

2019 APPLICATION PROTECTION REPORT

2ND EDITION

The Virtue of Visibility



Authors



Ray Pompon is a Principal Threat Research Evangelist with F5 Labs. With over 20 years of experience in Internet security, he has worked closely with federal law enforcement in cyber-crime investigations. He was directly involved in several major intrusion cases, including the FBI undercover Flyhook operation and the NW Hospital botnet prosecution. He is the author of *IT Security Risk Control Management: An Audit Preparation Plan* published by Apress books.



Sander Vinberg is a Threat Research Evangelist for F5 Labs. He has worked in information security, geopolitical risk, and linguistic consulting. He holds a master's degree from the University of Washington in Information Management, as well as bachelor's degrees in History and African-American Studies from the University of Chicago.

Contributors

Sara Boddy | Director, F5 Labs

Debbie Walkowski | Threat Research Evangelist, F5

David Warburton | Senior Threat Research Evangelist, F5

Business and Data Partners



BAFFIN BAY NETWORKS is a team of dedicated researchers monitoring and investigating emerging attacks, advanced persistent threats, and the organizations and individuals responsible for them. They also develop research tools to identify, investigate, and track ongoing attacks and emerging threats. Working with Baffin Bay Networks, we analyzed global intrusion and honeypot data collected from web attacks on 21,010 unique networks over 2017.



CYENTIA INSTITUTE is a cybersecurity research services firm. They deliver high-integrity, high-quality, data-driven research which provides security companies with meaningful marketing content to build mindshare, drive sales, and attain greater visibility in competitive markets. In doing so, it seeks to advance cybersecurity knowledge and practice for the community at large.

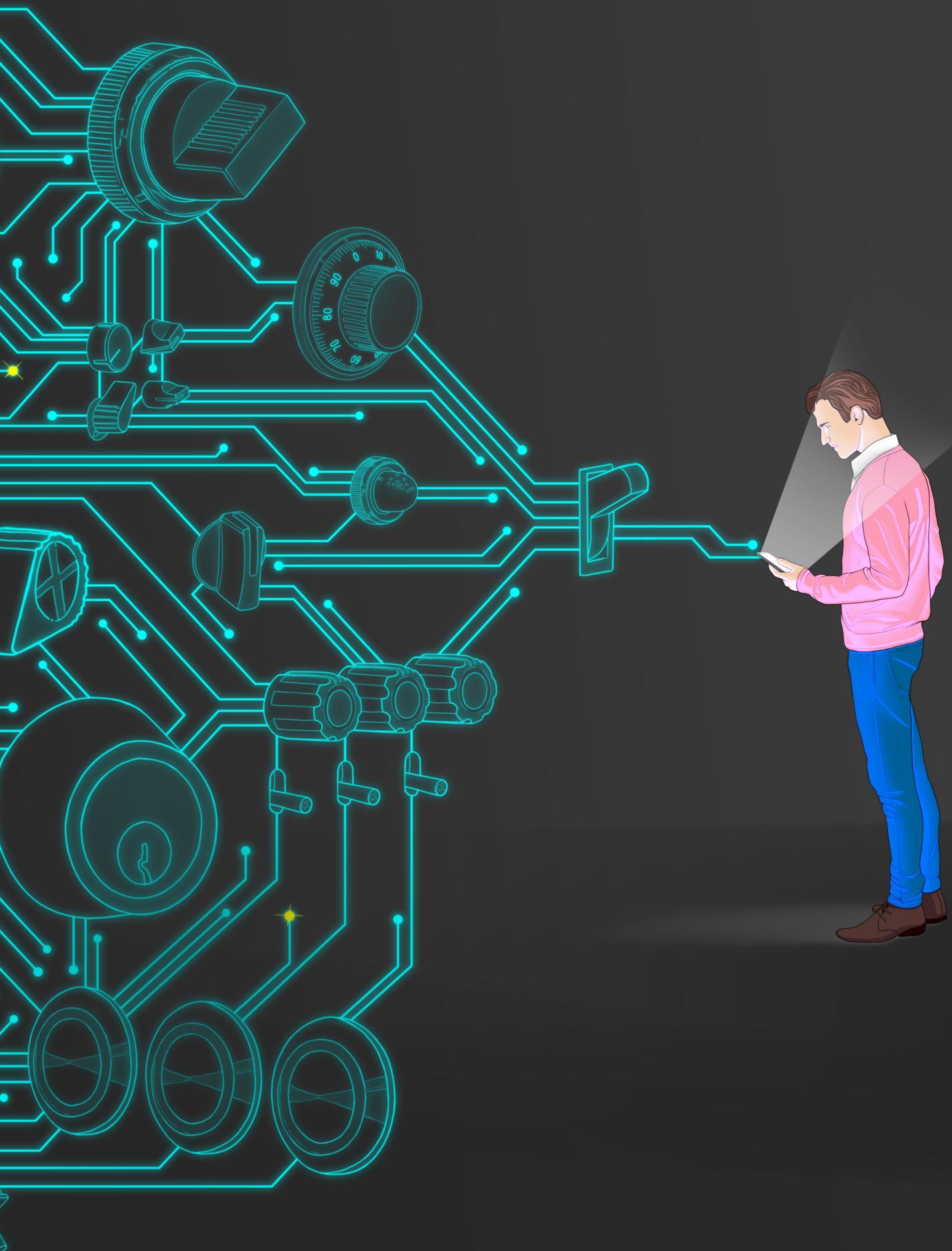


TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
PHP—PRETTY HEAVILY POUNDED	5
THE RELATIONSHIP BETWEEN BREACH CAUSES AND INDUSTRY SECTORS	5
FORMJACKING INJECTIONS GET MEANER	6
ACCESS CREDENTIAL ATTACKS	6
ATTACKER'S EYE ON THE API	7
THE FUTURE OF APPLICATION SECURITY	8
INTRODUCTION	10
Modeling the New Attack Surface	11
PHP RECON RUNS RAMPANT	14
Sensor Network Reveals the Lowest Common Denominators	14
Findings	14
Coordinated Campaign Against phpMyAdmin	15
Two IP Addresses Doing All of the Dirty Work	15
BREACH TRENDS IN 2018	16
Top Threats	16
Sorting Breaches by Cause	17
Industry Profiles	20
What Does All this Mean?	21
INJECTION FOR A DECENTRALIZED AGE	24
Again, We Ask: What Is an App?	24
Injection Still?	24
New Software Supply Chain, Old Attacks	26
What the Data Say	27
Mitigating the Risk of Injection Attacks	29
ACCESS ATTACKS' ENDURING PREVALENCE	34
Phishing	34
Credential Stuffing and Brute Force Attacks	35
Email Hacks	37
Mitigating the Risk of Access Attacks	39

TABLE OF CONTENTS

APIS AND THE NEW-OLD PROBLEM OF VISIBILITY	44
API Usage	44
How Does an API Fit Into a Network?	45
What Makes an API a Good Target?	45
What Kinds of APIs Get Hacked and Why?	46
Public Records Review	50
Incidents (Not Breaches)	50
Mitigating the Risk of API Attacks	50
FINAL THOUGHTS	56
PHP Recon Shows That We Can't Ignore the Trailing Edge	56
The Big Picture with Breach Trends	56
The Future of Injection and Decentralized Web Content	58
Putting Access Attacks in Perspective	58
The Future of API Security	59
Service Level Agreements Coming for Third-Party Services?	60
WE WANT TO KNOW WHAT YOU THINK	60



F5

LABS

Executive Summary

Welcome to F5 Labs' second annual Application Protection Report, a comprehensive investigation of the threats facing applications today and what organizations can do to protect their apps. This year, we have newer, deeper insights from within F5, combined with threat intelligence from Baffin Bay Networks and its global network of over 1,500 sources. We also worked with the Cyentia Institute, the pioneers of security research, who created Verizon's Data Breach Investigations Report. We've got a lot of data-driven insights that we're excited to share!

As always, we focus on applications because that's what our adversaries do. Applications are the battlefield of information security. As the meeting point of users and networks, they are the defining value proposition for most businesses, and the gateway to that which attackers value most: data.

EPISODE 1

PHP—Pretty Heavily Pounded

Since F5 Labs began examining sensor traffic in 2017, the widespread programming language PHP has continually stood out as a favored attack vector, particularly for the less sophisticated end of the attacker spectrum. In 2017, PHP was targeted in 58% of indiscriminate web attacks. In 2018, we saw this rise to 81%. Based on the paths this traffic was looking for, we believe that most of this traffic was looking for old (and probably neglected) systems MySQL databases with known vulnerabilities, such as PHP database management systems.

The simple takeaway is that if you're using PHP, you're being scanned for weaknesses. Make sure you're patched up, with a careful eye towards known PHP exploits like cve-2018-12613 and cve-2018-20062. And if you've got any PHP-enabled admin interfaces online, you need to lock them down tight. The higher-level takeaway is that traffic like this is a reminder that old vulnerabilities never quite go away. It is easy to watch the threat landscape change and become focused on new threats. However, reconnaissance campaigns that are seeking systems with eight-year-old vulnerabilities demonstrate that we are always building on top of our past, and the new threats and new vulnerabilities that come on the scene do not erase the old ones.

EPISODE 2

The Relationship Between Breach Causes and Industry Sectors

To determine which tactics were resulting in actual breaches, we examined public breach disclosures that organizations provided to U.S. state attorneys general from 10 states. We looked at 761 unique events in 2018, and 1,025 in 2019. Access-related breaches made up the largest proportion for both years, at 47% in 2018 and 52% in 2019, with web breaches next at 17% in 2018 and 19% in 2019.

The nature of the breaches depended on how different industry sectors tend to collect and store their valuable data assets. Organizations from industries that depend on ecommerce, like retail, are more likely to be breached by web attacks, whereas other industries such as finance and healthcare are more likely to be compromised by access attacks. We will explore each of these two most prevalent attack types in more detail below, with an emphasis on how they have evolved in response to changes in how applications are built and how people interact with them.

EPISODE 3

Formjacking Injections Get Meaner

Injection attacks took a front seat in breaches in both 2018 and 2019, largely due to the prevalence of formjacking attacks. Formjacking uses code injected by an attacker to siphon payment card information from an online form and deliver it to the attacker. We found 83 breaches that were attributable to formjacking in 2018, and 145 in 2019.

While formjacking is not a new technique, it has become more lucrative in the last few years because of a trend of decentralization in web applications. More and more web applications are using third-party services that run in the client browser to deliver mission critical business functions, and payment card processing is one of the most prevalent services to be outsourced. This allows attackers to target third-party services with injection attacks whose malicious payload runs on the client browser, unbeknownst to both the main application owner and the victim.

THE RISK OF THESE KINDS OF ATTACKS IS MAGNIFIED WHEN THE TARGET WEB APPLICATION USES THIRD-PARTY CODE RUNNING OFFSITE.

The recent rise of formjacking indicates that any organization that accepts payment card information over the web is going to have their shopping cart targeted, regardless of sector. The risk of these kinds of attacks is magnified when the target web application uses third-party code running offsite. We strongly recommend thorough testing and watching of all third-party components on sites with forms accepting critical information.

EPISODE 4

Access Credential Attacks

While injection played a significant role in the breaches that were caused by web attacks, those breaches were far surpassed by the number of access-related breaches. In 2018, 47% of all breaches we looked at were attributable to some kind of access attack, and 52% were in 2019. The percentage of breaches that were specifically caused by compromises of email systems rose from 20% to 33% between 2018 and 2019. Even though organizations often go to great lengths to protect known sensitive information in hardened databases, there is often also a huge amount of sensitive information like passwords, personal information and intellectual property, sitting unencrypted in email.

The number of phishing breaches actually dropped, from nearly 20% of breaches in 2018 to 14% in 2019. We suspect that this is more due to differences in reporting than a true drop in phishing, since we know that phishing has grown more sophisticated across the attacker spectrum.

Brute force attacks are another type of access attack that have raised questions in 2018 and 2019. Brute force attacks have low success rates, but are simple and cost attackers almost nothing in terms of effort, so they remain popular. Even when they fail, they can impede victims' network performance, resulting in indirect denial-of-service attacks. Furthermore, as more organizations have implemented systems to recognize blatant brute forcing, some attackers have adapted by moving to "low-and-slow" techniques that are more difficult to identify without the most advanced web application firewalls (WAFs).

AS MORE ORGANIZATIONS HAVE IMPLEMENTED SYSTEMS TO RECOGNIZE BLATANT BRUTE FORCING, SOME ATTACKERS HAVE ADAPTED BY MOVING TO "LOW-AND-SLOW" TECHNIQUES THAT ARE MORE DIFFICULT TO IDENTIFY WITHOUT THE MOST ADVANCED WEB APPLICATION FIREWALLS (WAFS).

Because access attacks bring force to bear on the interface between users and systems, many of the controls that would theoretically stop attacks also result in preventing legitimate access as well. As a result, they can be significantly more difficult to mitigate in practice than in theory. While multifactor authentication represents a significant improvement, it can be challenging to implement. Therefore, we recommend substantive monitoring and logging practices, updating password policies according to contemporary guidelines that are significantly more user-friendly, and working to reduce the amount of sensitive information at rest in email.

EPISODE 5

Attacker's Eye on the API

Application programming interfaces (APIs) are like user interfaces for other machines. They allow applications or sub-application services to connect with one another to exchange data, which offers significant opportunities for integration and improved scaling. As part of the broader trend of decentralization, APIs have played an increasingly prominent role in emerging patterns of web architecture.

However, because they are not intended for human use, and because they are often set up and put into production quickly, they have come to represent an easy path for attackers to circumvent controls and gain access to data. Often APIs are set up with no authentication at all, or with overly broad permissions, which means that a shrewd attacker who knows how to find it can exfiltrate data quickly and quietly.

In 2018 and 2019, API breaches fell into three categories: large web applications with hundreds of APIs, mobile applications, and misconfigured applications lacking authentication. Mitigating the

risk that APIs pose requires many of the same controls and practices that we would employ for any web application: inventory, access control, encryption, and vulnerability management.

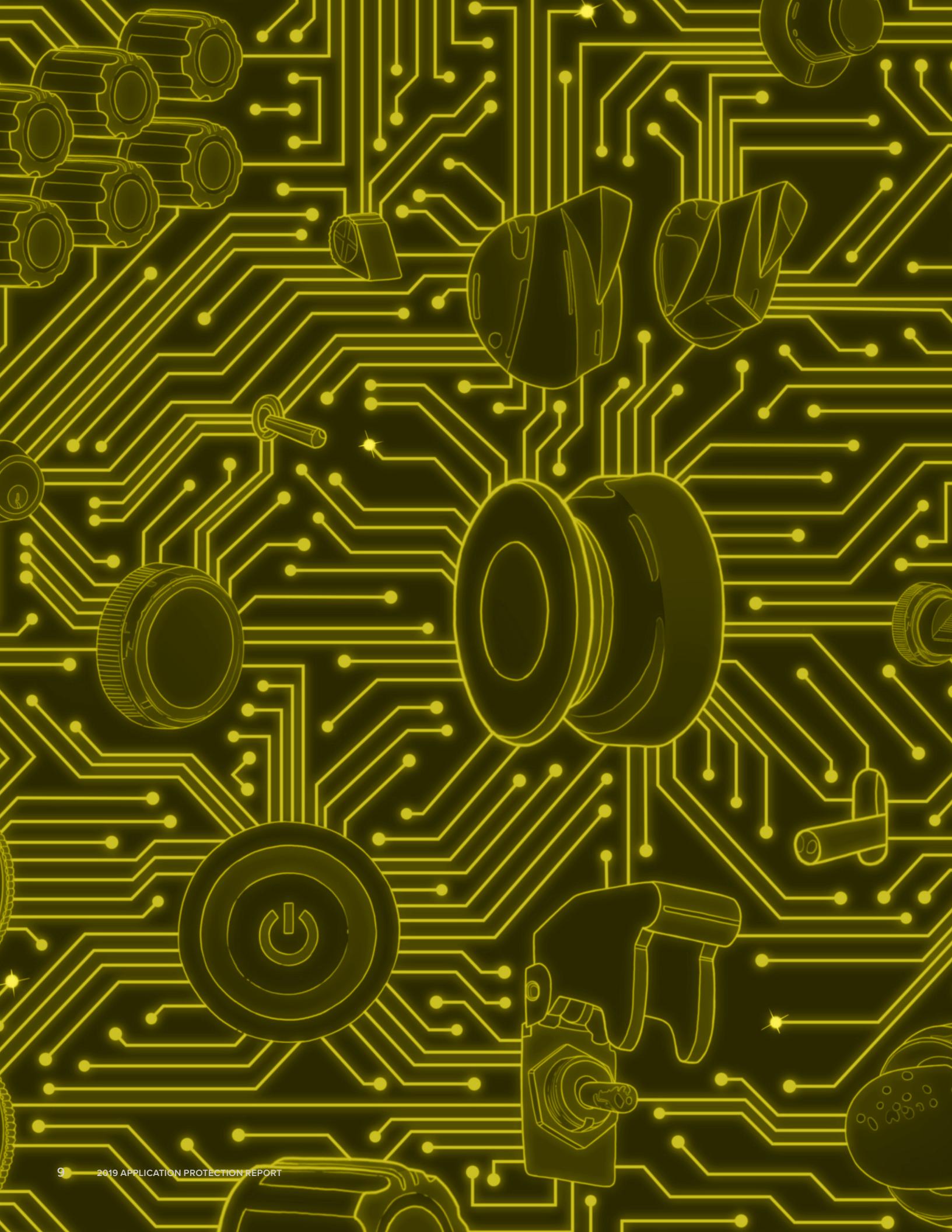
The Future of App Security

Our analysis of the 2018 threat landscape has illuminated ways that known risks have transformed as a result of widespread changes in web environments. The broad trend of decentralization, along with the added abstraction and infrastructural complexity that that entails, has given new life to simple practices like injection or circumventing authentication. Conversely, nothing we saw in the breach reports or sensor data indicated that truly new or particularly sophisticated techniques were resulting in breaches, or at least the breaches we saw.

