud Security Intelligence (CSI), an internal tool for storage and analysis of security events detected on the Akamai Intelligent Edge Platform. This is a network of over 300,000 servers in more than 4,000 locations on more than 1,400 networks in 135 countries. Our security teams use this data, measured in petabytes per month, to research attacks, flag malicious behavior, and feed additional intelligence into Akamai's solutions.

Credential abuse

Credential abuse attempts were identified as unsuccessful login attempts for accounts using an email address as a username. We use two algorithms to distinguish between abuse attempts and real users who can't type. The first is a simple volumetric rule that counts the number of login errors to a specific address. This differs from what a single organization might be able to detect because Akamai is correlating data across hundreds of organizations.

The second algorithm uses data from our bot detection services to identify credential abuse from known botnets and tools. A well-configured botnet can avoid volumetric detection by distributing its traffic among many targets, using a large number of systems in its scan, or spreading out the traffic over time, just to name a few evasion examples.

This data was also drawn from the CSI repository. One customer with significant attack volume was removed from this data set prior to 2020, due to not having a full year of data.

It is important to note the credential abuse (CRAB) data collection was interrupted from May 19 through June 7, 2021, and as a result no data was collected during this time. To continue utilizing the CRAB data set, the number of daily total malicious login attempts was imputed using a simple median calculation.

The median daily total malicious login attempts for Q2 was calculated with the existing data and imputed as the number of daily total malicious login attempts for each day missing data in this time frame. This method, as well as many others, was tested against the full Q1 2021 data set, and the simple median calculation was determined to be the closest estimate to the known total value. No data relating to accounts (i.e., vertical, subvertical, target country, source country, regions) was imputed, and therefore no calculations concerning that information will be available for Q2 2021.

DDoS

Prolexic Routed defends organizations against DDoS attacks by redirecting network traffic through Akamai scrubbing centers, and only allowing the clean traffic forward. Experts in the Akamai security operations center (SOC) tailor proactive mitigation controls to detect and stop attacks instantly, and conduct live analysis of the remaining traffic to determine further mitigation as needed.

DDoS attack events are detected either by the SOC or the targeted organization itself, depending on the chosen deployment model – always-on or on-demand – but the SOC records data for all attacks mitigated. Similar to web application traffic, the source is determined by the source of the IP traffic prior to Akamai's network.

Credits

Editorial Staff

Martin McKeay
Editorial Director

Steve Ragan
Senior Technical Writer, Editor

Amanda Goedde
Senior Technical Writer, Managing Editor

Chelsea Tuttle
Senior Data Scientist

Guest Author: Chris Eng
Chief Research Officer, Veracode

Marketing

Georgina Morales Hampe
Project Management

Shivangi Sahu
Program Management

Within this report, Akamai cited Gartner analysis and research data. The information attributed to Gartner came from the following report:

1) Gartner ID: G00404900, 01-March-2021 API Security: What You Need to Do to Protect Your APIs

More State of the Internet / Security

Read back issues and watch for upcoming releases of Akamai's acclaimed State of the Internet / Security reports. akamai.com/soti

More Akamai Threat Research

Stay updated with the latest threat intelligence analyses, security reports, and cybersecurity research. akamai.com/our-thinking/threat-research

Access Data from This Report

View high-quality versions of the graphs and charts referenced in this report. These images are free to use and reference, provided Akamai is duly credited as a source and the Akamai logo is retained. akamai.com/sotidata

Akamai secures and delivers digital experiences for the world's largest companies. Akamai's intelligent edge platform surrounds everything, from the enterprise to the cloud, so customers and their businesses can be fast, smart, and secure. Top brands globally rely on Akamai to help them realize competitive advantage through agile solutions that extend the power of their multi-cloud architectures. Akamai keeps decisions, apps, and experiences closer to users than anyone — and attacks and threats far away. Akamai's portfolio of edge security, web and mobile performance, enterprise access, and video delivery solutions is supported by unmatched customer service, analytics, and 24/7/365 monitoring. To learn why the world's top brands trust Akamai, visit www.akamai.com, blogs.akamai.com, or @Akamai on Twitter. You can find our global contact information at www.akamai.com/locations. Published 10/21.