SINCE APIs ARE OFTEN SET UP AND PUT INTO PRODUCTION QUICKLY, THEY HAVE COME TO REPRESENT AN EASY PATH FOR ATTACKERS TO CIRCUMVENT CONTROLS AND GAIN ACCESS TO DATA.

Underneath these widespread movements toward disintegration, however, the correlation between industries and attack vectors shows that even amidst broad changes in how web applications are designed and run, security programs need to be designed with a granular understanding of where and how valuable data are stored. In other words, the transformations reshaping web applications only serve to strengthen the argument for risk-based security programs. With that in mind, F5 Labs will continue to collect, analyze and deliver the most actionable and contextually rich threat intelligence for security practitioners everywhere.

If you have feedback, data to share, requests for topics, or thoughts about our approach, please let us know. You can reach us on Twitter @f5labs, or email us at F5LabsTeam@f5.com.



Introduction

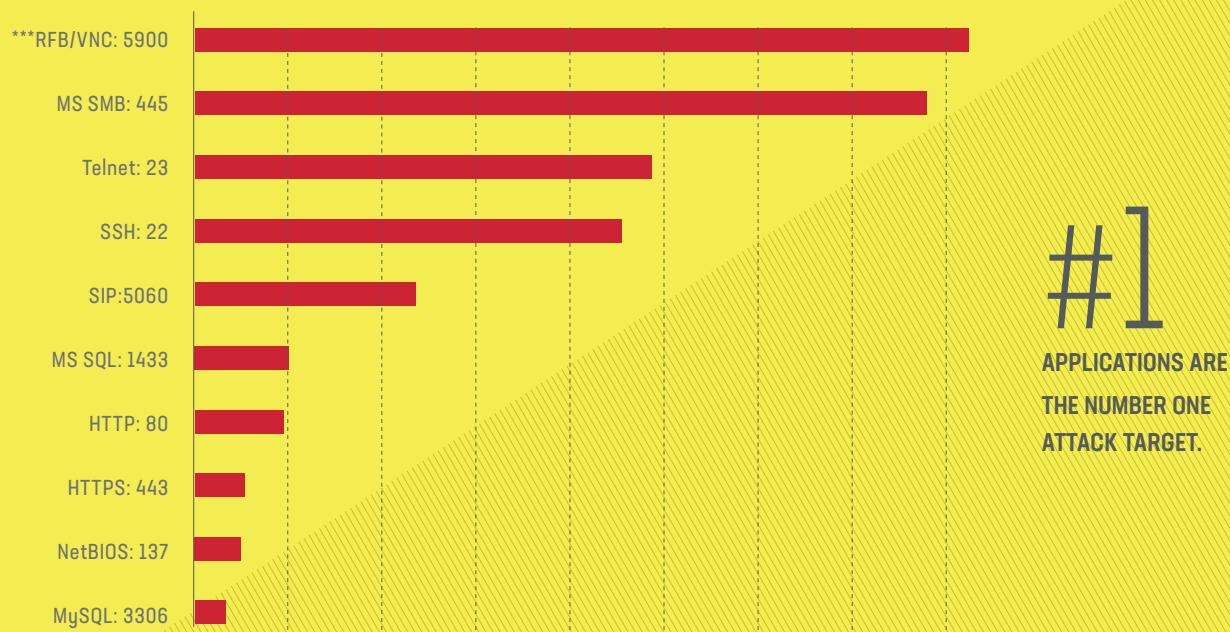
**FIGURE 1
2019 ATTACKS BY TOP 10 DESTINATION PORTS**

This chart shows the top ten target ports by honeypot traffic volume. While not every port shown here is used exclusively for application traffic, these data still demonstrate the degree to which known-malicious traffic focuses on a small number of ports that are associated with applications.

In day-to-day conversation, we still tend to describe the Internet as being composed of “sites.” However, today web applications, not sites, are the engines that drive both Internet traffic and business. Webmail, ecommerce, social media, online banking, eLearning, web search, and media streaming all happen through web applications. Unlike static web sites, web applications do not just process requests for existing HTML files on a server and deliver them; they also accept user input, process it, and return data. Because they transmit, store, and process data, they are also the single most attacked asset on the Internet and account for a wide range of compromises. In a separate F5 Labs breach trends report, our research showed that applications were the initial targets in 53% of breaches over the past decade.

In short, this report series focuses on applications because they have quietly but inexorably taken center stage as the defining component of the Internet. This might sound obvious, but the security industry isn’t really keeping up with the trends (for more on this topic, see our three-part series on the gap between theory and practice in information security).¹ Attackers, on the other hand, are on top of it. As you’ll see in this report, the only things that attackers focused on more in 2019 than web applications were the users connecting to those applications—and then only as a first step to gain access to data.

FIGURE 1: 2019 TOP 10 ATTACKED PORTS GLOBALLY



***Attack campaign not “normal” attack traffic. SIP 5060 began in 2018 and ended in Q2 2019, VNC 5900 began June 2019

Modeling the New Attack Surface

At the heart of the Application Protection Report series is the aggregation and sharing of big-picture data. We believe that all information security needs to be risk-based. Everyone at F5 Labs has worked in security in one form or another, and we understand the realities at play in the industry, so we mean no rebuke. However, as a community, we need to move beyond compliance, checklists, anecdotes, hunches, astrology, and tea leaves for building security programs. At the same time, information systems themselves are not naturally forthcoming with information, organizations often don't share unless they have to, and we don't usually know what risks our individual organizations face until we've already been hacked. This is what makes threat intelligence the leading edge of risk-based security, and this is why we work the way we do.

As we were developing the 2018 report last year, we realized that we needed to reconceptualize how we think about applications and their attack surface. The result was a conceptual model that uses tiers to display the various transmission and processing functions necessary for an internetworked application to run. We mapped different attack techniques to the different application tiers (and found, unsurprisingly, that it was a many-to-many mapping), and developed a model of attack surfaces that is two-dimensional, containing breadth as well as depth. This is far more effective than just counting how many machines had a specific vulnerability.

Because the app tiers capture the complexity and interdependence of contemporary applications, they now underpin all of our research, from tactical threat pieces to our strategic CISO-to-CISO series. They also helped us structure the questions and analyses that we used to analyze each specific risk that we examined for this report. You will see us refer to the app tiers throughout this report to help tie each piece of intelligence into a broader picture.

FIGURE 2: APPLICATION STRUCTURE AND THE ATTACKS AT EACH LAYER

To protect your apps, you need to understand how they're structured and how they work—and the threats that target each layer.

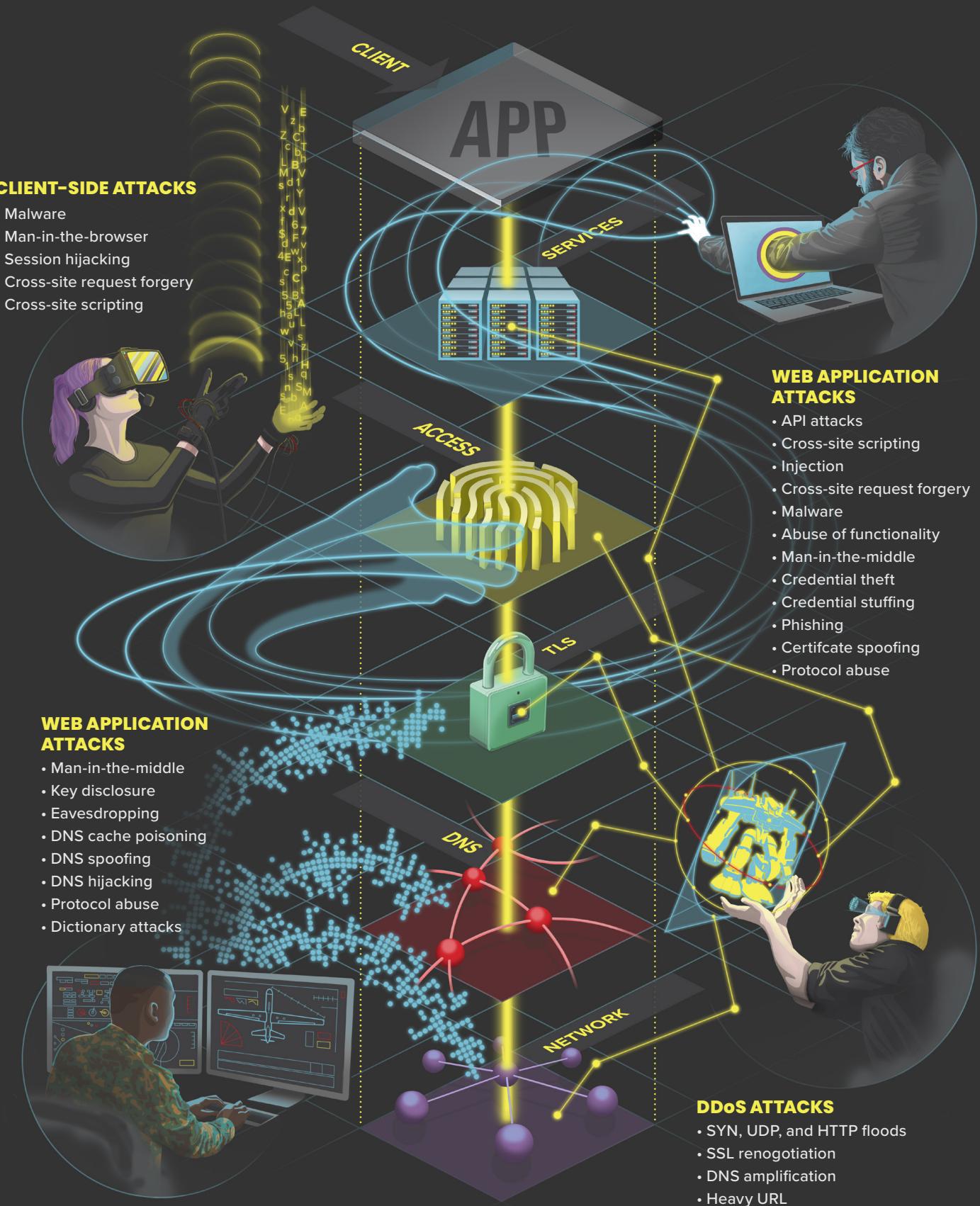
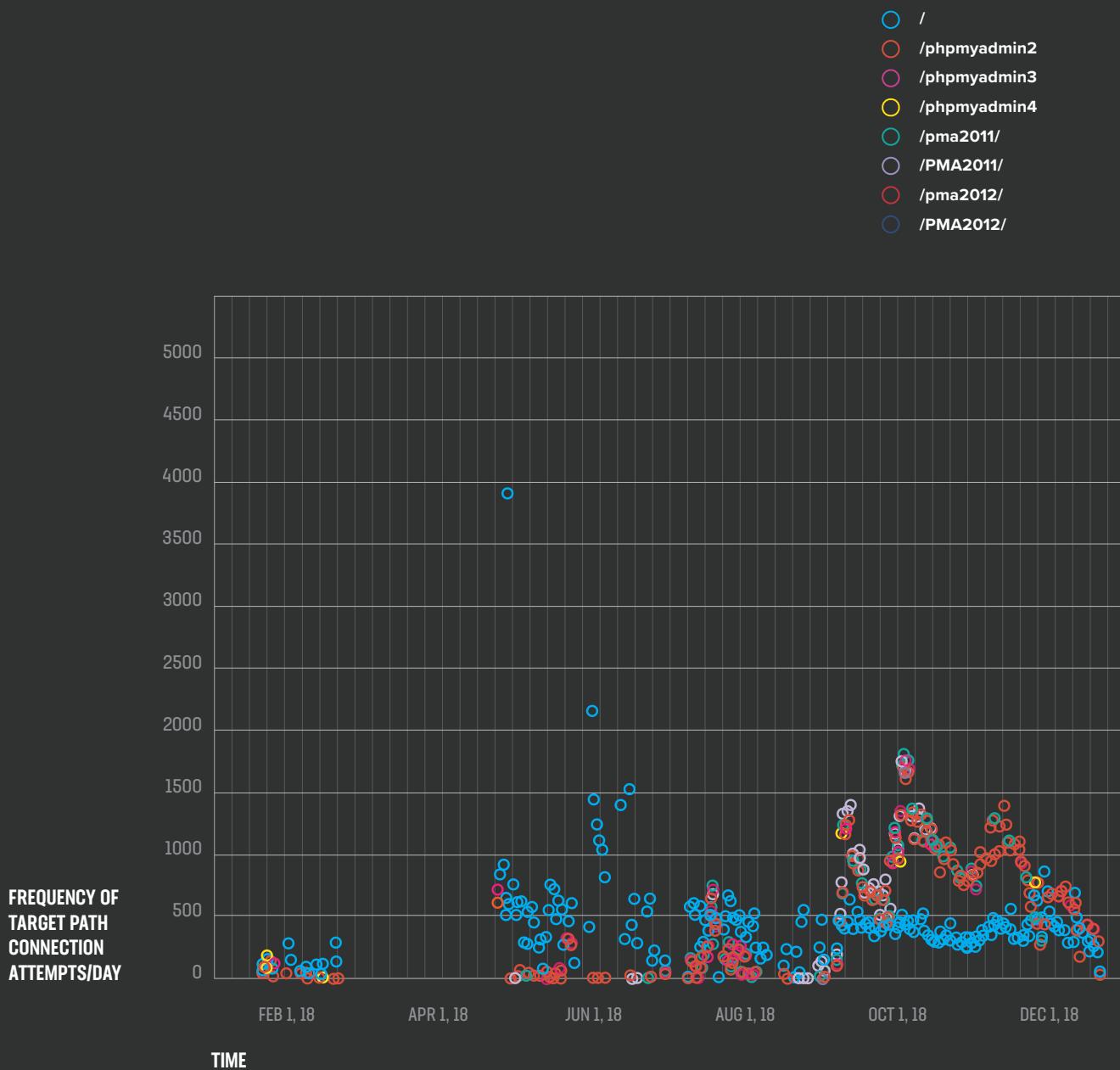


FIGURE 3: PMA CAMPAIGNS VERSUS DOMAIN-ONLY TRAFFIC

Coordinated campaigns targeting seven phpMyAdmin paths compared with traffic targeting web servers with no specified paths.

*Note that the data show a gap between March and June 2018 when Loryka's port 80 sensors went dark.



EPISODE 1

PHP Recon Runs Rampant

As mentioned earlier, [PHP](#) is a widespread and powerful server-side language that's been used in 80% of sites on the web since 2013.² It underpins several of the largest web applications in the world, including WordPress and Facebook.³ This prevalence, particularly among beginning web developers, also makes it a big target. One of our most valuable partners provided us with information that elaborated just how big a target it is.

Sensor Network Reveals the Lowest Common Denominators

Baffin Bay Networks, with its distributed network of more than 1,500 sensors around the world, provides a rare view of the threat landscape. Because these sensors neither advertise nor provide any actual service, we can assume that all traffic that they see is either malicious or misconfigured. The global nature of the sensor network allows us to compare different geographical locations and get a sense of patterns across time, space, target types, and payloads.

Generally speaking, sophisticated threat actors do not like to reveal their hands, and so tend not to spew traffic all over the Internet looking for targets. The Baffin Bay data, therefore, tell us more about reconnaissance campaigns, unsophisticated attacks, and people looking for targets of opportunity than it does about the leading edge of threat intelligence. However, this “trailing edge” of the threat landscape still represents a significant risk to organizations because of the inevitable difficulties in managing vulnerabilities and complex environments over time and space. No organization is immune to these unsophisticated campaigns and as such, they represented a good place for us to start digging.

Findings

From a security standpoint, 2017 was a bad year for PHP. As part of our 2018 report, which focused on trends in 2017, we found that PHP represented 69% of the exploits that Exploit DB published. Our data partner Baffin Bay found that 58% of the web attacks they observed in the wild targeted PHP as their primary attack vector.

The early data indicate that 2018 was just as bad. PHP had almost the same representation (68%) in published exploits on ExploitDB as it had in 2017. In the wild, we saw an even greater proportion of PHP-related traffic. Baffin Bay sensors found that 81% of the malicious traffic they detected was focused on PHP in one form or another. The implication is that PHP will likely remain one of the Internet's weakest links and broadest attack surfaces for the foreseeable future.

On closer examination, the Baffin Bay data also shed some light into the specific tactics of attackers who target PHP, as well as some low-cost steps to mitigate some of the risks posed to one of the web's most prevalent platforms. Baffin Bay sensors identify connection attempts and

capture the source IP and target URL, among other things.

The target domain or target IP address is not significant, since attackers often cycle through millions or billions of targets looking for opportunities to attack. However, the back half of the target URL contains the target file or path, the specific location on a web server that the attacker is targeting across all of its target IP addresses. This tells us much about an attacker's goals and tactics.

Coordinated Campaign Against phpMyAdmin

The first thing that stood out about the Baffin Bay dataset was that while there was a great deal of variation in the target paths—with more than 100,000 unique values in the dataset—a huge portion of traffic was focused on just seven paths or filenames. All seven are commonly used for managing phpMyAdmin (also known as PMA), which is a PHP web application used for managing MySQL databases. Of the ~1.5 million unique events that Baffin Bay captured targeting more than 100,000 different URL paths, 667,000 (42%) were targeted at PhpMyadmin.

Two IP Addresses Doing All of the Dirty Work

When we dug deeper, we found that 87% of the traffic pointed at these common PhpMyadmin paths came from just two IP addresses, out of more than 66,000 IP addresses that hit Baffin Bay sensors. In fact, the two IP addresses in question represented a huge proportion (37%) of the total traffic. Furthermore, all of the traffic from these two IP addresses was pointed at the seven PMA paths. By contrast, no other single IP accounted for anything near as much traffic, nor did any other IP feature traffic patterns over time matching these two, even when they targeted these paths.

WHEN WE DUG DEEPER, WE FOUND THAT 87% OF THE TRAFFIC POINTED AT THESE COMMON PHPMYADMIN PATHS CAME FROM JUST TWO IP ADDRESSES, OUT OF MORE THAN 66,000 IP ADDRESSES.

The two IP addresses in question have been allocated to systems on a North American university campus since before the beginning of the campaign. There are a handful of published exploits regarding the specific versions of phpMyAdmin referenced in these paths, such as 2011-4107 and 2013-3241.⁴ However, to our knowledge all of them require prior authentication to phpMyAdmin, which means that in all likelihood the traffic targeting these paths was looking for poorly controlled authentication portals.

In sum, our investigation of Baffin Bay's sensor data revealed the specifics of the unsophisticated end of the threat spectrum: PHP continues to lead the pack in terms of providing rich, soft targets, and situational awareness remains important in terms of mitigating both vulnerabilities and threats.

EPISODE 2

Breach Trends In 2018

Our data breach analysis is based on U.S. state-level breach notifications that organizations are legally obligated to provide whenever personally identifiable information under their control is exposed either to attackers or to the public. Data in this form have some limitations—most specifically that these are legal documents, not technical ones, and lawyers often include only the minimum legally obligated amount of detail in order to protect their clients. Furthermore, the obligation to report varies from state to state. Some states mandate reporting for breaches involving their citizens, while others require reporting for any organization that operates in their state. Nevertheless, the legal underpinning for these notifications also means that they are the closest thing we have to verifiable and comparable breach data.

For 2018 and 2019, we looked at breaches reported to 10 states representing 21.4% of the U.S. population: California, Washington, Wisconsin, Vermont, New Hampshire, Iowa, Maryland, Oregon, Idaho, and Delaware. Here are the conclusions we drew.

Top Threats

In our 2018 report, we found that two threat vectors—code injection and phishing—had become significant and growing problems. Payment card skimming via injection (also known as formjacking) was the single greatest threat to applications. These attacks exploit injection vulnerabilities to load payment card skimmer scripts into payment forms. Attacks of this type constituted 21% of the breaches we analyzed, and included many of the most significant injection-based breaches in 2017. Most of the compromised payment forms and shopping cart applications ran on PHP; additionally, we found that PHP exploit attempts made up 58% of the total attack traffic observed in 2017 by Baffin Bay sensors.

Phishing and other access control attacks were the second greatest threat, representing 14% of all 2017 breaches that we analyzed. However, over 2018 and 2019, phishing and formjacking have traded places back and forth. In 2018, phishing, at 21% of known-cause breaches, surpassed formjacking at 12%, but in 2019, phishing only accounted for 14%, while formjacking constituted 16.6%. (This is more likely due to changes in breach reporting than significant changes in tactics, given that the vague breach mode ‘email compromise’ grew from 20% to 33% from 2018 to 2019.)⁵

Specific exploits and tactics may shift slightly, but it looks as though the two weakest points on the Internet—people and PHP-based payment card forms—are set to retain their unenviable crowns.

Sorting Breaches by Causes

As noted above, breach notification letters usually lack technical detail. Thirteen percent of breach letters in 2018 did not attribute a specific cause, including the four largest breaches by number of exposed records, and 15% had no cause in 2019. As a result, the breach notifications as a whole provide a sense of overall trends as opposed to specific diagnoses.

Here are the significant breach cause categories as we found them, with our notes in parentheses:

ACCESS-RELATED BREACH CAUSES INCLUDED:

- Email (yes, we also found this annoyingly vague)
- Phishing that resulted in access to email (no other details noted)
- Phishing to gain access to login credentials
- Social engineering by email to gain access (yes, this is probably the same as phishing)
- Brute forcing of credentials
- Credential stuffing
- Stolen access credentials (possibly from a phishing attempt?)
- Access credentials stolen from a third party (could be related to credential stuffing)
- Social engineering by telephone to gain access credentials (“Hi, I’m the county password inspector.”)⁶

WEB BREACH CAUSES INCLUDED:

- Web app code injection attacks (also known as formjacking, Magecart, or skimmer malware)
- Web hacking (no other details noted)
- Web application hacking

ACCIDENTAL BREACH CAUSES INCLUDED:

- Sending information to unintended recipients (wrong attachment or wrong recipient)
- Lost, stolen, or misplaced physical assets (mostly laptops in cars)
- Access misconfigurations that allowed unauthorized access

PHYSICAL SECURITY BREACH CAUSES INCLUDED:

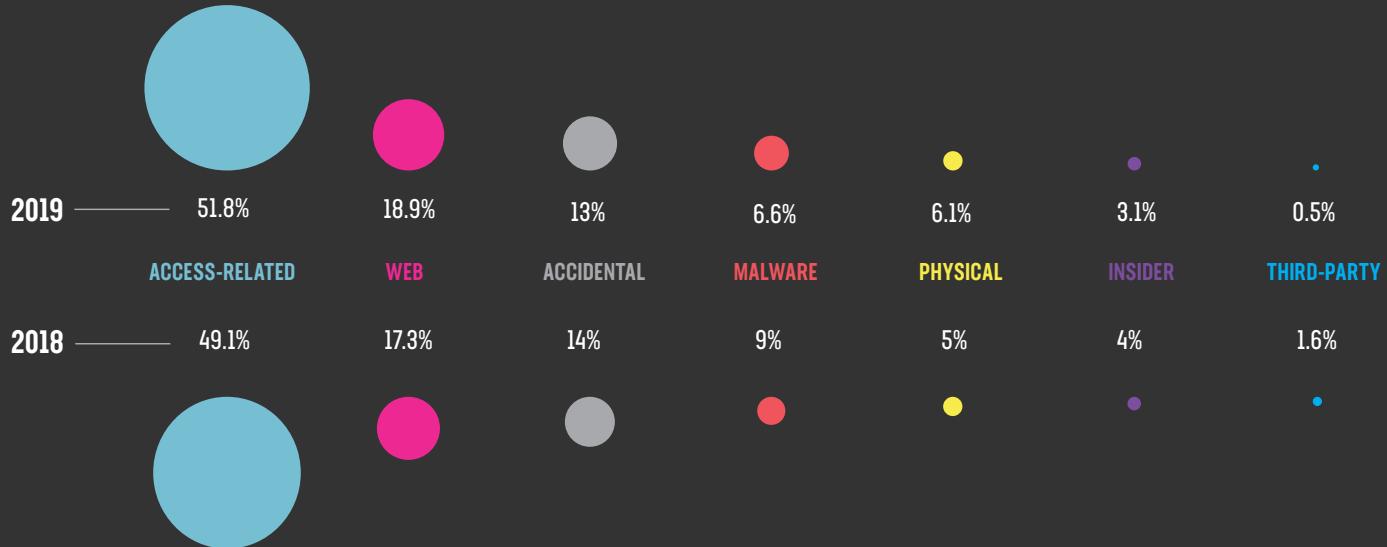
- Physical infiltration (mostly burglary and laptops stolen out of cars)
- Point-of-sale device attacks and the placing of physical skimmer devices

INSIDER BREACH CAUSES INCLUDED:

- Malicious data exfiltration
- Intentional misconfiguration or sabotage ([more on this](#))
- Insiders at trusted third parties that abuse their authorized access

FIGURE 4: 2018-2019 U.S. BREACHES BY CAUSE

Distribution of causes of U.S. breaches in 2018, by breach count. The lack of detail in the breach reports means that there is partial overlap between many of these categories.



MALWARE BREACH CAUSES INCLUDED:

- Any use of malware to manipulate or gain control of remote systems
- Ransomware attacks (which triggers a breach notification in some venues)

THIRD-PARTY HACKED refers to specific incidences where a cyber-related breach at a third party led to unauthorized access to organizational data.

PHISHING (NO OTHER DETAILS GIVEN) are cases where phishing was mentioned but it was not clear whether the phishing was used to obtain access credentials, or to drop malware.

We found that access-related breaches, at 47% in 2018 and 52% in 2019, constituted the largest proportion of known breach causes. The subcategories within access-related breaches have significant overlap, but all of them are reducible to either some form of social engineering (as in the case of phishing) or credential abuse. This is partly a reflection of the strength of other technical controls: if it were easier to circumvent authentication completely, we would not see so many attackers getting in this way. As it stands, real breaches show that, at present, humans and their access structures remain the weakest entry points.

FIGURE 5: U.S. BREACH METHODS BY SECTOR, 2018-2019



Industry Profiles

While the lack of details in the dataset prevent us from identifying more specific trends, we were able to group the breaches into two rough target/vector profiles that corresponded to the two most common breach causes identified above.

INDUSTRY PROFILE 1:

ORGANIZATIONS THAT ACCEPT PAYMENT CARDS ON THE WEB

One of the profiles was centered on a pattern of industries that tended to be compromised by payment form injection. The retail sector, which relies heavily on ecommerce transactions, had particularly high and growing rates of compromise by injection. The proportion of retail breaches that were injection attacks was 72% in 2018 and 82% in 2019. In 2018, there were a few other sectors, such as manufacturing and governments, that also tended to be breached this way.⁷ However, in 2019 the retail sector stood alone as a significant formjacking target.

In short, for organizations that accept credit cards for online payment, payment form injection attacks such as Magecart are a significant risk and specific controls must be put into place to prevent and monitor for these attacks.

INDUSTRY PROFILE 2:

ORGANIZATIONS WITH IDENTITY DATA USABLE FOR FRAUD

The other profile we identified centers on organizations that are significantly more likely to be compromised through phishing or illicit email access. In both 2018 and 2019, this pattern included sectors like accounting, finance, law firms, health, education, and non-profits. These sectors differed over time and between one another in the second and third most likely breach modes, but all sectors in this profile were disproportionately likely to experience a breach through phishing, credential abuse, or other kinds of email compromise.

In many of these cases, the breach notification letter mentioned how unauthorized parties found unencrypted personal information within the organization's email caches. Even though most security policies explicitly prohibit users from storing data of this nature in email boxes for exactly this reason, we are seeing that it is still common and still presents a risk.

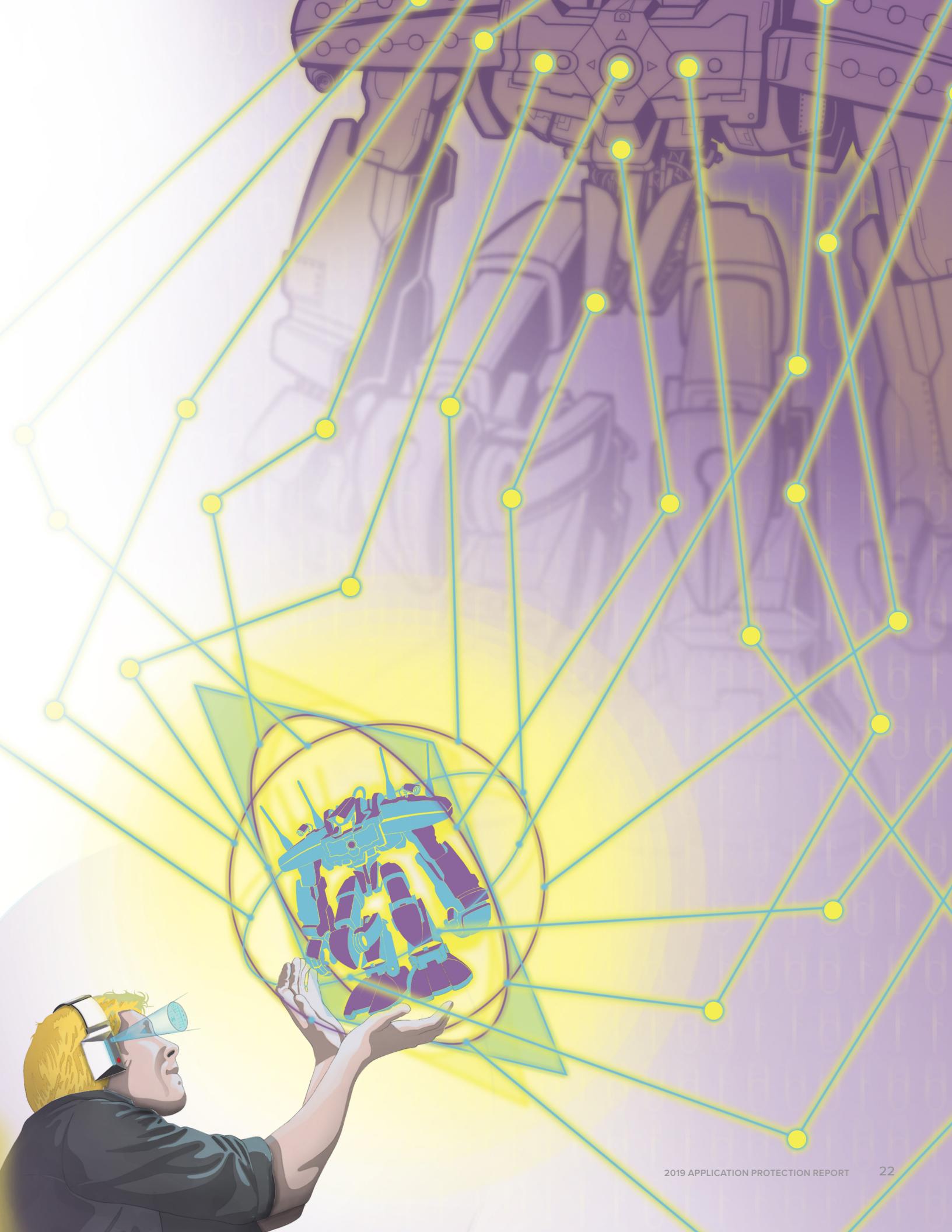
What Does All this Mean?

These trends make sense from the standpoint of how organizations in different sectors tend to store and transmit valuable assets. For sectors whose business models emphasize ecommerce, the ecommerce applications themselves represent a path to the goods that is relatively direct and unimpeded unless otherwise controlled. It is no surprise that the initial Magecart campaigns were directed against Magento storefronts. Magento runs on PHP, which, as we noted previously, presents rich targets to attackers.

By contrast, organizations in the second profile, coming from industries such as finance, healthcare, and education, can afford to store sensitive assets far from web front ends. Getting to the good stuff on such networks often entails complex, multi-stage attacks that require an initial foothold—which is almost always what phishing and email breaches provide. While it is certainly possible to find valuable information in email (and we saw some breaches happen this way in 2018), data exfiltration from human-structured data such as email is comparatively laborious and usually only worth the effort for small, value-dense data, such as intellectual property or political communications. For large-scale, profit-minded attacks, email is often just the first step in a broader campaign to reap valuable personal information.

FOR ALL THEIR LIMITATIONS, PUBLIC BREACH REPORTS SUCH AS THESE ARE UNIQUELY VALUABLE IN THAT THEY ARE LEGALLY VETTED, VERIFIED DATA POINTS ABOUT SUCCESSFUL ATTACKS.

For all their limitations, public breach reports such as these are uniquely valuable in that they are legally vetted, verified data points about successful attacks. In a world in which unambiguous information about other organizations' security is rare, they allow us to sort out both the rudimentary attacks that most mature organizations should expect to withstand and the leading-edge advanced tactics that are normally reserved for nation-states. They enable us to focus instead on the threats that are most likely to hit "regular" organizations. With that said, we will now explore these tactics—*injection* and *access* attacks—in greater detail to understand what you can do to stop them.





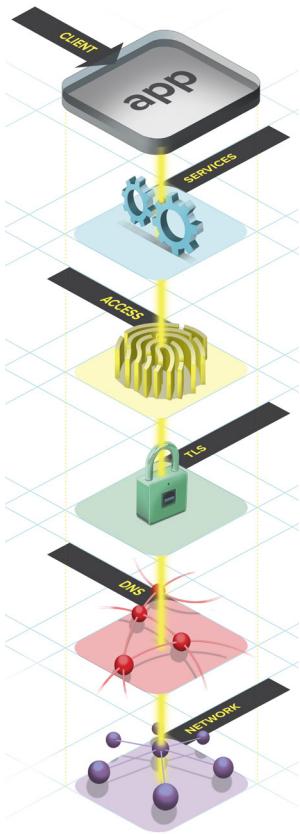
Attackers target third-party app services to come in sideways and steal form submissions, the way bank robbers tunnel into a vault from an adjacent storefront.

EPISODE 3

Injection for a Decentralized Age

Injection has taken on new significance as a result of accelerating trends in how websites are constructed. We will unpack these trends, link them to new attack data, and offer tactics for mitigating this new-old risk. But first, to place it all in context, we need to revisit some big-picture questions.

Again, We Ask: What Is an App?



As we pointed out in the introduction, to a user, an application may appear to be a single thing: a program running on a server somewhere with a user interface. In reality, most applications on the Internet are swarms of microservices and sub-applications all converging at the last minute into a coherent user experience.

These embedded services can include things like user analytics, chat features, debugging tools, social media sharing capabilities, advertising, and enhanced animation. Increasingly, these microservices are being linked to and run from external third-party sites. In other words, the active web code is running on a server that has nothing to do with the “primary” application.

While the pace of this trend is accelerating, microservices and sub-applications are not actually a new thing. At the 2014 RSA Conference keynote, noted security researcher Dan Geer said, “While writing this, the top level page from cnn.com had 400 out-references to 85 unique domains, each of which is likely to be similarly constructed and all of which move data one way or another.”⁸ By 2016, researchers at University of Illinois, Urbana Champagne found that 64% of website resources were loaded from an external domain.⁹ In 2019, a simple scan of the F5 main web page reveals 70 external domain references.¹⁰

This tendency to assemble distributed content and functions at the browser is, in itself, not a bad thing. It saves server bandwidth and processing while freeing developers from having to reinvent wheels. It also, however, broadens the attack surface for that old and venerable vulnerability classification, injection. For that reason, we will revisit what makes injection such a durable flaw, and why its latest incarnation makes third-party content such a significant problem.

Injection Still?

Traditionally, when tech people hear about web injection, they think of SQL injection, which is a common form, but is far from the whole picture. Injection is a broad category of attacks that can manifest differently depending on the context. These attacks occur when an attacker secretly adds, or *injects*, their own instructions into an existing authorized application execution process. Injection can sit almost anywhere in an attack chain, from the initial

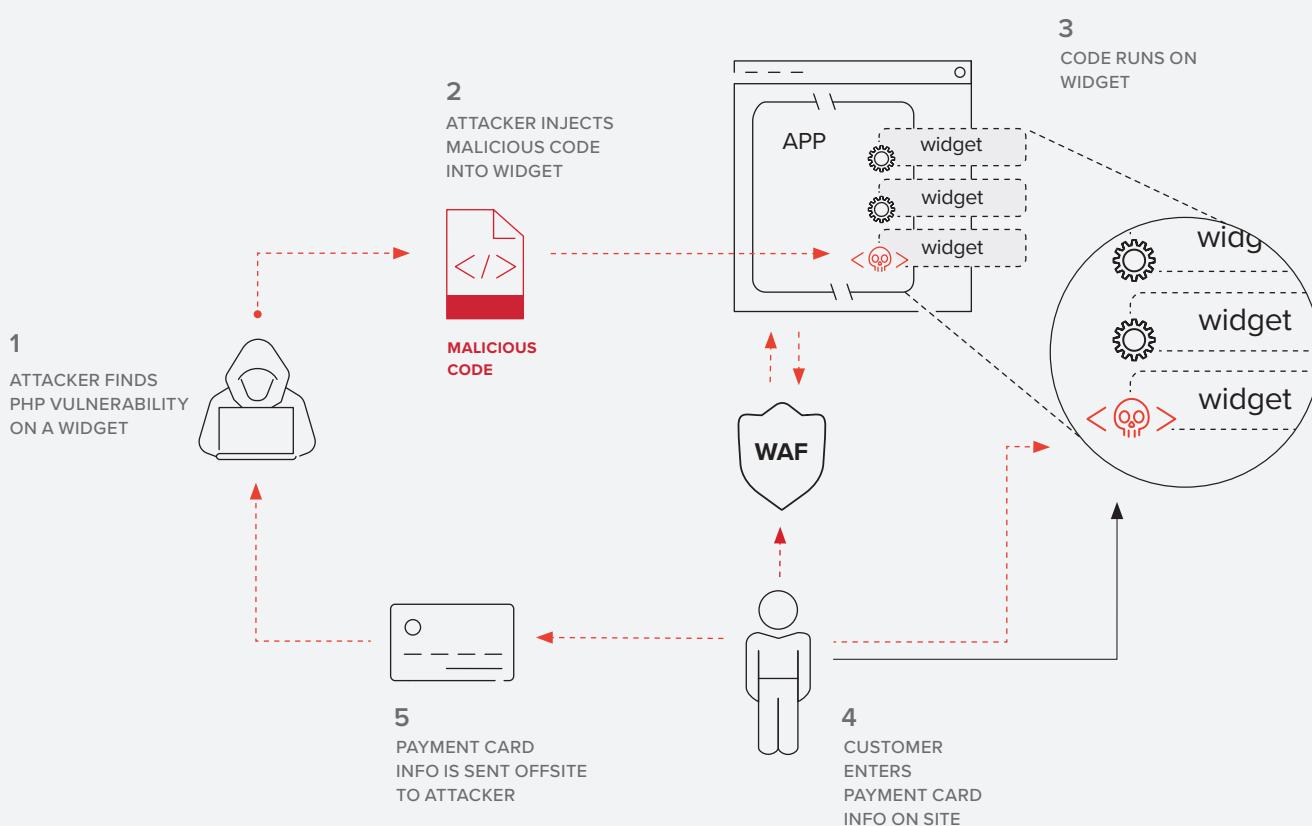
contact with the target to the final exploit.¹¹ It is often used as a way to circumvent authentication,¹² but there are also many injection exploits that depend on the attacker already being authenticated.¹³ The insertion point for the payload can be at the server level,¹⁴ the web application itself,¹⁵ or, as we are seeing more and more, in third-party application modules or content (more on this below).¹⁶ In some cases, the injected code resides in a Content Delivery Network (CDN).¹⁷ It also is feasible to do a man-in-the-middle injection into a CDN.¹⁸

The goal of the injection payload can vary widely, as well. Some injection exploits allow attackers to execute commands within the operating system (OS).¹⁹ Some are used to circumvent authentication, as we mentioned earlier.²⁰ Some inject code, such as malware, or other objects, including code for escalation of privilege.²¹

A subtype of injection vulnerability that warrants a specific mention is file inclusion, in which an attacker injects a path to malicious files, either local or remote, that the server processes at runtime.²² In these cases, only a few lines of code need to be injected since the majority of the exploit runs away from the victimized server. And, while we generally treat cross-site scripting (XSS) vulnerabilities as separate because they exist in a specific context, they are also a form of injection in which the injection payload is a step to compromising another visitor to the site, not the site itself.

FIGURE 6
COMMON INJECTION
ATTACK PATH

The path that both malicious code and valuable financial information take during an injection attack, now that third-party services are such a common component for web applications.



However, the most impactful form of injection at the moment is known as formjacking. A formjacking attack siphons information that users put into online forms such as login screens or payment forms and delivers it to a location under the attacker's control. Most of the time the information sought by attackers is login credentials or financial information such as payment card numbers.

Formjacking is not a new type of attack, but it has exploded in popularity over the last two years, primarily in the form of Magecart attacks. The name Magecart originally stems from a name collectively assigned to the different threat actor groups who carried out the initial exploits of a shopping cart vulnerability on the Magento ecommerce platform (Magento + Shopping Cart = Magecart).²³ These exploits, which vary subtly over time and from group to group, mostly rely on a vulnerability in Magento, which runs on PHP. The vulnerability itself boils down to an inherent flaw in PHP's unserialize() function that allows attackers to inject PHP objects, including executing arbitrary PHP code for formjacking.²⁴ Although formjacking is not limited to PHP systems, PHP is highly targeted by attackers, and formjacking remains one of their preferred tactics.

71% IN 2019, FORMJACKING PAYMENT CARDS WAS RESPONSIBLE FOR 71% OF WEB BREACHES AND 12% OF KNOWN BREACHES IN TOTAL.

In 2018 we reported that of the analyzed breaches that were based on web attacks (as opposed to phishing or other vectors), 70% were formjacking, or injections of malicious code on websites for payment card theft. We predicted that, because of their profitability and difficulty of detection, formjacking attacks against ecommerce platforms would grow, and so it has proven. In 2018, formjacking was responsible for 71% of web breaches and 12% of known breaches in total. In 2019 they became even more prevalent, making up 87% of web breaches and 17% of total breaches.

New Software Supply Chain, Old Attacks

As alarming as these numbers are for our industry, they only hint at a deeper problem. As webpages pull content from increasingly disparate and nebulous sources, we're seeing more content getting injected in the browser from third-party add-ons. These exploited third-party tools run in the same computing context as the main web application and often request sensitive content, such as credit card numbers in payment input fields. Compared with other types of injection, this has some significant advantages for attackers, which are illustrated by the 2017 breach of [24]7.ai, a customer experience software and services company.

In autumn 2017, a number of large enterprises, including Delta Airlines, Sears, Kmart, and Best Buy, experienced coordinated breaches of customer financial information. The total number

of exposed records is thought to number in the hundreds of thousands. The source of the breach was an unspecified piece of malware that was delivered to the target sites through a compromised customer service chatbot module created by [24]7.ai, one of the industry leaders in customer service automation tools.

Even though this attack is outside of the timeframe of this report, it encapsulates many of the factors that make it difficult to prevent formjacking today:

VISIBILITY

Standard web application firewalls (WAFs) protect the primary site by examining traffic between the client and the app server. Third-party scripts, however, are loaded directly by the client browser, completely bypassing perimeter security controls. The WAF may see that a script, such as an advertisement loaded from an ad network, is included as part of the app page, but it will not see the contents of that script. Traditional security tooling will view it as completely legitimate. Furthermore, sites that deliver malware or receive skimmed financial information tend to have legitimate encryption certificates on look-alike domains. In the case of [24]7.ai, the target enterprises almost certainly had no indication from their own network monitoring surfaces. We know that they were notified by [24]7.ai, the third-party service provider, about the breach, not the other way around.

TARGETING

Because more web applications are outsourcing critical functions such as ecommerce, the vendors become an outsized target, specifically because their code is called by such a wide range of customers. Attackers know that if they compromise a single vendor for a microservice, they stand to skim data from a huge pool of potential victims, often across multiple industries. This acts as a force multiplier for attackers.

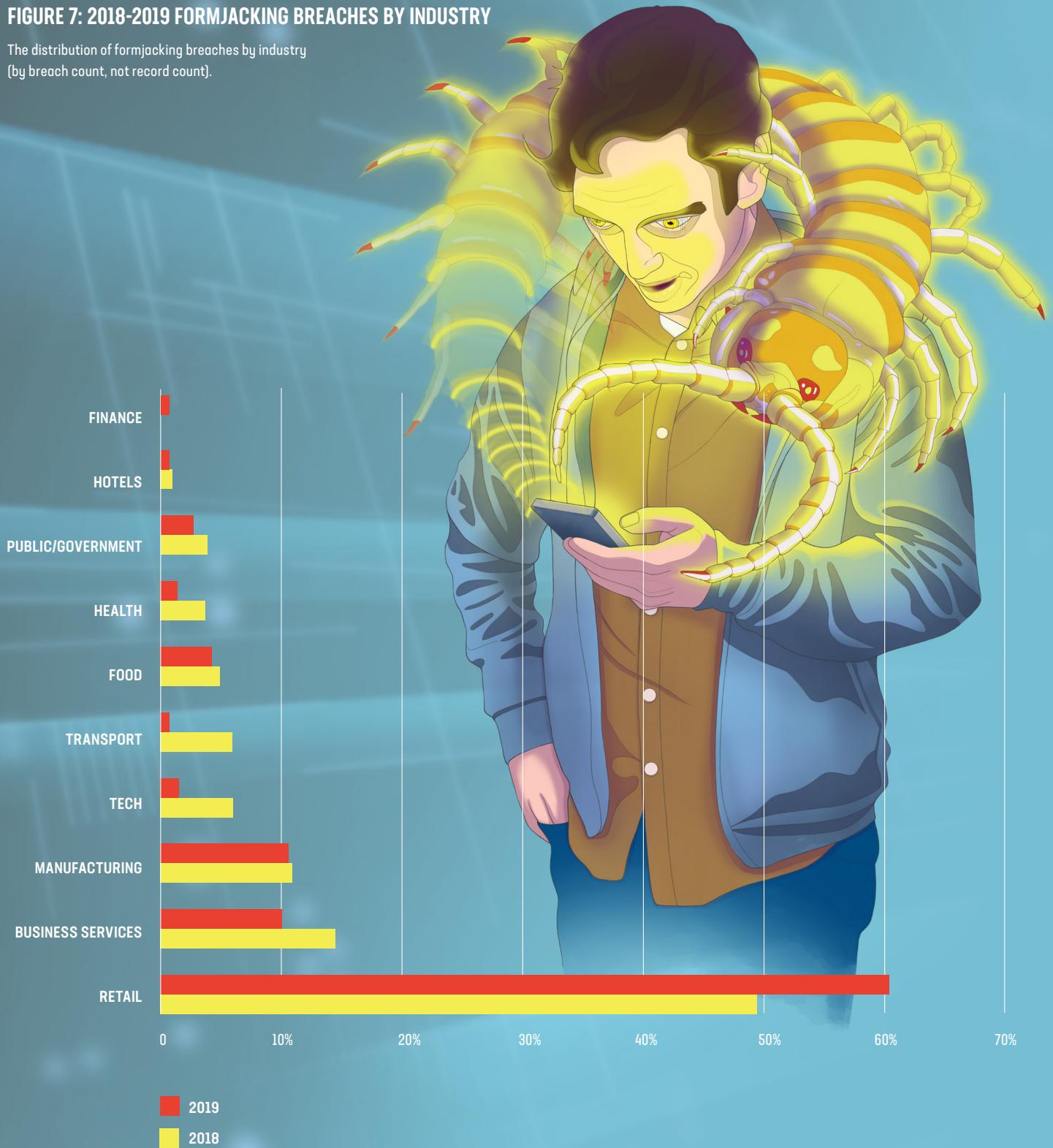
What the Data Say

The 2018 breaches that we examined revealed 83 breaches attributable to formjacking attacks on web payment forms. In terms of number of breaches, nearly half of these came from the retail industry. However, in terms of number of stolen records, the transportation industry, including companies like Delta, British Airways, and Amtrak, was responsible for 60% of the stolen payment cards. In 2019, of the 145 formjacking breaches, 61% targeted retail, with manufacturing and services taking most of the rest.

The lesson is clear: for any organization that accepts payment cards via the web, their shopping cart is a target for cyber-criminals. Furthermore, formjacking has emerged as a reliable technique for extracting high value data like financial information specifically because of the decentralization trend in web content and services.

FIGURE 7: 2018-2019 FORMJACKING BREACHES BY INDUSTRY

The distribution of formjacking breaches by industry
(by breach count, not record count).



The lesson is clear: for any organization that accepts payment cards via the web, their shopping cart is a target for cyber-criminals.

Mitigating the Risk of Injection Attacks

We hope it's clear by now that injection is a tricky devil. While no checklist will ever be enough on its own to control against such a mutable attack type, the suggestions that follow will get you off on the right foot.

INJECTION DETECTION

Injection vulnerabilities can be detected during development but are more difficult to detect in deployed systems. Because injection flaws can be exploited in any stage of an attack, finding and evaluating their impact depends on context. Often attackers use lower-priority vulnerabilities such as cross-site scripting (XSS) to gain an initial foothold to inject malicious JavaScript into a website.²⁵ In other cases, attackers can inject PHP commands into an [application programming interface \(API\)](#) or server-side applet, as in the case with Magento and the Magecart campaign.²⁶

As we mentioned previously, the risk of these kinds of attacks is magnified when the target web application uses third-party code running offsite. It is more difficult to detect changes to third-party code, harder to whitelist source IP addresses for content, and given the growth in malicious use of encryption, harder to inspect traffic.

TAMPERING OF SCRIPTS HOSTED DIRECTLY WITHIN A BUCKET WOULD LEAD TO COMPROMISE OF EVERY SITE THAT LINKED TO THAT SCRIPT.

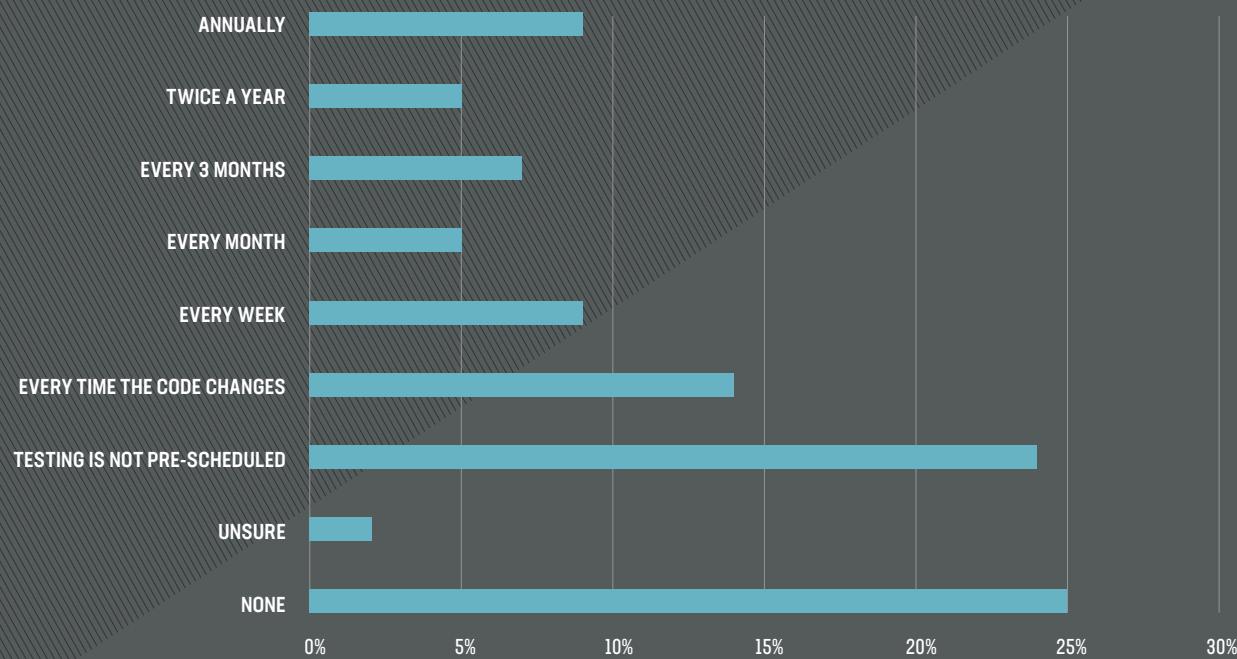
Many sites, even high-profile ones that receive a huge amount of traffic, link to scripts posted directly in Amazon S3 buckets or other cloud storage solutions. With the huge number of S3 buckets that are [deliberately configured](#) to have [weak or no authentication](#), this poses another significant risk: tampering of scripts hosted directly within a bucket would lead to compromise of every site that linked to that script. This is a similar campaign strategy to the one we explored above with the [24]7.ai campaign. The growing use of third-party content also means that attackers can exploit vulnerabilities across their targets' customer lists to achieve a huge impact overnight.

INVENTORY

As always, [a proper inventory is a cornerstone of managing risk](#). Conducting an inventory of web applications in your environment with a specific focus on auditing for third-party content will tell you about your supply chain attack surface (at least with respect to software). This, however, can be extremely complex when the providers of our script libraries, advertising, and our resources will themselves link to yet more third parties. Also, consider that some of these third parties, such as web widgets or user trackers, will have a lower security stance than your average ecommerce site, which must meet PCI DSS standards.

FIGURE 8: FREQUENCY OF VULNERABILITY SCANNING

Frequency of app vulnerability scanning reported by over 3,000 IT security professionals surveyed.



PATCHING

Patching your environment is also a critical part of managing risk as things change. While patching won't necessarily fix the flaws in third-party content that present the newest form of risk, it will make it harder to escalate from an initial foothold into a substantive compromise. Since injection is such a versatile technique, patching applications running in your own environment is still critical to preventing escalation from a compromised third-party asset.

SCANNING

Similarly, vulnerability scanning not only remains important, but takes on a new dimension. Many CISOs have recognized for years that it is important to run external scans in addition to internal ones to get the "hacker's eye view."²⁷ The fact that so much content is now being assembled at the client side makes this even more important.

CHANGE CONTROL

Monitoring for code changes on the site, regardless of where that code is hosted, will provide an added degree of visibility irrespective of whether new vulnerabilities are emerging. This means monitoring GitHub and AWS S3 buckets as well as native code repositories.

MULTIFACTOR AUTHENTICATION

Given that injection is so often used to bypass authentication to gain access to web server code, multifactor authentication should be implemented on any system that can connect to high-impact assets. Ideally, application-layer encryption can also supplement TLS/SSL to maintain confidentiality at the browser level. Many well-known web application firewall (WAF) products have this capability.

WEB APPLICATION FIREWALLS

More broadly, modern WAFs will provide greater control over who connects to your systems, how they connect, and how their user input is protected. While technology is rarely a simple solution, and a firewall is only as good as the team that sets it up, modern WAFs offer a level of application-layer visibility and control that can help mitigate the distributed and polymorphic risk that injection presents.

SERVER TOOLS

There are also a number of server software tools at your disposal. You can set up a Content Security Policy (CSP) to block unauthorized code injections into a website or application.²⁸ On top of that, you can add Subresource Integrity (SRI) web methods to verify that those third-party apps have not been altered.²⁹ Both of these tools require some work to properly fit to a web application.³⁰ This is where a good, flexible WAF can help.

MONITORING

Monitor for newly registered domains and certificates that include your brand name. These are often used to host malicious scripts while appearing genuine to end users.³¹

FORMJACKING ILLUSTRATES THE WHACK-A-MOLE PRINCIPLE

Despite fundamental changes in the structure of web applications, significant progress in security tooling, and the fact that it is a well-understood type of attack, injection has endured as a go-to vulnerability type for attackers for decades. On one level, this is due to its flexibility, for which the rise of formjacking is a timely reminder. On another level, however, its persistence is partly due to the fact that it exploits interactions between systems and people, and controlling people is much more difficult than controlling systems.

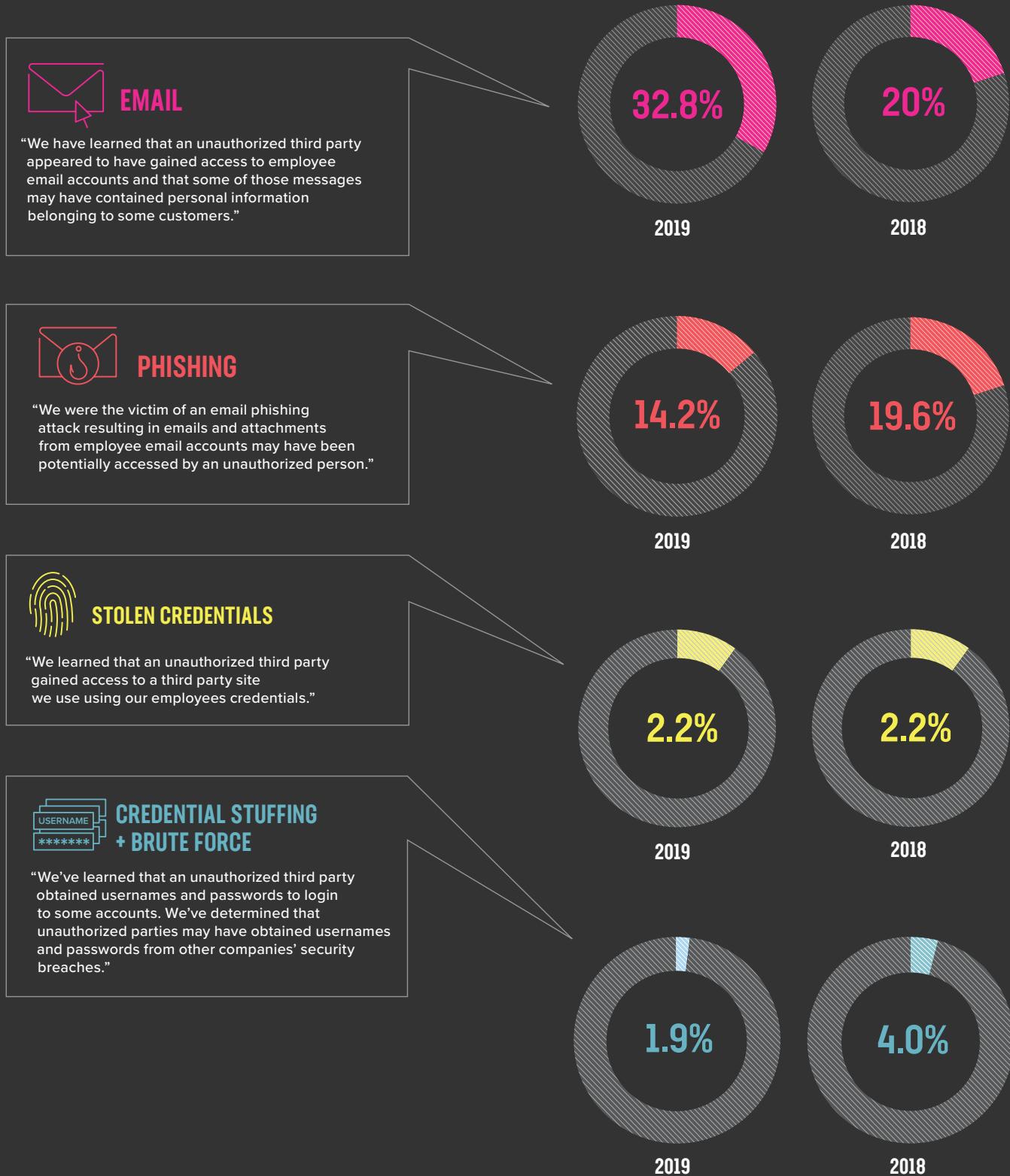
As long as user input is part of how a system generates value, there will always be an affordance for injection. This, in turn, brings us to the question of human users, which were the biggest single source of breaches in 2018 and 2019.



Formjacking's persistence is partly due to the fact that it exploits interactions between systems and people. Controlling people is much more difficult than controlling systems.

FIGURE 9: 2018 AND 2019 ACCESS BREACHES BY CAUSE-U.S.

The sample text from breach notification letters for each category shows the overlap between categories and the difficulty doing of root cause.



EPISODE 4

Access Attacks' Enduring Prevalence

Access tier attacks are any that seek to circumvent the legitimate processes of authentication and authorization that we use to control who gets to use an application, and how they can use it. The result of this kind of attack is a malicious actor gaining entry to a system while impersonating a legitimate user. They then use the legitimate user's authorization to accomplish a malicious goal—usually data exfiltration. Attacks of this nature can be hard to spot because as far as the system is concerned, the attacker appears to be the rightful user. We can further break down the broader category of access attacks into subtypes, as shown in Figure 8.

However, as we noted earlier, because these data come from legal documents that often lack detail, many of these categories have significant or total overlap, and it is difficult to be sure what part of an attack chain these access attacks occupied. Figure 9 shows samples of text from various breach letters that we used to sort these into categories.

We know that according to the data available to us, access attacks represent the most likely vector into an application. This is particularly true in industries whose business models require storing valuable information far from their perimeters. To understand why, we need to explore the various forms of access attacks we saw in 2018-19, how they work, and who uses them.

Phishing

Phishing is a form of social engineering in which attackers use email or another form of electronic communication to impersonate an entity whom the victim trusts. Phishing is frequently the first aggressive action an attacker takes as part of a longer effort.

Despite the fact that phishing is a low-tech attack type that is already fairly well known in the mainstream, we can see from the data that it is still gaining steam. This is partly because it is a reliable strategy for attackers across the spectrum of training and sophistication. It works whether the goal is to steal financial information, intellectual property, or military intelligence.

Other than inducing a victim to enter credentials into a spoofed authentication portal, the most common objective of a phishing campaign is to install malware on the victim's computer. From here an attacker can capture credentials, escalate privileges, look for other valuable forms of information, pursue a cryptojacking or ransomware campaign, or use the target machine as part of a botnet.

Credential Stuffing and Brute Force Attacks

In addition to phishing, we are seeing access attacks take less surgical forms. In these cases, attackers either try known passwords from stolen databases of credentials or enter passwords that are known to be common. This might look like a rudimentary tactic, but the prevalence of password reuse, and the large amount of personal information already breached and available for sale on the Internet, make these tactics valuable in spite of their simplicity.

CREDENTIAL STUFFING

Credential stuffing is the practice of trying passwords that attackers already know the victim uses elsewhere, with the expectation that the victim uses the same password in multiple places. These attacks are surprisingly successful as a result of two linked trends.

The first is that as more organizations move to the Internet, the number of credential pairs we need to remember has grown tremendously. After all of the data breaches over the last few years, it is highly likely that any given individual in the U.S. or Western Europe has had at least one set of credentials leaked online. The result is that we've given attackers both the motive and the tools necessary to try to stuff known credentials into authentication portals.

Because attackers recognize that many organizations have monitoring in place to detect many rapid authentication attempts, credential stuffing attacks tend to fall into one of two patterns. Attackers either try to log in to a large number of accounts using a small number of known passwords, or a small number of accounts using a large number of passwords.

Over the years, most organizations have implemented some kind of password rotation policy with the intent of reducing the risk that a leaked password will still be valid. This has had some unintended consequences, with many people using the same pattern of characters and something memorable. For instance, if policy requires a password change every 90 days, chances are that someone in the target organization is going to use something cued to the seasons, such as Spring19 or Summer19.³²

BRUTE FORCE

We typically define brute force attacks as either ten or more successive failed attempts to log in less than a minute, or 100 or more failed attempts in one 24-hour period. However, attackers realize that these kinds of behaviors are easily monitored and so have begun to alter behavior.

One of the biggest threat intelligence sources we have for brute force attacks comes from our own F5 Security Incident Response Team (SIRT). The SIRT reported that in 2018, brute force attacks against F5 customers were the second most frequent type that they encountered, and they constituted 19% of the incidents they addressed.

The F5 SIRT reported that in 2018, brute force attacks against F5 customers were the second most frequent type that they encountered.



Depending on how robust your monitoring capabilities are, brute force attacks can appear innocuous, like a legitimate login with correct username and password. Detecting these attacks hinges on both capturing and examining detailed logs. The number of attempts, the location of the login, and the time of day can be clues to anomalous behavior even if the attacker happened upon the right credentials early on. We'll dive into monitoring more in the mitigation section below.

BRUTE FORCE DATA

F5's SIRT noted lots of brute force attacks in 2018.³³ As we mentioned earlier, nearly a fifth of all confirmed attacks take this form, and 90% of customer support calls to the F5 SIRT were about brute force attacks. While the SIRT also noted a low success rate, even failed brute force attacks can affect an organization's environment. On six separate occasions, the SIRT found that brute force attacks had actually caused the target's entire authentication infrastructure to go down. Even when the servers stayed up, authentication for legitimate users locked out or bogged down, resulting in an indirect denial-of-service attack.

Unfortunately, these kinds of secondary effects were sometimes the only way that organizations even knew they were under attack. Many organizations go weeks, months, or years without looking at log data, which is where the context necessary to identify this kind of attack resides. We regret to say that this is all too common in the security industry, with the widespread implementation of low-cost basics taking a backseat in favor of high-tech, expensive solutions that are slick but narrow in scope.

For those interested in more detail on those brute force attacks, Figures 11 and 13 break them down by industry and region. Keep in mind that these data points represent only current F5 customers who called the F5 SIRT for response help. They do not represent the industry as a whole but, given the size and scope of F5's installed base, they do give a clue as to general brute force attack trends.

Email Hacks

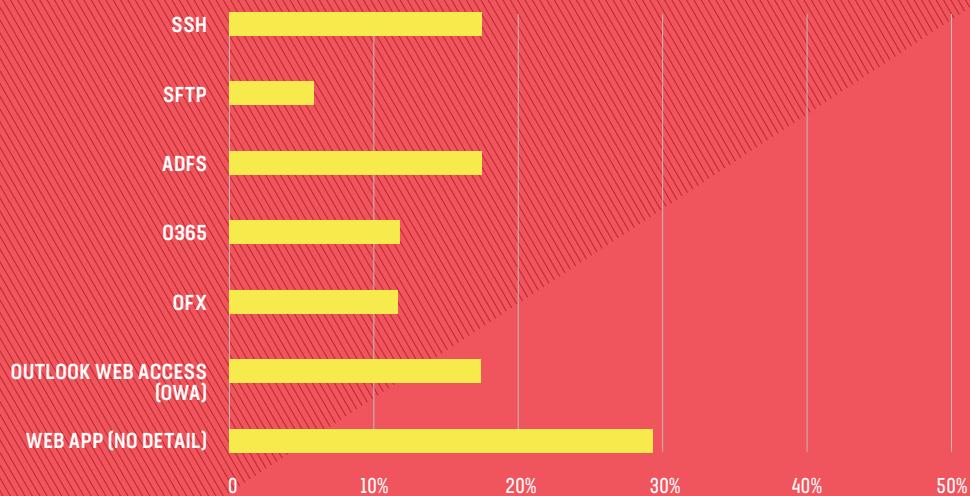
We mentioned earlier that 20% of the confirmed breaches in 2018, and 33% in 2019, started by targeting email access. When attackers go after organizations this way, email is often both a useful staging ground to get to valuable information elsewhere, and a source of valuable information in itself.

The breach data also featured email as a primary target. Email was involved in the top two subcategories of access breaches, representing 39% of all breach causes in 2018 and 47% in 2019. Think about that: email is directly attributed as a factor in nearly half of all breach reports with a cause. A typical breach notification letter goes something like "Unauthorized persons used stolen credentials to gain access to emails containing confidential records...."

FIGURE 10: 2018 BRUTE FORCE ATTACKS BY PROTOCOL/SERVICE

(as a percentage of 2018 F5 SIRT incidents)

Brute force attacks mitigated by the F5 SIRT, broken down by protocol/service.

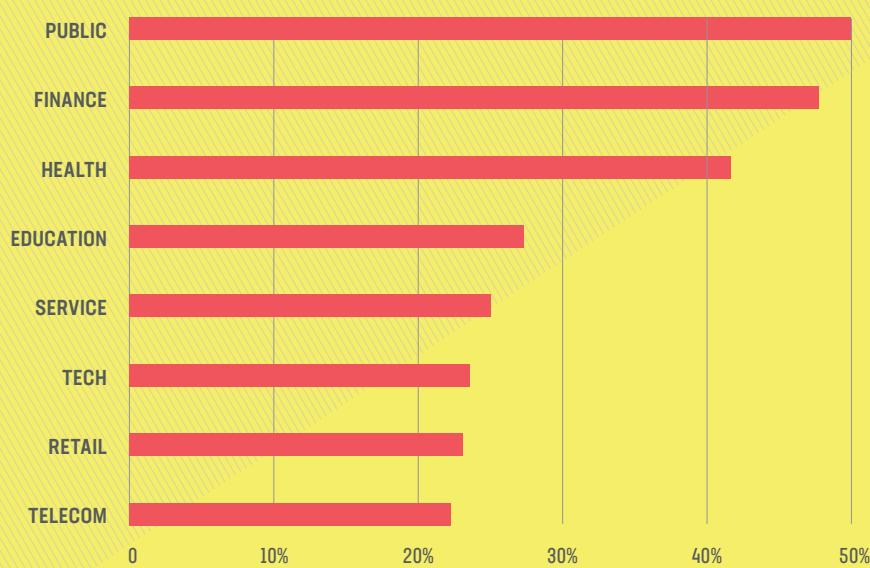


29%

AT 29%, ATTACKS AGAINST
WEB APPLICATIONS
SURPASSED ALL OTHER
SERVICES.

FIGURE 11: 2018 BRUTE FORCE ATTACKS BY INDUSTRY

F5 SIRT brute force attacks by industry, as a percentage of reported 2018 SIRT incidents.



19%

BRUTE FORCE ATTACKS AGAINST
F5 CUSTOMERS WERE THE
SECOND MOST FREQUENT
TYPE, CONSTITUTING 19% OF
ADDRESSED INCIDENTS.

By accident or oversight, organizations are still storing unencrypted medical and financial data in weakly protected email boxes, and show no sign of change.

While the transport layer for email is usually not secured, and it is feasible to sniff email in transit, the chances of finding anything valuable in a single message is low. The more fruitful attack against email is focused on where the mail lands, because mail storage often persists in perpetuity. Email boxes are filled with gigabytes of easily searchable information. In addition to sensitive information that might be sitting there, contact lists provide more targets for future attacks. Users often forward or redistribute email messages as well, so it is difficult to say for sure where any sensitive information is, once it has been transmitted through email.

Mailboxes are not a good long-term storage option for private information. Large-volume, unencrypted mailboxes can be an unexpected magnet for lawsuits, as they often contain information that is equal or greater in value to assets that are stored under much greater control, such as databases of customer information.

Mitigating the Risk of Access Attacks

So, how do you reduce the risk that access attacks pose? We'd love to say "just MFA it" and drop the mic, but we realize that multi-factor authentication can be hard to implement and is not always feasible in the timeframes we'd like. As much as passwords are flimsy protection, we found in 2018's report that 75% of organizations still used simple username/password credentials for critical web applications, so we can't just pretend that multi-factor is the standard.

To start, make sure your system can at least detect brute force attacks. Setting up alarms is a good start, but it's better to slow down the session by throttling or using CAPTCHA, or even blacklisting the IP address. However, one of the things that makes access such a tricky tier to work on is that confidentiality and integrity can sometimes find themselves at odds with availability. Locking the account in perpetuity is good for protecting unauthorized access, but also results in denial of service for the user. If you're going to lock someone out, make sure you can fail gracefully, and look out for the bane of the false positive. Set up reset mechanisms that work for both you and your users to get the legitimate traffic back online as quickly as possible.

In other words, it's not enough to set up some firewall alarms on brute force attempts and take a nap. You have to test these monitoring and response controls, run incident response scenario tests, and develop incident response playbooks so that you can react quickly and reliably.

The NIST Digital Authentication Guidelines offer principles that represent a good baseline and get away from some well-intentioned but obsolete ideas about access control:³⁴

FIGURE 12: BRUTE FORCE ATTACKS AS PROPORTION OF ALL ATTACKS OVER TIME

AS A PERCENTAGE OF 2018 F5 SIRT INCIDENTS

SIRT brute force attacks as a percentage of reported 2018 SIRT incidents.

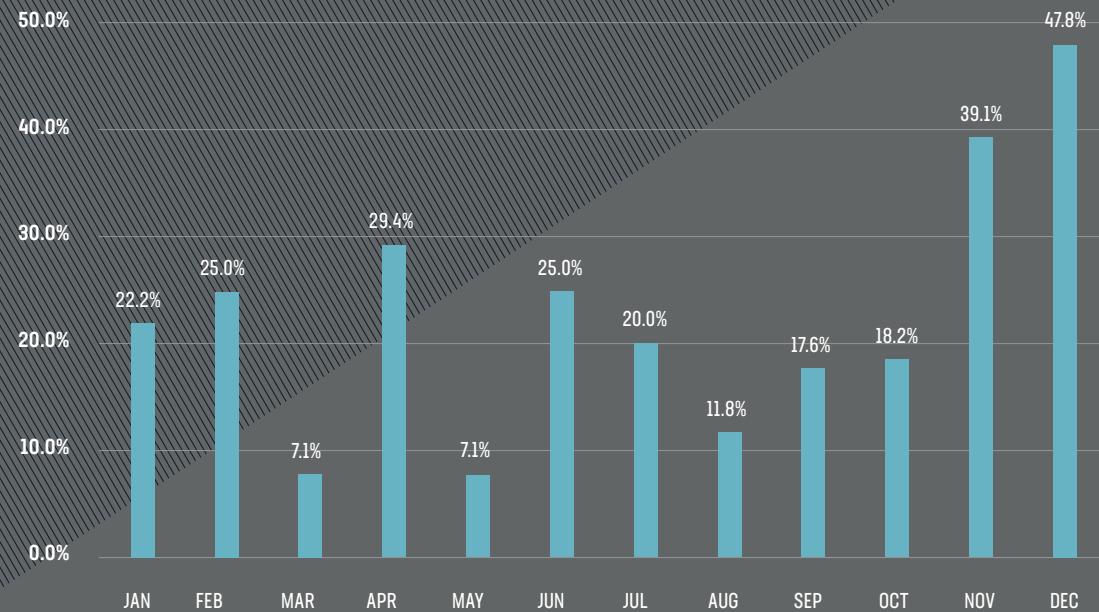


FIGURE 13: BRUTE FORCE ATTACKS BY REGION

AS A PERCENTAGE OF 2018 F5 SIRT INCIDENTS

SIRT brute force attacks by region from reported 2018 SIRT incidents.



43.5%

BRUTE FORCE ATTACKS WERE
AGAINST F5 CUSTOMERS BASED
IN EUROPE, THE MIDDLE EAST,
AND AFRICA.



Incident response should include a streamlined and guiltless method for users to report suspected phishing. Users should feel no shame in asking about or reporting a phish so you can catch and contain them quickly.

- Make your password policies user friendly
- Check passwords against a dictionary of default, stolen, and well-known passwords, both when users choose a password, and on a recurring basis
- Password reset should never use hints
- Use long passwords
- Avoid arbitrary 30/45/60/90-day password rotations
- Lock or remove unnecessary credentials

At a more advanced level, authentication can turn into a continuous practice instead of a one-time check. We don't want to make users re-enter a password every time they act on a system, such as accessing or changing data. Such a thing would be about as user-unfriendly as we can get. However, there are backend authentication tools, like cookies and session tokens, that can be used to reduce the attack surface, prevent escalation of privilege and network traversal, and effectively function as a sort of digital quarantine.

Some cloud providers have suspicious activity alert capability for their customer accounts. Specifically, Microsoft Azure has a mechanism to flag and block the use of known bad passwords in Active Directory cloud deployments.³⁵

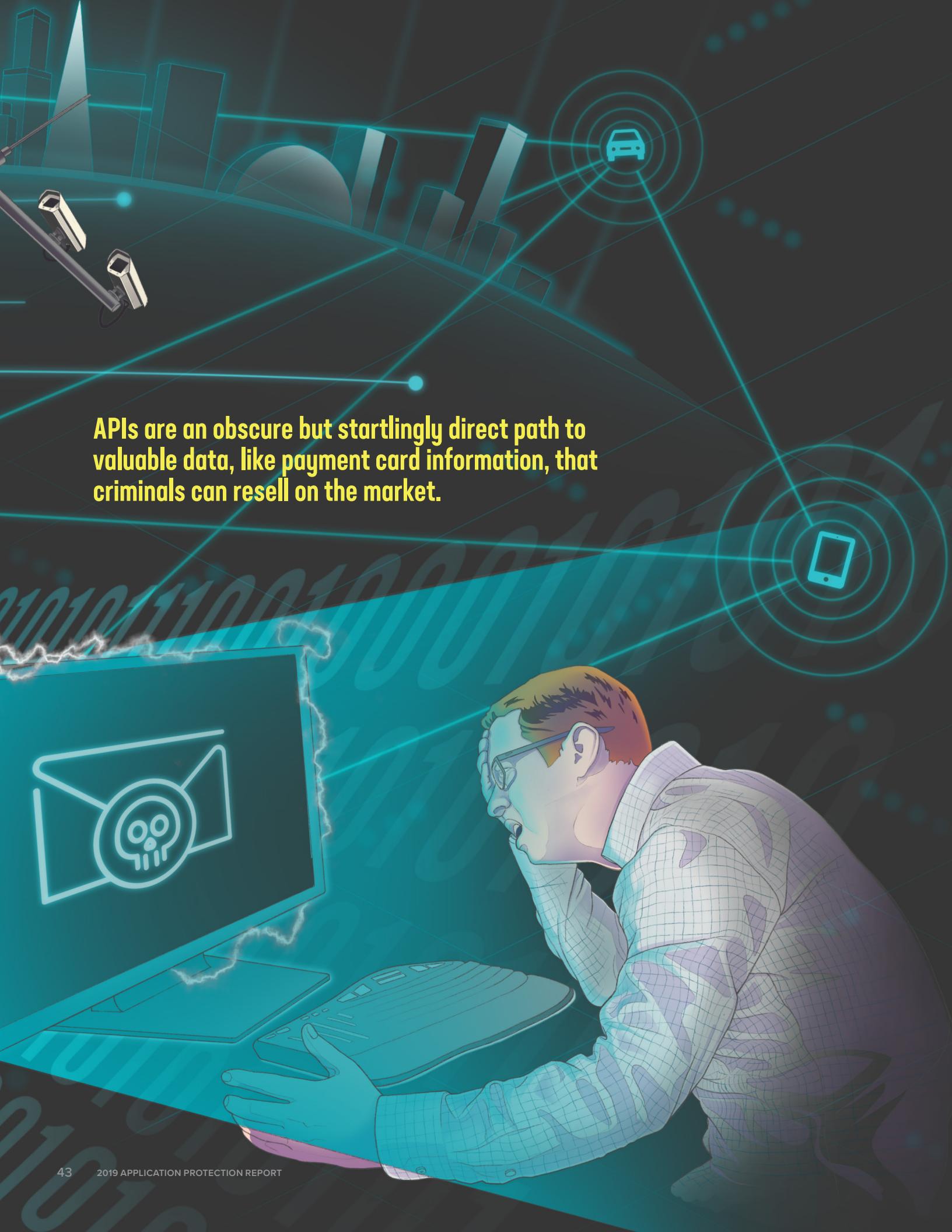
The same accidental denial-of-service issues we outlined previously apply, especially to email, so controlling risk around email attacks is tricky. Make sure you monitor load on your authentication infrastructure using threshold alarms.

As part of an assume breach approach, plan for an attacker to gain access to email, and gear your forensics accordingly. Assume that attackers will set up email forwarding and account delegation on a stolen mailbox, and the user may not even know it. Write up procedures on how to review this and make them part of the incident response plan.

When setting up logging, check what level of detail your email system provides. Can you recreate an entire email session with log data? Could you tell what settings the attacker might have changed? Can you tell exactly what they downloaded or forwarded? This will figure prominently in your breach reporting. Set the log settings and test them by logging in and see what actually appears in the logs. In the event of an incident, these logs may be your lifeline, so plan and test accordingly.

Incident response should include a streamlined and guiltless method for users to report suspected phishing. Users should feel no shame in asking about or reporting a phish so you can catch and contain them quickly.

Web mail authentication sessions can remain active for hours after changing a password. Test the timing and verify the procedures for this to add to your incident response procedures.



APIs are an obscure but startlingly direct path to valuable data, like payment card information, that criminals can resell on the market.

EPISODE 5

APIs and the New-Old Problem of Visibility

The growing importance of application programming interfaces (APIs) has dramatically changed the threat landscape in a short time. In the simplest possible terms, an application programming interface (API) is a user interface for apps instead of users. It creates a connection point through which other app services or mobile apps can push or pull data without human input. While APIs have been around for a long time, the growth of serverless architectures, containers, and DevOps have made them critical for contemporary web operations. You might say that web applications have shifted from being blankets to being quilts, and APIs are the stitching that holds the quilts together.

APIs are hard to categorize. They serve a combined access control and data translation role, coordinating disparate and distributed functions behind the scenes to present the user with a unified application service. From a security standpoint, this is what really matters: the user experience is that of one app. In other words, APIs raise the same fundamental issue of visibility that we discussed earlier in the section about injection attacks. They can be compromised in many of the same ways that web application can be breached. But because APIs are increasingly important and also hidden from view, they arguably represent a bigger risk to the business than other assets.³⁶

API Usage

One of the reasons that APIs have become so common is that they are a relatively simple component that can transform an organization's business model by open up new revenue streams. In the days of monolithic apps, whatever core business logic generated value needed to be supported by a user interface, storage, and other meta-functions. Now it is sufficient to develop a single specialized service, and use APIs to either outsource other functions to bring an app to market, offer the service to other app owners, or both. Google's Earth Engine, the core service that underpins Google Earth, is a good example. While we can use that service through the main Google Earth app, many other apps use the Earth Engine backend with their own specialized UIs, including apps that visualize habitat destruction and others that project an apocalyptic zombie wasteland onto our present.³⁷ Mobile apps connect to the back-end through APIs as well, even when connecting to their own organizations.

As a result of this trend, organizations are recognizing new opportunities to generate traffic and revenue, often using existing components of their environments with minimal modification. In 2015, the Harvard Business Review reported that 90% of Expedia's revenue came through APIs. eBay and Salesforce also generated much of their revenue this way, reporting 60% and 50% API-driven revenue, respectively.³⁸

F5 Networks' [2019 State of Application Services \(SOAS\) report](#) found that of 2,000 technology professionals who responded to the survey, 42% had already deployed API gateways, with 27% planning to deploy gateways in 2019.³⁹ The API information portal and community forum [programmableweb.com](#) listed 12,000 APIs in its directory in 2015, and in early 2019 listed nearly 22,000, illustrating the rapid growth in their use.⁴⁰

How Does an API Fit Into a Network?

The term API refers simultaneously to the process, network location and data structure through which applications (or sub-application functions) can connect to one another and communicate autonomously. It provides everything that a developer needs to set up automated data transfers in pursuit of functional integration. An API gateway is a piece of lightweight software, running on an application server, that coordinates and manages API traffic.⁴¹ Ideally, API gateways should have some form of access control, and they typically support a range of authentication, authorization, and federation services such as OAuth, JWT, OpenID Connect, and SAML. The API gateway authenticates the requesting device and validates that its request or payload is structured correctly for the software that sits behind the API. This allows other applications to use the output of the original service in a different way, in a different app, or in a different environment, without having to recreate the original service from scratch. This is what makes APIs such a great way to scale and embed functionality into other apps.

What Makes an API a Good Target?

A combination of factors make APIs rich targets. One of the biggest issues is overly broad permissions. Because they are not intended for human use, APIs are often set up to access any data within the application environment. Permissions are usually set up for the user making the original request, and these permissions are, in turn, passed to the API. That is all well and good until an attack bypasses the user authentication process and goes directly to the downstream app. Because the API has unrestricted access, attacks through the API provide attackers with visibility into everything. Like basic web requests, API calls incorporate URLs, methods, headers, and other parameters. All of these can be abused in an attack. In fact, most typical web attacks, such as injection, credential brute force, parameter tampering, and session snooping work surprisingly well. To attackers, APIs are an easy target for all their old, dirty tricks.

Another issue is visibility. As an industry, we haven't often maintained a lot of situational awareness of APIs. They are supposed to run behind the scenes; this is great until they are compromised behind the scenes, and all of our valuables get stolen behind the scenes. APIs often connect to ports other than 80 and 443, they are frequently buried in deep paths somewhere on web servers, and the details of their architecture are often known only by development teams. All of this adds up to the reality that security teams may be unaware that connections with that potential impact are even possible in their environment.

What Kinds of APIs Get Hacked and Why?

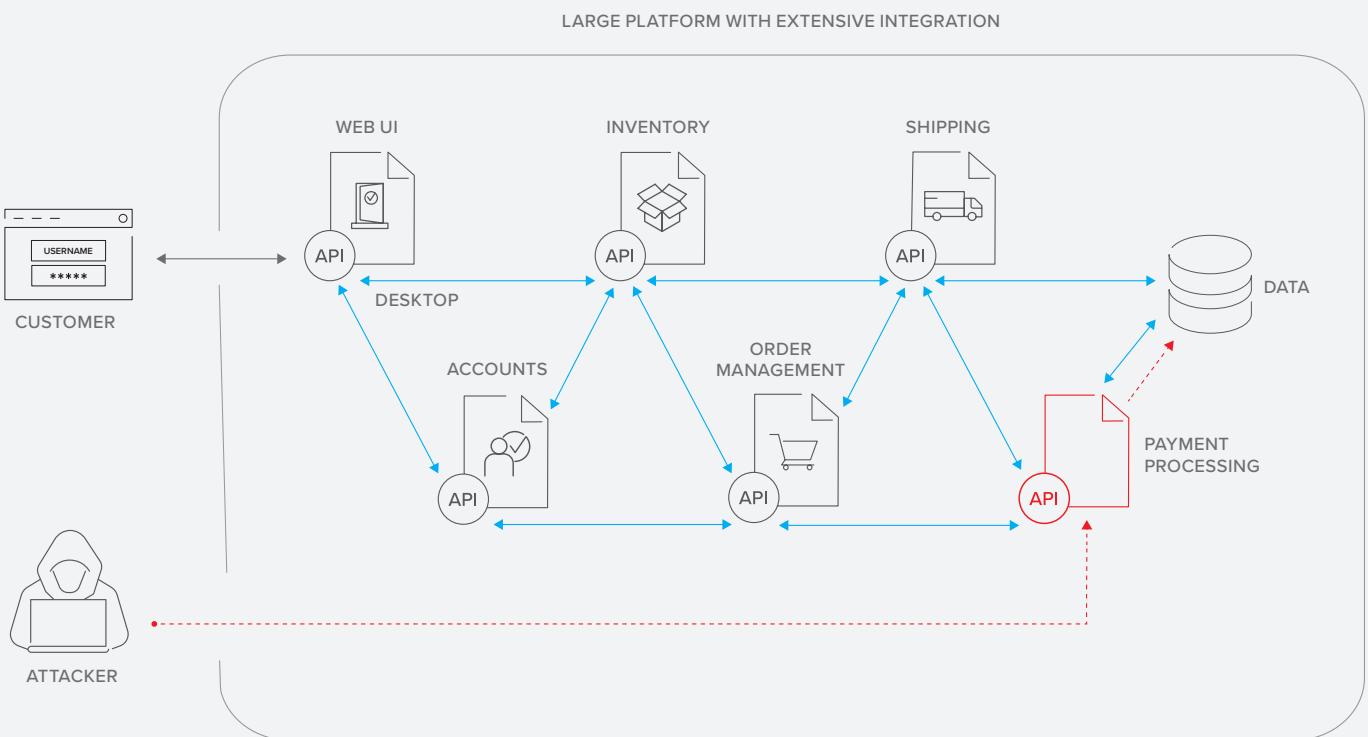
In a 2018 F5 Labs article entitled [Reviewing Recent API Security Incidents](#), we found that API compromises tended to fall into two patterns of API use that correspond to common breaches. We have since identified a third emerging pattern, which we will discuss as we explore the landscape for successful API breaches.

LARGE PLATFORMS WITH LOTS OF THIRD-PARTY INTEGRATIONS

FIGURE 14
LARGE PLATFORM API BREACH

This large web platform with a theoretical microservices architecture depends heavily on APIs for communication and integration between functions. An attacker could exploit a vulnerability or a simple lack of access control.

Organizations with high traffic sites offering a wide range of services often feature a large number of third-party integrations. These integrations rely on APIs to collect data from third parties and serve it up to the user in a seamless fashion. The growing decentralization of infrastructure, represented by multi-cloud environments, third-party functions and content, and serverless and containerized architectures means that APIs are not a luxury. For modern high-volume, complex application platforms such as Netflix and Facebook, they are essential to everyday operations. Some of these apps have hundreds of APIs, all of which need to be managed and monitored. These kinds of organizations and business models have figured prominently in the API breach notifications we've seen.



MOBILE APPS

As we noted briefly earlier (and [in our series on mobile app security](#)), most mobile apps rely on APIs to pull data from servers, which allows the apps to use fewer resources on the devices themselves. Because of some of the inherent challenges with securing mobile applications, there is a vibrant community of attackers who decompile and reverse engineer mobile applications looking for vulnerabilities or opportunities, such as hardcoded credentials or weak access control. The API is often a focal point for these efforts.

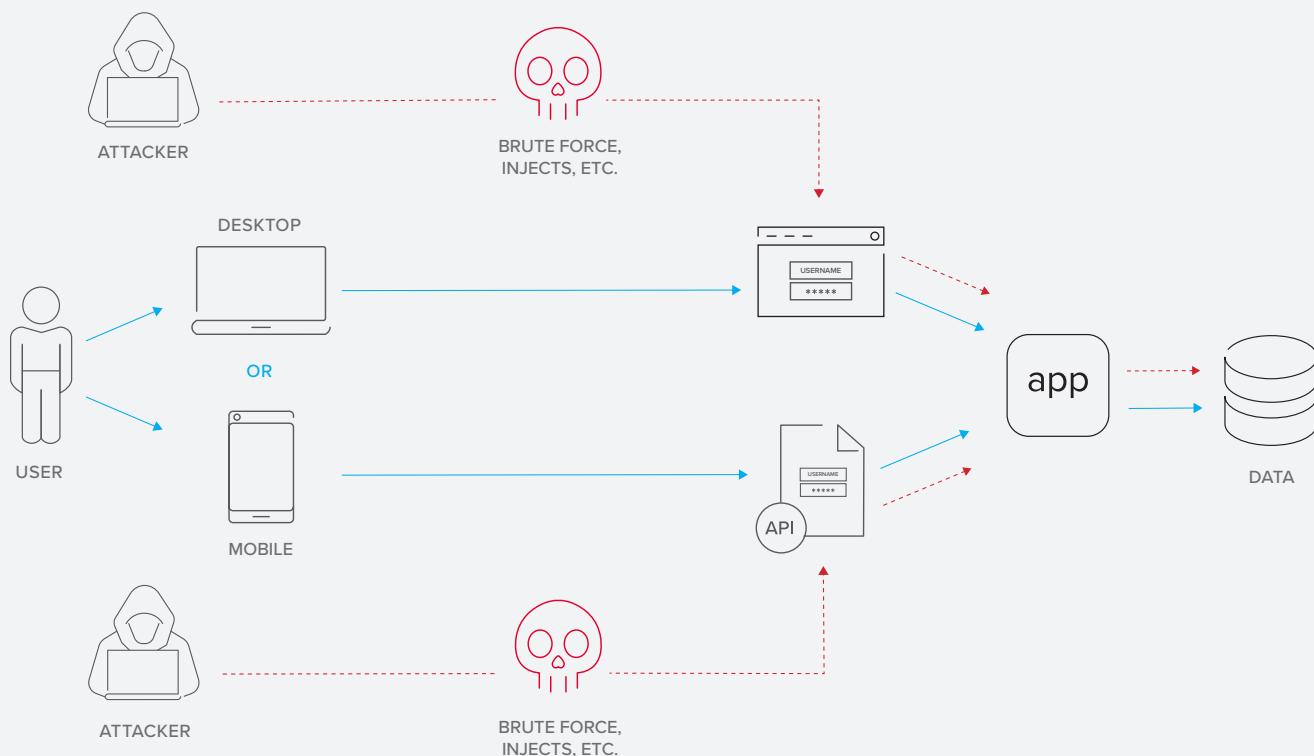
FIGURE 15

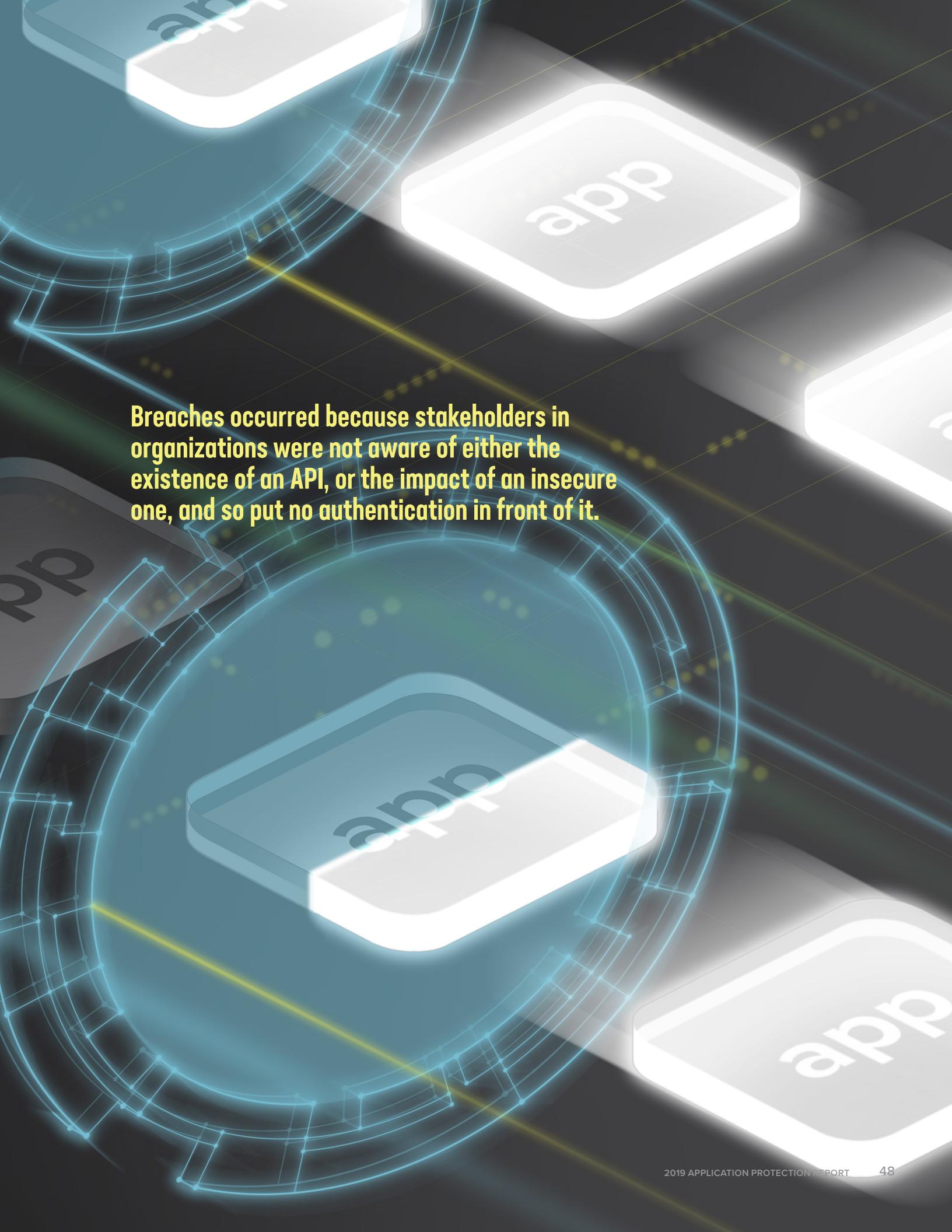
MOBILE APP API BREACH

An app routes mobile traffic through a mobile-specific API while desktop users connect to the same back-end through a web interface. The same tactics that would work against a traditional web application, such as brute force or injection, would work against the API.

THE MISCONFIGURED BIG APP

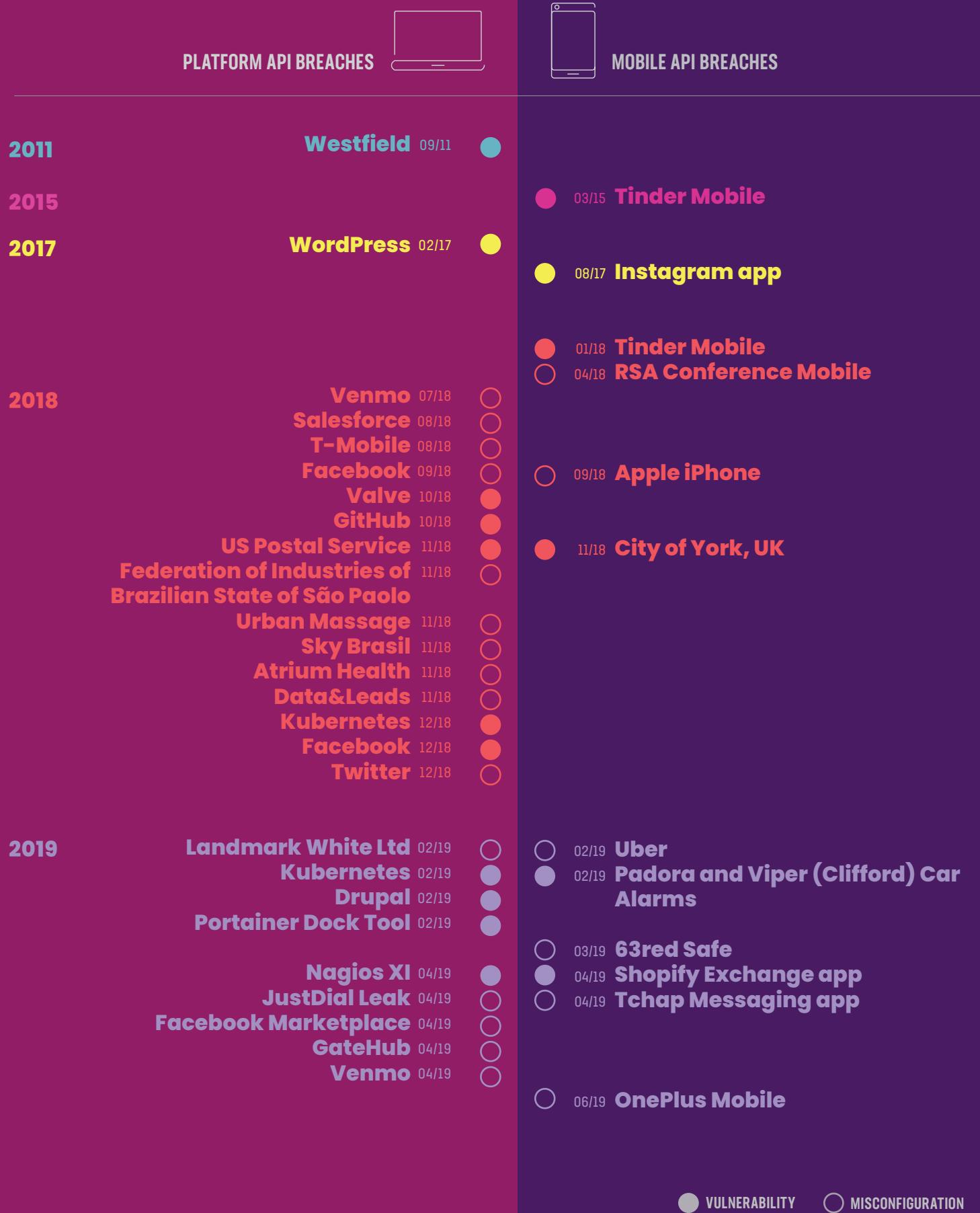
Both of those patterns—large apps with many third-party integrations and mobile apps—were common in breaches we reviewed in our 2018 report, but an additional emerging trend deserves mention: the misconfigured API. In many of the new cases we saw in the second half of 2018 and the first half of 2019, breaches occurred because stakeholders in organizations were not aware of either the existence of an API, or the impact of an insecure one, and so put no authentication (or weak authentication) in front of it. As silly as it sounds, it is hardly surprising, since it once again points to the fundamental challenge of visibility, both from the standpoint of information systems and large organizations.





Breaches occurred because stakeholders in organizations were not aware of either the existence of an API, or the impact of an insecure one, and so put no authentication in front of it.

FIGURE 16: API BREACHES TIMELINE



Public Records Review

All of the API breaches we identified fit into the first two patterns listed above—huge platforms with loads of APIs, or mobile apps that depend heavily on a few APIs to function.

BREACHES

In terms of new events, every single breach we've identified between November 2018 and August 2019 were attributable to misconfigured access controls. In other words, system owners did not realize that their APIs were wide open. However, as we noted earlier, the security researcher looking for a headline to put his or her name on was the principal threat actor in these scenarios, so the good news is that data were not leaked to the black market. You might say that these system owners got lucky, as did the researchers.

Another piece of evidence from the same time period that illustrated the prevalence and importance of APIs—as well as their weak security—was a study by North Carolina State University (NCSU). In this study, the university found that more than 100,000 code repositories on GitHub had API tokens and cryptographic keys—the literal tools for API access control—stored in plaintext and visible in plain sight.⁴² These findings illustrate a trend we've seen for a long time: developers using workarounds or insecure practices during development, when it doesn't matter, and then failing to mitigate those issues when the project goes live. Storing API authentication information in plaintext on GitHub is the latest incarnation of this old issue.

Incidents (Not Breaches)

The list of API incidents that did not lead to a breach had the same characteristics as the confirmed breaches. For the most part, these incidents were caused by security researchers finding routine bugs that had escaped detection precisely because they were in APIs.

Mitigating the Risk of API Attacks

Because the techniques that attackers use against APIs are similar to other web application attacks, the tactics of mitigation are generally similar. However, the visibility challenges with APIs mean that organizations often neglect the basics, so our focus is just as much on implementing simple controls such as inventory, authentication, and logging.

INVENTORY

To start, as with all assets, you'll need an [inventory of your APIs](#). For some organizations, this is not as easy as it sounds. The large platforms that we mentioned earlier often depend on APIs to pull specific functions into their application, such as payment information processing, or linking to social media. The growing complexity of information systems, along with the accelerating rate of change, means that obtaining this inventory is not a trivial task. Maintaining the inventory over time can be even more difficult, because it is usually harder for people with other jobs to budget

time for ongoing, routine maintenance tasks than it is to get together for a single discovery session. Nevertheless, it is essential to keep the inventory up to date, since APIs are quickly becoming the new nexus of business logic and application architecture.

As of June 2019, we know of approximately 22,000 published APIs. The majority of those are estimated to be private APIs, which means that access to them is restricted to internal developers or specific partners, as opposed to the public.⁴³ Understanding which ones are which within your organization, and how each one contributes to your business operations is important for adding context to your inventory. This information allows you to begin the process of assessing the risks that APIs might bring to your environment. This is important because all the breaches we've seen so far have been attributable not to sophisticated zero-day exploits by genius hackers, but instead to failures to apply basic security principles to new operational needs.

There are two specific things we recommend for every API inventory, reflecting the role of APIs in contemporary systems:

- 1) Conduct perimeter scans. Perimeter scans are valuable particularly because they replicate the hacker's eye view.⁴⁴**
- 2) Perform in-depth discovery interviews with development and operations teams. This may save time and get you valuable information to supplement the results of internal scans. Find out what the de juris and the de facto states are, and prepare risk assessments for both.**

API AUTHENTICATION

By design, APIs execute commands. They should be considered root/admin interfaces and require strong authentication. F5's State of Application Services 2019 found that 25% of the organizations surveyed didn't deploy authentication for APIs at all.⁴⁵ Thirty-eight percent reported that they did "some of the time," and 37% said "most of the time." Two things stand out about these responses. The first is that a full quarter of F5's customer base is not controlling access to a component that offers a direct path right to the heart of some of their most valuable capital. The other is that even the leaders in this issue are still not controlling access all of the time. This is a prime example of risk models failing to keep up with a changing landscape.

There are different forms of API authentication; practitioners must consider the pros and cons of each type based on risk.⁴⁶ Generally speaking, OAuth 2.0 is considered the best option for most REST APIs, but it depends on the nature and design of the API, its intended uses, and the business model and threat model of the organization.⁴⁷ (Note also that OAuth 2.0 authentication traffic must be encrypted to be effective.)⁴⁸

There are also emerging frameworks for service-level authentication in modern architectures such as the SPIFFE framework, which offer a potential step forward in terms of managing authentication in complex environments.⁴⁹ The real point here is that authentication of APIs is not optional given the trends and risk we've outlined.

Furthermore, for all authentication, credentials must be stored in a secure way. Whether your credentials take the form of user/password combinations (for either machines or human users) or API keys (which are simplified authentication strings that have specific uses), it is critical to treat them as sensitive information, because they really are the keys to the kingdom.⁵⁰

CREDENTIALS MUST BE STORED IN A SECURE WAY. IT IS CRITICAL TO TREAT THEM AS SENSITIVE INFORMATION, BECAUSE THEY REALLY ARE THE KEYS TO THE KINGDOM.

API AUTHORIZATION

The other half of API access control is authorization, meaning the permissions associated with any credential set. As with all credentials, API credentials must be treated using the principle of least privilege. Role-based access control is the best way to do this for API accounts of all types. At a minimum, this should take the form of limiting the HTTP methods that specific roles can implement (DELETE being an obvious one to limit, but not the only one—again, let your own environment and business logic dictate this in a risk-based way). To go one better, define specific sequences of actions that correspond to the specific API use case, and limit the API to that sequence instead of simply specifying a list of permitted actions.

At no time should APIs be allowed to pass unsanitized or unvalidated input to the application. That is a sure recipe for an [injection attack](#).

LOG THE API CONNECTION

After implementing authentication and authorization controls, the next level of maturity is to implement monitoring on the API. As we mentioned previously, brute force attacks can easily go unnoticed; all the more so with the advent of clever, distributed, low-and-slow access attacks. As a result, you should log API connections regardless of their outcome and behavior, and review those logs, whether through humans, bots, or cloud services. Furthermore, given the number of APIs and connections across them, it's best to also monitor the assets that the APIs serve to ensure their integrity and confidentiality.

ENCRYPT THE API CONNECTION

Even though API traffic is machine-to-machine, its traffic still moves across the web like any other traffic. We increasingly encrypt all user traffic on the web—API traffic should be treated the same

way. Encrypt those connections and validate the certificates like you would for any other service. That also takes care of the requirement to encrypt OAuth 2.0 authentication traffic, as noted earlier.

LOOK AT API SECURITY TOOLS

Consider looking at a proxy or a firewall that is “API aware” to inspect, validate, and throttle API requests. Some API security services can analyze the originating client and attempt to determine if a request is legitimate or malicious. They can also ensure that API requests stay where they’re supposed to stay, and do not escalate their privileges or exceed their reach into the app or data. A tool like this makes all of the aforementioned mitigation tasks easier. Many API security tools ingest OpenAPI/Swagger files, which will auto-configure the enforcement of the expected behavior.

TEST YOUR APIs

APIs must be scoped in for scans, vulnerability assessments, and penetration tests. The point here is that the prevalence of APIs is matched only by their obscurity. Many organizations don’t even have a single person who is aware of all of the organization’s APIs, and almost none of the organizations we’ve talked to have authentication on all of them. Testing is a part of any decent security program, and it applies here as well.

Given the difficulty of maintaining situational awareness, it is also a good idea to place a bug bounty on API vulnerabilities. Security researchers are constantly looking for things to report about, and Shodan has made it easy for researchers (and attackers) to discover your systems in detail. It is much, much cheaper to pay them quickly and quietly, patch your issues, and release a statement when applicable, than for researchers to go public—which could lead to bad press at best and a compromise at worst.



At no time should APIs be allowed to pass unsanitized or unvalidated input to the application. That is a sure recipe for an injection attack.

FIGURE 17: OVERVIEW OF ATTACK TYPES AND DEFENSE TOOLS

The techniques that played a significant role in the attacks that F5 Labs examined over 2018 and 2019, and how specific controls mentioned in the report can mitigate these risks. Note that this list of controls and attacks is not extensive; there are many more attack techniques available to attackers, and many more controls than those listed here. Also note that, as always, the implementation and maintenance of these controls is the most impactful determinant of their efficacy. No control can be set and forgotten about.

	Formjack Injection	Phishing	Brute Force	Cred Stuff	API Attacks
Change Control	Defined Change Process	✓			
	Monitoring unauthorized changes	✓	✓	✓	✓
	Subresource integrity monitoring	✓			
Vulnerability Management	Scanning	✓			✓
	Pen Testing / Red Team	✓	✓	✓	✓
	Patching	✓	✓		✓
	Bug Bounty	✓			✓
Encryption	Encryption/Transmission	✓			✓
	Encryption/Storage		✓		
	Inspection of encrypted connections		✓		
Inventory	Assets - systems & apps	✓	✓	✓	✓
	Assets - data	✓	✓	✓	✓
	Users and rights	✓	✓	✓	✓
	Inventory Audit	✓	✓	✓	✓
Access Control	Authentication		✓	✓	✓
	Authorization		✓	✓	✓
	Accounting/Logging	✓	✓	✓	✓
	Multifactor Authentication	✓	✓	✓	✓
	Human or bot challenge		✓	✓	✓
	Throttling/Rate Limiting			✓	✓
	Blacklist/Whitelist			✓	✓
	Continuous Authentication		✓	✓	✓
Web Application Firewall(or similar)*	Check PWs against dictionaries/dumps		✓	✓	✓
	Behavioral alerting/blacklisting	✓	✓	✓	✓
	Content security policy	✓			
	Traffic monitoring	✓	✓	✓	✓
	Sanitize Input	✓			
	Bot detection	✓	✓	✓	✓
	In-Browser Web Form Encryption	✓	✓		
	IP intelligence filtering/alerting	✓	✓	✓	✓
	Domain Monitoring		✓		
	Throttling/Rate Limiting			✓	
Threat Intelligence	API-specific content inspection				✓
	Threat intelligence data-feed	✓		✓	✓
	Monitor lookalike domains and certs	✓			
	Monitor user IDs in password dumps		✓	✓	✓
Training	Phishing Training		✓		
	Incident Response Rehearsal	✓	✓	✓	✓

* The WAF category is largely a category of convenience. While a firewall of any level of capability is not a control objective or control category, the growth in capability of WAFs and WAF-like service offerings (also referred to sometimes as Web Application and API Protection or WAAP) means that they offer many capabilities that would otherwise fit into other control objectives such as Access Control or Logging. Because WAFs were the only technical control that we recommended in the research series we have included them here as a top-level category despite the odd fit.

Final Thoughts

Security is a constantly shifting balance between critical details and the big picture. We need this balance to be able to place shifts in the landscape in context and understand the magnitude of their effects. With that in mind, here are some of the more general conclusions and predictions that have stood out in 2019.

PHP Recon Shows That We Can't Ignore the Trailing Edge

Baffin Bay's sensor networks revealed that unknown actors are using a small number of compromised systems on a university network to look for specific targets: old and probably neglected MySQL databases with weak authentication. These actors have defined a narrow set of target parameters, but are scanning the entire web from a small number of addresses—and are not trying too hard to cover their tracks. Aside from this specific threat, however, the sensor data indicate that unsophisticated campaigns against obsolete, obscure, or difficult-to-secure targets are omnipresent, and mitigating the risk they present should be considered table stakes for security practitioners.

The Big Picture with Breach Trends

Although open source data about actual breaches are not rich in detail, they do tell us enough to conclude that the most likely threat vector depends on where and how organizations store assets that malicious actors want. If, by dint of their business model, ecommerce-oriented organizations have to store personal information close to their application front ends, those front ends are going to become a focus for attacks, as they have in the various shopping cart campaigns we've seen over the last few years. If organizations store their valuables elsewhere on their networks, then social engineering and escalation of privileges are the order of the day—hence the rise in phishing campaigns.

We can draw a few broader conclusions from these data, which, for all their flaws, are valuable specifically because they are as empirical as data get in the security industry. For example, we can cross reference these empirical findings with the Open Web Application Security Project (OWASP) Top 10. Injection is the number one issue in the OWASP Top 10—and has been continuously since 2010. The breach data from 2018 confirm its ongoing relevance in the wild, despite being a known and solvable problem for many years.⁵¹ At the same time, broken access control is ranked fifth in the OWASP Top 10 2017, but it featured in nearly half of the 2018 breaches we examined.

We believe that this disparity stems not from any issue with OWASP's work or methods (which are enormously valuable), but from the bigger questions around what an application really is (spoiler: it's an onion). Applications are not just the code that they execute, but also everything around them that makes them tick: architecture, configuration, other assets to which the application connects,



Easy targets will remain popular. This means that unsophisticated campaigns against obsolete, obscure, or difficult-to-secure targets will remain prevalent.

and—not least—the user population. In other words, some things that have little to do with the narrow definition of a web application can have huge effects on that application’s security. The prevalence of access attacks (like phishing) in the breach reports are a good reminder of this bigger context. Cross-referencing our findings with the OWASP Top 10 is therefore a sign that we need to understand applications both at the level of the individual components that comprise them, and at the larger level of the entire environment in which humans use them.

The other conclusion we can draw from the industry breach profiles is that actual breaches validate risk-based security programs instead of best practices or checklists. If we know that successful attacks map with some precision to where target organizations store sensitive assets, it follows that organizations need to tailor controls and architecture to reflect the threats they actually face. This supports our long-held assertion that risk assessment needs to be a cornerstone of any security program, and the first step in any risk assessment is a substantive (and ongoing) inventory process. We realize this seems like common sense, but given the obstinate gap between theory and practice in our industry, it bears repeating.

The Future of Injection and Decentralized Web Content

Injection vulnerabilities are not new and mitigating them is theoretically simple. However, their enduring prevalence is not just because of the lag in mitigating or preventing these well-known flaws, or new, inexperienced developers recreating known issues in PHP.

Instead, injection remains such a problem because new trends are opening up new forms of risk. In other words, the injection landscape is not just sticking around, it is transforming along with our behavior. Detecting and mitigating injection flaws in light of these trends depends on adapting our assessments and controls to this new reality, not just fixing code.

Putting Access Attacks in Perspective

All of the data we’ve seen have shown that access tier attacks are one of the two most prominent types of attacks, both in terms of number of attempts and successful breaches. They present defenders with unique challenges in terms of visibility and graceful failure. The wide disparity in awareness and lack of suspicion within the user population means that there will always be a viable human to target, so while the tactics of access attacks will certainly change with technologies and defenses, the core principles will remain significant for the foreseeable future.

However, protecting applications against access attacks is especially problematic for a bigger reason. As we noted above, these tactics often place availability in direct conflict with confidentiality and integrity; the same processes whose weaknesses attackers are exploiting are the ones with which users directly interact. In other words, they drive to the heart of the conflict between the application protection team and the users, who now realize that they

cannot trust one another and do not necessarily have the same goals. The tools of protection, that is, MFA and service lockout, are antithetical to what the users want: free and easy access.

This trend, then, forces us security professionals to confront the question of whom a security program serves. Convoluted access controls, failing closed, and multiple forms of verification might serve the business' security goals, but at the cost of the value of the application to its audience. This means that, at a high level, access tier attacks threaten the value proposition of the Internet as a marketplace. From the standpoint of an individual security practitioner, there is much that you can do to control the current manifestation of this risk. We hope that the mitigation section above provides a strong platform for that. However, the underlying questions that access attacks pose are fundamental to the relationship between the digital world and the real one. How this specific part of the arms race evolves will determine much in terms of how we interact with internetworked information systems on the broadest possible level.

The Future of API Security

APIs are not new, but they are newly relevant for the way the Internet is growing and applications are evolving. As such, they do not so much introduce new risks as they reintroduce existing risks in forms that are more likely to be exploited, more impactful, and harder to recognize. At the same time, they are an unavoidable component in contemporary architectures, which means that avoiding or ignoring their issues is not an option for security teams. OWASP has released one of their excellent Top 10 lists [specifically for APIs](#). It is a good place to begin to explore these issues in greater detail.

Each episode of this research series has featured well-understood risks, many of which have been around for decades, but that have taken on new forms as a result of changes in the tools and techniques that we use to provide web services. While this is in keeping with our long-running assertion that successful security operations are mostly about the thorough implementation of the basics across space, time, and diverse systems, this is not to say that we think the changes are bad. When we talk about increased architectural complexity, this should be understood in light of the fact that applications also feature less complexity in the code itself. Other than for operating systems and similar heavy pieces of software, the days of millions of lines of code in a single piece of software are probably coming to an end. Instead, that complexity is reappearing in architecture and infrastructure, as we assemble many small parts to create a complete whole. There are many advantages to these trends, but they also introduce added layers of abstraction in operations, which, in turn, raises issues for visibility and cascading failure modes. This is why the old issues are coming back in new forms. As security practitioners, we must not lament these changes but adapt our practices to them with evidence, perspective, and a focus on enabling operations rather than hindering them.

Service Level Agreements Coming for Third-Party Services?

Over time, as new risks emerge from changing technology and from the information security arms race, we gradually incorporate those risks into our business models. Cloud computing has gradually shifted from a perceived bleeding-edge risk to a cornerstone of modern infrastructure. The risks associated with the cloud have either been mitigated or displaced to contractual risk in the form of service level agreements and audits. In other words, as the business world comes to grips with new trends in service provision, risks gradually morph from purely technical exploits that are managed reactively to facets of a business model that are managed proactively.

We predict that the same will happen with the trend of third-party web functions and content. Organizations will begin to manage this risk in the form of security-oriented service level agreements. The mitigations we have suggested above are the beginning, an initial bulwark as the industry comes to terms with new trends. But as we digest the ramifications of this latest manifestation of the web, the management of these new risks will mature. Doubtless, injection will morph as well, and find new ways to trouble us. In the meantime, we hope that the perspective and practices above assist in managing the latest incarnations of these old risks.

We Want to Know What You Think

The structure and content of our research is based on the broad and deep experience that our team brings to our work. However, in the end, our success hinges on whether our audience is able to apply this intelligence to mitigate risk. If you have feedback, data to share, requests for topics, or thoughts about our approach, please let us know. You can reach us on Twitter @f5labs, or email us at F5LabsTeam@f5.com.

ENDNOTES

- 1 For an in-depth review of the present state of vulnerability management, see Cyentia Institute's and Kenna Security's excellent report from Spring 2019 on Getting Read about Remediation.
https://www.kennasecurity.com/prioritization-to-prediction-report-volume-two/images/Getting_Real_About_Remediation.pdf.
- 2 https://w3techs.com/technologies/history_overview/programming_language/ms/y
- 3 https://www.w3schools.com/php/php_intro.asp
- 4 <https://www.exploit-db.com/exploits/18371>
- 5 We know, for instance, that phishing campaigns have been growing in both prevalence and sophistication over the last few years: <https://www.f5.com/labs/articles/threat-intelligence/2018-phishing-and-fraud-report--attacks-peaks-during-the-holidays>
- 6 <https://www.smbc-comics.com/?id=2526>
- 7 The high rate of formjacking attacks against the public sector is primarily attributable to the Click2Gov injection campaigns that targeted that sector in 2017 and 2018. <http://fortune.com/2018/12/18/click2gov-local-government-portals-hackers-credit-card-breach/>
- 8 <https://www.youtube.com/watch?v=hxLJExWk9GE>
- 9 kumarde.com/papers/tangled_web.pdf
- 10 <http://requestmap.webperf.tools/>
- 11 For example, see <https://www.exploit-db.com/exploits/44969>, which chains four separate injection vulnerabilities to gain a root shell.
- 12 See, among others, <https://nvd.nist.gov/vuln/detail/CVE-2018-1160#vulnCurrentDescriptionTitle>, as well as <https://medium.com/tenable-techblog/exploiting-an-18-year-old-bug-b47afe54172> for a more detailed explanation.
- 13 See <https://nvd.nist.gov/vuln/detail/CVE-2018-12613>. Also, in the exploit listed above, <https://www.exploit-db.com/exploits/44969>, the last three of the four vulnerabilities (2) (3) (4) are predicated on being authenticated to Nagios.
- 14 See <https://www.exploit-db.com/exploits/45925> and <https://nvd.nist.gov/vuln/detail/CVE-2018-1335> for examples of an injection exploit targeting a server.
- 15 See <https://www.exploit-db.com/exploits/46509>, <https://www.exploit-db.com/exploits/45858>, and many others
- 16 The Magecart attacks of the last eighteen months are an example of these third-party or supply chain attacks.
- 17 <https://www.bleepingcomputer.com/news/security/feedify-hacked-with-magecart-information-stealing-script/>
- 18 <https://duo.com/blog/malicious-hackers-take-over-media-sites-via-content-delivery-providers>
- 19 See <https://www.exploit-db.com/exploits/46539>.
- 20 See <https://www.exploit-db.com/exploits/44138>
- 21 See <https://www.exploit-db.com/exploits/43374> and <https://nvd.nist.gov/vuln/detail/CVE-2017-7411>
- 22 <https://nvd.nist.gov/vuln/detail/CVE-2018-12613>
- 23 <https://www.riskiq.com/blog/category/magecart/>
- 24 <https://www.exploit-db.com/exploits/39838>
- 25 See, for example, <https://www.exploit-db.com/exploits/46091>
- 26 <https://magento.com/security/patches/supee-8788>
- 27 Pompon, Ray. IT Security Risk Control Management. (New York: Apress, 2016). 170.
- 28 https://en.wikipedia.org/wiki/Content_Security_Policy
- 29 <https://www.troyhunt.com/protecting-your-embedded-content-with-subresource-integrity-sri/>
- 30 https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.md
- 31 For more information on monitoring domains, see David Warburton's [article on certificate transparency](#).
- 32 <https://cry.github.io/nbp/>
- 33 We have not been able to explore the SIRT's 2019 brute force data at time of writing.
- 34 <https://pages.nist.gov/800-63-3/>
- 35 <https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-password-ban-bad>

ENDNOTES

- 36 For a brief perspective on APIs and microservice architectures, see
<https://www.kuppingercole.com/blog/balaganski/api-security-in-microservices-architectures>
- 37 https://earthengine.google.com/case_studies/, <http://www.wonder-tonic.com/zombie/>
- 38 <https://hbr.org/2015/01/the-strategic-value-of-apis>
- 39 <https://www.f5.com/state-of-application-services-report/download-form>
- 40 <https://www.programmableweb.com/news/research-shows-interest-providing-apis-still-high/research/2018/02/23>
- 41 Not to be confused with AWS' API gateway, which is confusingly named API Gateway. For a non-AWS implementation of API gateways, see <https://docs.microsoft.com/en-us/azure/architecture/microservices/design/gateway>. For AWS' approach, see <https://aws.amazon.com/api-gateway/>
- 42 <https://www.zdnet.com/article/over-100000-github-repos-have-leaked-api-or-cryptographic-keys/>
- 43 <https://www.upwork.com/hiring/development/public-apis-vs-private-apis-whats-the-difference/>
- 44 Pompon, Ray. IT Security Risk Control Management. (New York: Apress, 2016). 170
- 45 <https://www.f5.com/state-of-application-services-report/download-form>
- 46 <https://blog.restcase.com/restful-api-authentication-basics/>
- 47 For more on OAuth 2.0 and some suggestions on implementation, see <https://oauth.net/2/oauth-best-practice/>
- 48 <https://aaronparecki.com/oauth-2-simplified/>
- 49 <https://spiffe.io/spiffe/>
- 50 For different types of API authentication, see <https://cloud.google.com/docs/authentication/>. For Google's take on securing API keys, (which is as good as any we've seen), see https://cloud.google.com/docs/authentication/api-keys#securing_an_api_key. For Google's implementation of OAuth 2.0 for service accounts, see <https://developers.google.com/identity/protocols/OAuth2#serviceaccount>. They also have an API authentication sandbox at <https://developers.google.com/oauthplayground/>.
- 51 https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf



APPLICATION THREAT INTELLIGENCE



US Headquarters: 401 Elliott Ave W, Seattle, WA 98119 | 888-882-4447 // Americas: info@f5.com // Asia-Pacific: apacinfo@f5.com // Europe/Middle East/Africa: emeainfo@f5.com // Japan: f5j-info@f5.com

©2020 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at [f5.com](#). Any other products, services, or company names referenced herein may be trademarks of the respective owners with no endorsement or affiliation, expressed or implied, claimed by F5. RPRT-SEC-F5LABS-01/20